МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проєкту (роботи)
магістра
(ступінь вищої освіти)

на тему **ВЕБСАЙТ ГІМНАЗІЇ №55 ІЗ ІНТЕЛЕКТУАЛЬНИМ ЧАТ-БОТОМ**

Виконав: студент 2 курсу, групи КНТз-512м
спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерні системи та мережі»

ШВЕЦЬ Г. К.
(ПРІЗВИЩЕ та ініціали)
Керівник ТЯГУНОВА М.Ю.
(ПРІЗВИЩЕ та ініціали)
Рецензент КАПЛІЄНКО Т. І.
(ПРІЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти магістерський

Спеціальність 123 Комп'ютерна інженерія

(код і найменування)

Освітня програма (спеціалізація) «Комп'ютерні системи та мережі»

(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

"24" жовтня 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ШВЕЦЯ Григорія Костянтиновича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Вебсайт гімназії №55 із інтелектуальним чат-ботом

керівник проєкту (роботи) к. т. н., доцент, ТЯГУНОВА Марія Юріївна

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від "24"жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) застарілий сайт гімназії, технології розробки вебсайтів

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

1) Аналіз інструментів створення сайтів;

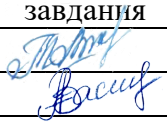
2) Проєктування вебсайта;

3) Реалізація вебсайту;

4) Результати досліджень;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

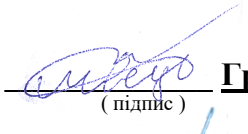
6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М.Ю. к.т.н., доцент		
нормоконтроль	ПОЛЬСЬКА О.В. ст. викл.		

7. Дата видачі завдання «_01_» жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз інструментів створення сайтів	06.10.2023р.	
2	Аналіз предметної області	10.10.2023р.	
3	Визначення вимог до розроблюваної системи	14.10.2023р.	
4	Проєктування сайту	24.10.2023р.	
5	Реалізація бекенду	03.11.2023р.	
6	Реалізація фронтенду	10.11.2023р.	
7	Створення та інтеграція чат-бота	17.11.2023р.	
8	Аналіз отриманих результатів	24.11.2023р.	
9	Оформлення пояснювальної записки	06.12.2023р.	

Студент(ка)  **Григорій ШВЕЦЬ**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)  **Марія ТЯГУНОВА**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 88 с., 50 рис., 9 табл., 18 джерел.

AXIOS, EXPRESS, HTTP-ЗАПИТИ, MARKDOWN MONGODB, NODE.JS, REACT, АВТОРИЗАЦІЯ, АУТЕНТИФІКАЦІЯ, БЕКЕНД, ВЕБСАЙТ, ФРОНТЕНД,ЧАТ-БОТ, ШИФРОВАННЯ

Об'єкт дослідження – вебсайт Запорізької гімназії №55 Запорізької міської ради.

Предмет дослідження – процес розробки вебсайту гімназії №55 із інтелектуальним чат-ботом.

Мета роботи – підвищення якості навчання у гімназії №55 за рахунок розробки вебсайту гімназії із інтелектуальним чат-ботом.

У першій частині проведено порівняльний аналіз інструментів для створення вебсайтів, конструктори, CMS, фреймворки. Для розробки вебсайту найбільш підходять фреймворки. Це пов'язано з тим, що вони надають широкий набір інструментів для створення вебсайтів з інтегруванням штучного інтелекту.

У другій частині проаналізовано предметну область, визначено вимоги до розроблюваного сайту. На основі висунутих вимог розроблено його структуру та макет. Аналіз предметної області дозволив визначити основні цільові аудиторії вебсайту.

У третій частині описано реалізацію вебсайту, як з точки зору бекенда, так і фронтенда.

У четвертій частині було наведено результати реалізації сайту та проведено дослідження його сприняття користувачами. Високий рівень зручності та інформативності був визнаний користувачами, які оцінили сайт та інтелектуального чат-бота як ефективний інструмент для отримання інформації.

ABSTRACT

Explanatory note to the master's work: 88 p., 50 figures, 9 tables, 18 sources.

AXIOS, EXPRESS, HTTP REQUESTS, MARKDOWN MONGODB, NODE.JS, REACT, AUTHORIZATION, AUTHENTICATION, BACKEND, WEBSITE, FRONTEND, CHATBOT, ENCRYPTION

The object of the study is the website of Zaporizhzhia Gymnasium No. 55 of Zaporizhzhia City Council.

The subject of the study is the process of developing a website for gymnasium No. 55 with an intelligent chatbot.

The purpose of the study is to improve the quality of education at gymnasium No. 55 by developing a website with an intelligent chatbot.

In the first part, we conducted a comparative analysis of tools for creating websites, including builders, CMS, and frameworks. Frameworks are the most suitable for website development. This is due to the fact that they provide a wide range of tools for creating websites with artificial intelligence integration.

In the second part, we analyze the subject area and define the requirements for the website under development. Based on the requirements, its structure and layout were developed. The analysis of the subject area allowed us to identify the main target audiences of the website.

The third part describes the website implementation, both in terms of backend and frontend.

The fourth part presents the results of the website implementation and analyzes its user acceptance. The high level of convenience and information content was recognized by users who rated the website and the intelligent chatbot as an effective tool for obtaining information.

ЗМІСТ

ВСТУП

Вебсайти є важливим інструментом для комунікації та взаємодії між людьми. Вони використовуються для різних цілей, таких як освіта, бізнес, розваги та ін.

Запорізька гімназія №55 Запорізької міської ради є освітнім закладом, який забезпечує якісну освіту для своїх учнів. Вебсайт гімназії є важливим інструментом для взаємодії між гімназією, її учнями, вчителями, батьками та громадськістю.

Розробка вебсайту гімназії №55 із інтелектуальним чат-ботом є актуальним дослідженням, оскільки воно дозволяє покращити якість взаємодії між гімназією та її екосистемою, надає учням, вчителям, батькам та громадськості доступ до актуальної інформації про гімназію. Розширить можливості для дистанційного навчання та спілкування.

Мета роботи полягає в підвищенні якості навчання у гімназії №55 за рахунок розробки вебсайту гімназії із інтелектуальним чат-ботом.

Для досягнення цієї мети необхідно виконати наступні завдання:

– проаналізувати засоби розробки вебсайтів;
– визначити вимоги до розроблюваного вебсайту;
– спроєктувати вебсайт;
– реалізувати вебсайт та інтелектуального чат-бота;
– провести дослідження якості розробленого вебсайту.

Об'єктом дослідження є вебсайт Запорізької гімназії №55 Запорізької міської ради.

Предмет дослідження – процес розробки вебсайту гімназії №55 із інтелектуальним чат-ботом.

Для досягнення поставлених цілей були використані наступні методи дослідження: аналіз аналогів, проєктування, розробка, тестування, опитування, анкетування.

Актуальність дослідження визначається зростанням популярності вебсайтів як інструменту комунікації та взаємодії, поширенням технологій штучного

інтелекту та потребою в покращенні якості взаємодії між гімназією та її екосистемою.

Новизна дослідження полягає в розробці вебсайту гімназії №55 із інтелектуальним чат-ботом. Такий вебсайт буде мати ряд переваг перед традиційними вебсайтами, в зручність використання, швидкість доступу до інформації. Можливістю отримання відповідей на запитання в режимі реального часу.

Теоретична значущість дослідження в тому, що воно буде сприяти розвитку теорії та практики розробки вебсайтів з інтегруванням штучного інтелекту.

Практична значущість дослідження полягає в покращенні якості взаємодії між всіма учасниками освітнього процесу гімназії №55.

1 АНАЛІЗ ІНСТРУМЕНТІВ СТВОРЕННЯ САЙТІВ

Вебсайти є важливим інструментом для бізнесу, освіти та особистого використання. Вони дозволяють людям ділитися інформацією, спілкуватися один з одним і здійснювати покупки.

Існує безліч інструментів для створення вебсайтів, які можна використовувати для створення вебсайтів. Деякі інструменти призначені для початківців, тоді як інші призначені для досвідчених розробників.

Програмні інструменти створення сайтів можливо розділити на 3 категорії: конструктори сайтів, CMS та фреймворки.

1.1 Конструктори сайтів

Конструктор сайтів – це зручний інструмент для легкого створення та редагування вебсайтів.

Він дозволяє швидко створювати ресурси, додаючи на сторінки різні блоки (галереї, слайдери, новини тощо) на основі готових шаблонів дизайну.

Крім того, дозволяє змінювати дизайн або створювати власний. Для ефективного використання конструктора сайтів не потрібні спеціальні знання або технічні навички. Також не потрібно встановлювати додаткове програмне або апаратне забезпечення [1].

Основний принцип роботи конструктора сайтів полягає у майже миттєвому створенні вебсайту, який потім можна вільно редагувати та наповнювати контентом. При необхідності блоки можна розміщувати на всіх сторінках сайту. Порядок блоків можна змінювати та налаштовувати різними способами. Зручно, просто та ефективно.

Звичайно, з такою кількістю доступних платформ може бути складно

вирішити, яку з них використовувати. Немає сенсу сліпо тестувати десятки конструкторів сайтів і вибирати найкращий для конкретної мети. Достатньо скористатися готовою інформацією про найкращі продукти у вашій ніші та обрати один з них.

При виборі п'ятірки найкращих конструкторів сайтів у дипломній роботі були взяті до уваги такі фактори:

– популярність конструктора сайтів, оскільки це гарний показник того, що він надійний і ефективний;

– конструктор сайтів повинен пропонувати широкий вибір шаблонів і функцій, щоб користувачі могли створити сайт, який відповідає їхнім потребам;

– простота використання, що є особливо важливим у використанні для початківців;

– ціна конструктора сайтів має бути доступною для користувачів з різним бюджетом.

Тому для аналізу у цій роботі було обрано наступні конструктори, проаналізовані за запропонованими у магістерській роботі критеріями (табл. 1.1).

Таблиця 1.1 – Порівняльна таблиця конструкторів сайту

Критерій	**Squarespace**	**Wix**	**Webflow**	**Shopify**
Легкість використання	Легка	Середня	Середня	Середня
Функціональність	Широкий спектр	Широкий спектр	Широкий спектр	Широкий спектр
Ціна	Від 16$ до 49$ на місяць	Від 4.5$ до 24.5$ на місяць	Від безкоштовного до 212$ на місяць	Від 32$ до 399$ на місяць
Підтримка	Є	Є	Є	Є
Складність	Легка	Середня	Середня	Середня
Масштабованість	Гарна	Гарна	Гарна	Гарна
Підтримка мобільних пристроїв	Так	Так	Так	Так
Швидкість і продуктивність	Гарна	Гарна	Гарна	Гарна
Безпека	Гарна	Гарна	Гарна	Гарна

1.1.1 Конструктор сайтів Squarespace

Послуги Squarespace [2] (див. рис. 1.1) коштують трохи дорожче, ніж пропозиції конкурентів. Однак на те є чудова причина: це справді один із найкращих конструкторів сайтів.

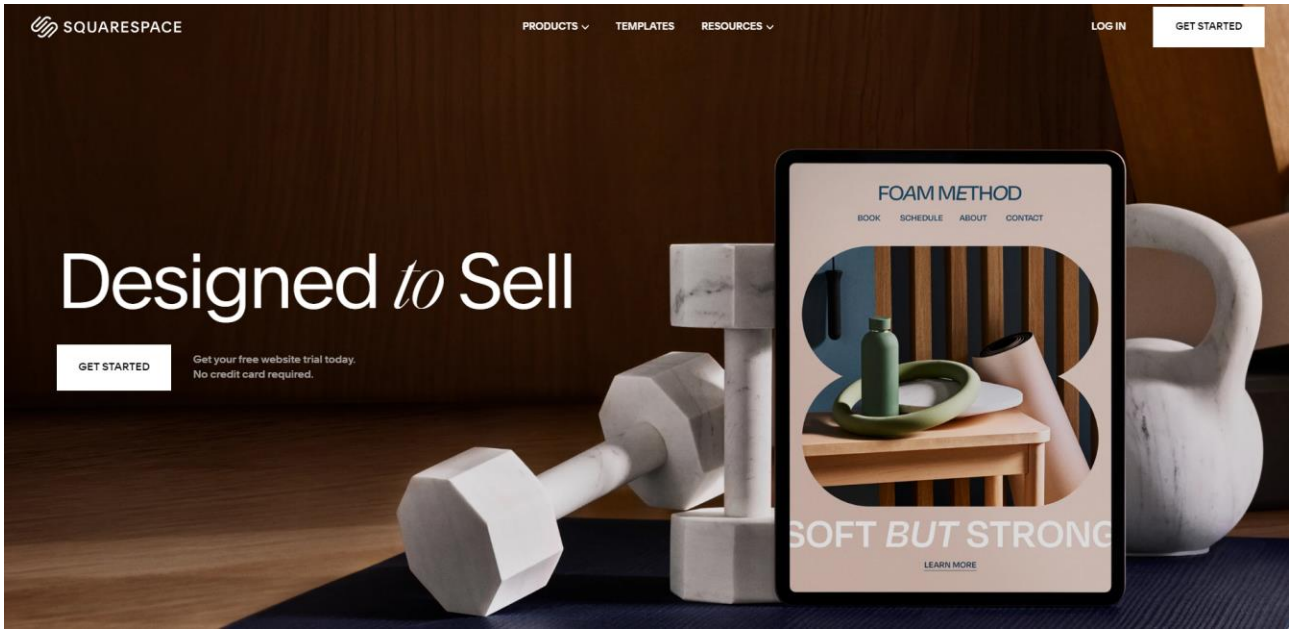


Рисунок 1.1 – Головна сторінка Squarespace

Під час налаштування вам буде доступно безліч опцій. Нехай навіть спершу їх може здатися занадто багато, у списку в лівій частині сторінки ви зможете знайти різні категорії, які допоможуть звузити пошук.

Зручний інтерфейс Squarespace дасть вам змогу одразу ж подивитися, який вигляд матиме ваш сайт із тим чи іншим шаблоном. При цьому "натягувати" шаблон на сайт вам для цього не доведеться.

У Squarespace дуже потужна вбудована вебаналітика. Вона покаже вам, як відвідувачі заходять на ваш сайт, як взаємодіють із ним, а також багато іншого. Все це, зрозуміло, дуже корисно для пошукової оптимізації вашого сайту.

Якщо у вас уже є обліковий запис Google Analytics, то ви можете інтегрувати його з Squarespace. Це дозволить вам відстежити, який ефект матимуть на продажі зміни вашого сайту.

Squarespace автоматично створює карту сайту для кожного створеного сайту.

Це дуже важливий момент, оскільки карта сайту допоможе пошуковим системам правильно проіндексувати ваш сайт.

Послуги Squarespace коштують дорожче, ніж аналогічні пропозиції конкурентів. Однак навіть на найдешевшому тарифі видно, що це того повністю варте.

Перший тариф називається Personal. Він дасть вам безлімітну пропускну здатність каналу і необмежений дисковий простір для зберігання файлів. Крім того, ваш сайт буде повністю оптимізований для мобільних пристроїв, ви отримаєте доступ до вебаналітики, зможете використовувати безплатний домен і отримати підтримку від цілодобово доступної техпідтримки.

Squarespace підійде для користувачів будь-якого рівня. Однак найкраще цей сервіс підійде тим, хто збирається створити сайт для інтернет-магазину.

1.1.2 Конструктор сайтів Wix

Щоб створити сайт на Wix[3](див. рис. 1.2), потрібно зареєструватися на сайті. Для цього вам потрібно буде ввести адресу електронної пошти та пароль. Після реєстрації вас буде перенаправлено на головну сторінку конструктора сайтів, де ви зможете почати створювати свій сайт.

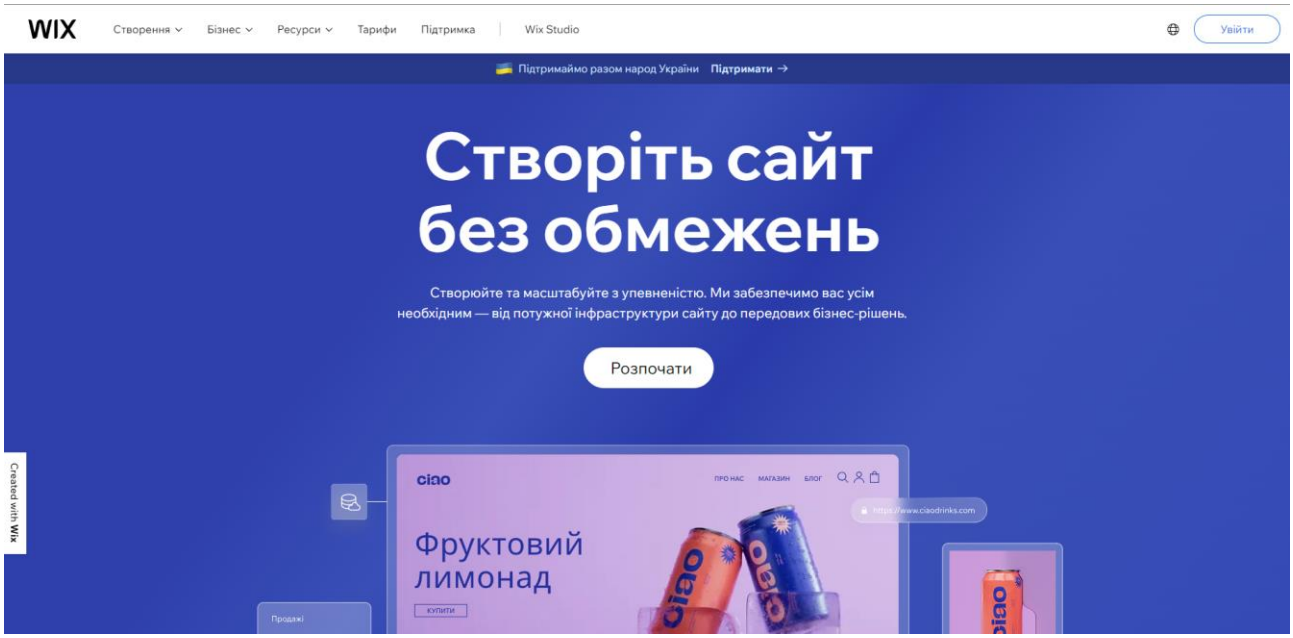


Рисунок 1.2 – Головна сторінка Wix

Першим кроком у створенні вебсайту на Wix є вибір типу вебсайту – Wix має широкий вибір шаблонів для різних типів вебсайтів, включаючи персональні сайти, бізнес-сайти та інтернет-магазини.

Після того, як ви вибрали тип сайту, оберіть шаблон – шаблони Wix розроблені професійними дизайнерами та програмістами і пропонують широкий вибір стилів та дизайнів.

Після того, як ви вибрали шаблон, ви можете налаштувати його відповідно до своїх потреб. Ви можете змінювати кольори, шрифти, фони та інші елементи дизайну.

Після того, як ви налаштували шаблон, ви можете додавати контент на свій вебсайт. Ви можете додавати текст, зображення, відео та інші типи контенту.

Важливо налаштувати SEO вашого сайту, щоб він добре ранжувався в пошукових системах; Wix пропонує ряд інструментів, які допоможуть вам налаштувати SEO.

Якщо ви хочете, щоб ваш сайт мав власний домен, ви можете додати його до вашого сайту Wix. Для цього вам потрібно буде придбати домен у реєстратора доменів.

Після того, як ваш сайт налаштований, ви можете опублікувати його. Після публікації ваш сайт буде доступний для відвідувачів в Інтернеті.

Переваги:

– один з найпростіших у використанні конструкторів сайтів;
– широкий вибір шаблонів для різних типів сайтів;
– шаблони Wix можна налаштувати відповідно до ваших потреб;
– різноманітні інструменти для налаштування SEO.

Недоліки:

– плани Wix досить дорогі порівняно з іншими конструкторами сайтів;
– деякі шаблони Wix занадто складні для початківців без досвіду в дизайні;
– деякі додатки в магазині Wix є неповними або не працюють належним чином.

Wix – це потужний конструктор сайтів, що пропонує широкий спектр функцій та інструментів. Це хороший вибір для користувачів, які хочуть створювати красиві та функціональні вебсайти без будь-яких навичок програмування.

1.1.3 Конструктор сайтів Webflow

Webflow [4] (див. рис. 1.3) – це єдиний адаптивний конструктор вебсайтів, який дає вам повну свободу творчості. Це чисте полотно, на якому ваша уява може розгулятися.

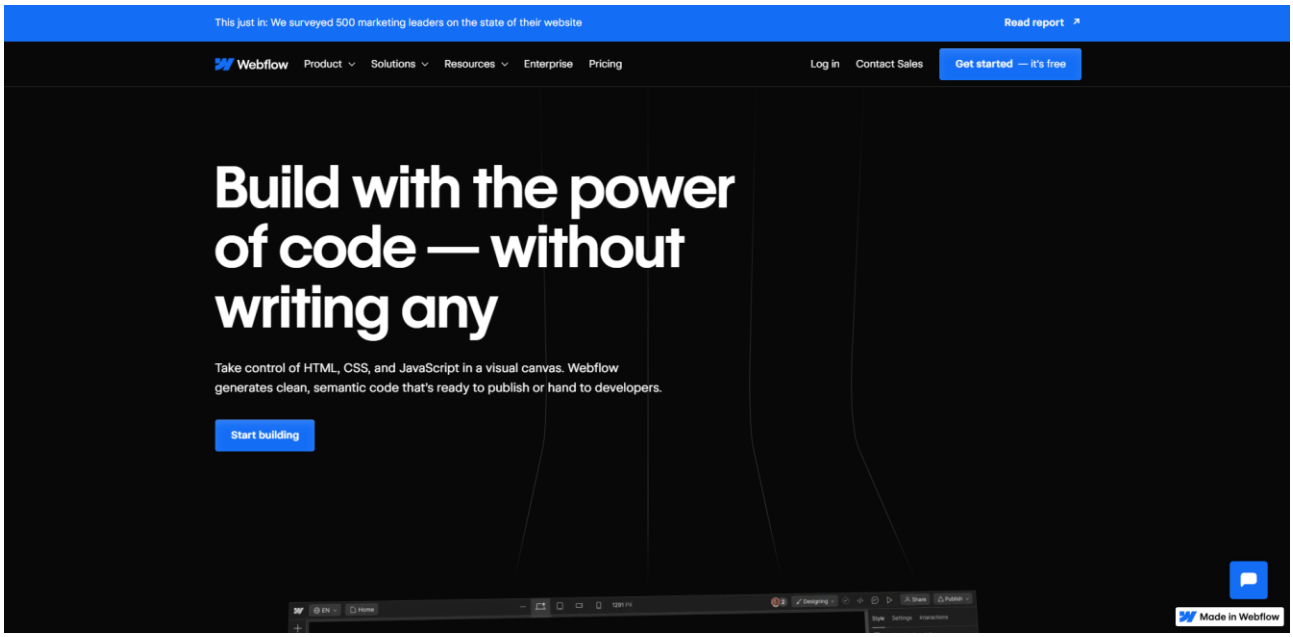


Рисунок 1.3 – Головна сторінка Webflow

Існує понад 100 адаптивних шаблонів вебсайтів на вибір. Використовуйте безкоштовний або платний адаптивний шаблон вебсайту, створений Webflow або одним із його надійних дизайнерів, щоб розпочати процес розробки. Клієнти та команди можуть використовувати інтуїтивно зрозумілий редактор вебсайтів. Немає необхідності опановувати складний бекенд із Webflow CMS. Просто запустіть свій вебсайт, увійдіть у CMS та почніть редагувати текст і фотографії.

Сайти Webflow реагують за замовчуванням, що означає, що вони відображаються та добре працюють на будь-якому пристрої.

Фільтри CSS можна застосувати до будь-чого, наприклад, додайте один або

кілька із восьми фільтрів CSS до будь-якого елемента вебсторінок Webflow всередині Designer, щоб створити приголомшливі ефекти.

3D-перетворення та анімація надає дизайну вебсайту абсолютно нову глибину за допомогою першого інструменту, який дозволяє генерувати 3D-перетворення CSS без необхідності писати код. Створюйте складні взаємодії та анімацію, не торкаючись рядка коду, не звертаючись до JavaScript.

Компонент вкладок у Webflow робить створення навігації простим і красивим. Компонент слайдера – дозволяє легко створювати чудову карусель із фотографій, інформації та іншого за допомогою HTML/CSS.

Webflow оптимізовано для пошукових систем, дає пошуковим системам все, що вони хочуть, прямо з коробки: чистий семантичний код, надшвидке завантаження сторінки та зручність для мобільних пристроїв. Автоматично генерує карту сайту XML.

Webflow використовує Tier 1 Content Delivery Network, Amazon Cloudfront і Fastly, щоб забезпечити найшвидший хостинг, доступний в Інтернеті.

Webflow пропонує різноманітні способи оплати, включаючи кредитні та дебетові картки, а також сторонні способи оплати, такі як Apple Pay, Stripe та PayPal.

Переваги:

– плагіни не потрібні;

– резервне копіювання та встановлення версій виконуються автоматично;

– 3D та анімація проста у використанні;

– шрифти в різних стилях;

– популярні можливості інтеграції програм;

– шаблони, які повністю налаштовуються;

– елементи, які можна перетягувати;

– миттєвий перегляд адаптивного вебсайту;

– альбомний або портретний режим на мобільному телефоні.

Недоліки:

– ціна вища, ніж у інших конкурентів Webflow;

– є скарги на погане обслуговування клієнтів;

– для тих, хто вперше розробляє сторінки, інтерфейс дизайну може здатися складним.

Webflow є чудовою альтернативою для людей, які шукають потужний, але зручний вебконструктор. Це потужний інструмент, а це означає, що оволодіння його розширеними можливостями займе деякий час. Webflow зробив багато ресурсів доступними для своїх клієнтів, зосередившись на досвіді та задоволенні клієнтів. Він пропонує докладні відеоуроки на додаток до текстового матеріалу, включаючи підібрані списки, які ведуть вас до тих, які будуть найбільш корисними для певної діяльності.

1.1.4 Конструктор сайтів Shopify

Shopify[5](див. рис. 1.4) – це платформа, яка дає інструменти для створення та ведення онлайн-бізнесу. Усі інструменти об'єднуються в одному місці: інтернет-магазин, складський облік, аналітика, CRM, системи лояльності, багатоканальний маркетинг (продаж через маркетплейси, соціальні мережі, email, контекстну рекламу тощо).

Shopify на зараз є найпопулярнішою ecommerce-платформою у світі. Загалом у світі запущено понад 2 000 000 магазинів. Він підійде тим, хто хоче продавати на українському та зарубіжних ринках, запустити інтернет-магазин максимально швидко, отримати гарний каталог товарів, об'єднати кілька каналів продажів в одному місці (Інтернет-магазин, Google, Instagram, і т.д.).

У Shopify App Store 4000+ додатків для розширення функціоналу. Більшість із них платні та поширюються на основі щомісячних платежів. Але є і безкоштовні, а також додатки з безкоштовними тарифами. Готові інтеграції з платіжними системами та сервісами доставки (понад 100 варіантів) для продажів у будь-якій країні світу.

Висока швидкість створення магазину на Shopify можлива завдяки великій кількості готових елементів, їхній злагодженості та можливості зробити простий

магазин у режимі конструктора (для чогось просунутого знадобиться розробник). У магазині шаблонів Shopify можна знайти дуже хороші варіанти шаблонів магазину. Досить буде завантажити свій контент і у вас вже готовий якісний магазин, який має чудовий вигляд.

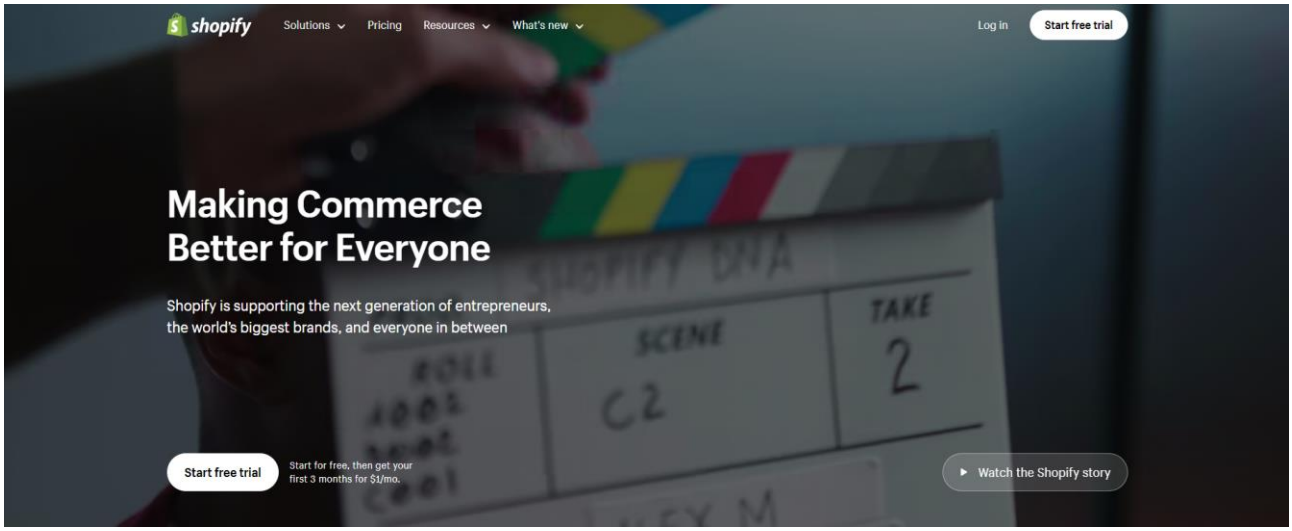


Рисунок 1.4 – Головна сторінка Shopify

Shopify – це хмарна платформа. Файли сайту розташовані на хостингу платформи і магазин перебуває ніби в оренді. Плюс, не до всіх файлів є доступ. Розширення функціоналу досягається здебільшого за рахунок платних додатків. Таким чином, щомісячна абонентська плата за магазин зростає. Але хороша новина в тому, що все основне і найнеобхідніше ви отримуєте безкоштовно. Сторінку "Оформлення замовлення" на звичайних тарифах редагувати не можна (така опція є тільки в Shopify Plus). До налаштувань хостингу доступу немає. Також доступу немає до багатьох системних файлів. Тому можливості оптимізації швидкодії сайту досить обмежені. І чим більше ви розвиваєте магазин, встановлюєте нові додатки, тим складніше мати швидкий сайт. Серед мов адмін-панелі немає української, доводиться працювати англійською.

Переваги:

– все для онлайн-торгівлі;

– основні інструменти онлайн-торгівлі об'єднані в одному місці: інтернет-магазин, складський облік, CRM, системи лояльності, аналітика, реклама;

– багатоканальність;

– керування рекламою та продажами в одному місці: продаж через маркетплейси, контекстна реклама, email-маркетинг, реклама в соціальних мережах тощо;

– додатки для розширення функціоналу;

– готові інтеграції;

– швидкий запуск магазину;

– велика спільнота розробників;

– якісні дизайнерські шаблони.

Недоліки:

– сайт не належить вам;

– за розширення функціоналу потрібно платити;

– не можна редагувати чекаут;

– обмежені можливості оптимізації сайту;

– високі комісії на приймання платежів онлайн.

Отже, Shopify має великі можливості для продажів на міжнародному ринку. Один із ключових плюсів платформи – можливість з одного місця керувати продажами через усі канали (сайт, маркетплейси, соціальні мережі, емейл, sms тощо). Також платформа зручна для швидкого створення магазину, враховуючи велику кількість готових компонентів.

Таким чином, проаналізувавши розглянуті вище конструктори сайтів можна виділити такі переваги та недоліки [6].

Переваги конструкторів сайтів:

– економія коштів;

– економія часу;

– безпека;

– оновлення та оптимізація.

Щоб розробити сайт на конструкторі не знадобиться команда спеціалістів зі створення дизайну, верстки тощо – все можна зробити самостійно. Зазвичай конструктори мають низький поріг входу і розраховані на початківців, тому

витрачати час на опанування веб програмування вам не доведеться. Створюючи сайт на конструкторі можна забути про питання, які пов’язані з налаштуваннями безпеки даних, працездатності сайту, шаблону, віджетів тощо, – все це бере на себе сервіс. Якщо ви виявите певний недолік, для його усунення необхідно лише написати у технічну підтримку. Оскільки на ринку представлено немало платформ для створення сайтів, конкуренція висока і мотивує розвиватися. Зазвичай сайти на конструкторах швидко завантажуються. І час від часу сервіси радують користувачів оновленнями функціоналу, віджетів, застосунків, шаблонів та ін.

Недоліками конструкторів сайтів є:

– однотипність сайтів;
– відсутня можливість перенести сайт;
– важче просувати;
– стороння реклама/оплата.

Дуже складно створити унікальний дизайн на конструкторі, використовуючи готовий шаблон, модулі та блоки. Саме можливостями глибокої кастомізації конструктори зазвичай обмежені. Навіть для професійних розробників буде складно кардинально змінити дизайн, адже у більшості конструкторів вихідний код закритий і максимум, що можна зробити – інтегрувати сторонній код. Конструктори розміщують сайт на своїй системі, тому не можна “переїхати” на інший хостинг або конструктор. Ви можете лише підтримувати сайт на вибраній платформі. Якщо вона припинить існування або просто перестане вам підходити – необхідно буде створювати сайт з нуля.

На жаль, сайти, які створені за допомогою конструктору часто платять за свою зручність і простоту саме можливостями з просування. Наприклад, на деяких платформах відсутня змога прописати метатеги. Ну і зазвичай пошукові системи у своїх рейтингах розміщують шаблонні сайти з конструкторів нижче від всіх інших. Багато конструкторів позиціонують себе як “безплатні” сервіси, проте як правило, на такому сайті буде розміщуватись реклама – це і є платою за обслуговування. Щоб прибрати рекламу доведеться оплатити тариф, і вони зазвичай недешеві.

1.2 Системи управління контентом

Система управління контентом (CMS) – це система управління вмістом вебсайту. З технічної точки зору, CMS також називають "движком вебсайту": За даними організації W3Techs [7], більше половини всіх вебсайтів в Інтернеті управляються за допомогою CMS.

На практиці, CMS – це вебдодаток, який дозволяє людям створювати та підтримувати вебсайт. Головна перевага CMS полягає в тому, що вона дозволяє людям створювати вебсайт без будь-яких знань з програмування.

Популярність CMS є хорошим показником того, що вона є надійною та ефективною. CMS повинна пропонувати широкий спектр шаблонів і функцій, щоб користувачі могли створити сайт, який відповідає їхнім потребам. CMS має бути легкою у використанні навіть для початківців. CMS має бути доступною для користувачів з різним бюджетом. Тому при виборі найкращої CMS було враховано наступні фактори:

– популярність;
– функціональність;
– простота використання;
– ціна.

Для аналізу у цій роботі було обрано наступні CMS, проаналізовані за запропонованими у магістерській роботі критеріями (табл. 1.2).

Таблиця 1.2 – Порівняльна таблиця CMS

Критерій	**WordPress**	**Drupal**	**Joomla**	**Magento**
Легкість використання	Середня	Середня	Середня	Складна
Функціональність	Широкий спектр	Широкий спектр	Широкий спектр	Широкий спектр
Ціна	Від $4 до $250 на місяць	Від $0 до $250 на місяць	Від $0 до $150 на місяць	Від $29 до $379 на місяць
Підтримка	Є	Є	Є	Є
Складність	Середня	Складна	Середня	Складна

Продовження таблиці 1.2

Критерій	**WordPress**	**Drupal**	**Joomla**	**Magento**
Масштабованість	Гарна	Гарна	Гарна	Гарна
Підтримка мобільних пристроїв	Так	Так	Так	Так
Швидкість і продуктивність	Гарна	Гарна	Гарна	Гарна
Безпека	Гарна	Гарна	Гарна	Гарна

1.2.1 Система керування контентом WordPress

WordPress [8] (див. рис. 1.5) – це найпопулярніша система управління контентом у світі на сьогоднішній день. На цій CMS працює понад 43% усіх вебсайтів в Інтернеті, і ця цифра постійно зростає.

Система управління контентом – це вебдодаток, який дозволяє власникам сайтів, редакторам і творцям контенту керувати своїми сайтами і публікувати контент без будь-яких знань програмування.

WordPress використовує PHP та MySQL і підтримується майже всіма хостинг-провайдерами.

Зазвичай цю CMS використовують для ведення блогів. Однак сайт на WordPress можна легко перетворити на інтернет-магазин, портфоліо, новинний сайт або будь-який інший тип ресурсу.

Однією з найважливіших особливостей платформи є її інтуїтивно зрозумілий і простий у використанні інтерфейс: Якщо ви знаєте, як користуватися Microsoft Word, вам не потрібно турбуватися про WordPress. Ви можете створювати та публікувати контент без жодних проблем.

А найкраще те, що платформа має відкритий вихідний код і є безкоштовною для всіх. Ця система управління контентом дозволила мільйонам людей по всьому світу створювати сучасні, якісні вебсайти.

WordPress починався як інструмент для ведення блогів і залишається найпопулярнішою платформою для цієї мети. Ви можете почати з малого з мінімальною ціною, наприклад, з простого стартового плану Hostinger. Потім, у міру зростання блогу, можливо перейти на більш складні рішення.

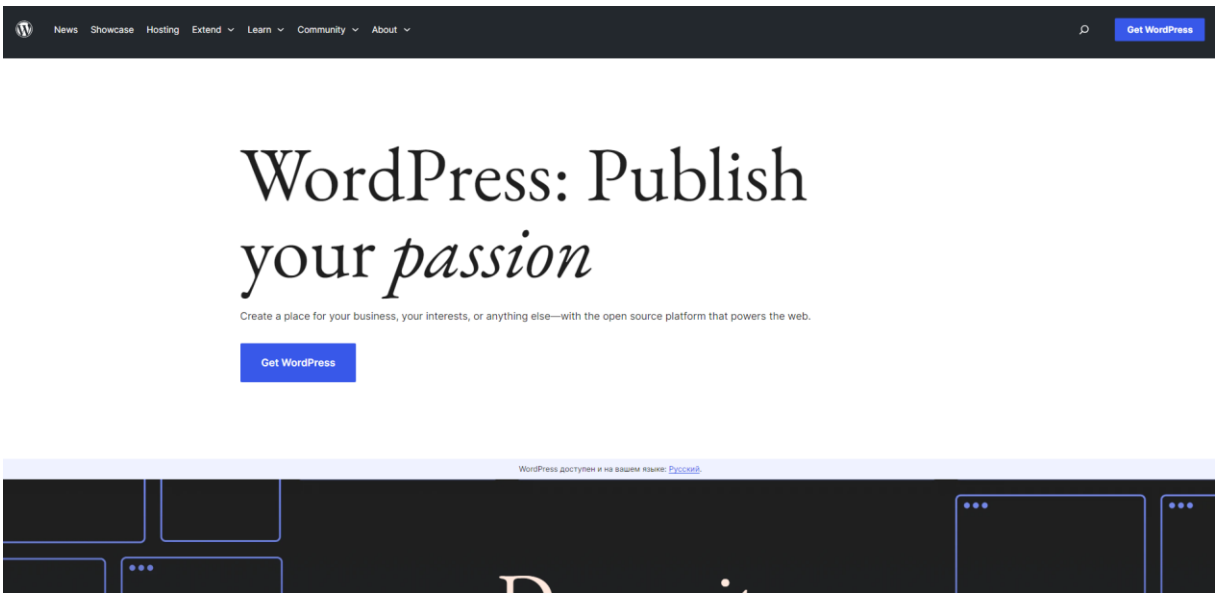


Рисунок 1.5 – Головна сторінка WordPress

Фрілансери, художники, фотографи, музиканти, письменники та інші творчі працівники часто використовують цю CMS для створення онлайн-портфоліо. Платформа має галерею, лайтбокси, сітки портфоліо та багато інших тем для портфоліо з усіма необхідними функціями.

Оскільки ця платформа не вимагає великих інвестицій для обслуговування і дуже проста в освоєнні, не дивно, що вона популярна серед благодійних організацій і церков; завдяки WordPress все більше і більше некомерційних організацій присутні в інтернеті в наші дні.

Якщо ви хочете створити власний інтернет-магазин, але нічого не знаєте про веброзробку, ця CMS – те, що потрібно WooCommerce – найпопулярніший плагін для електронної комерції для WordPress з кошиком для покупок, системою управління замовленнями, сторінками оформлення замовлення, поверненням коштів в один клік, а також ви можете додати інші важливі функції електронної комерції на свій вебсайт.

WordPress має багато переваг Це, безумовно, найкраща платформа для тих, хто хоче створити вебсайт без жодних знань з програмування. Однак вона також підходить і для програмістів, оскільки платформа має відкритий вихідний код і може бути модифікована відповідно до потреб будь-якого проєкту. Однак у CMS є деякі підводні камені, на які слід звернути увагу.

Переваги:

– низька вартість, тільки плата за домен і хостинг, більшість програмного забезпечення, плагінів і тем WordPress є безкоштовними;

– легко встановлювати та оновлювати на відміну від багатьох інших систем управління контентом, WP вимагає мінімальної установки і може бути оновлена одним кліком;

– легкість в управлінні для виконання повсякденних завдань, таких як створення та редагування постів, завантаження та редагування зображень, управління користувачами, додавання меню, встановлення плагінів і тем, не потрібні знання з програмування;

– індивідуальний дизайн з тисячами доступних тем можливо легко вибрати дизайн, який підходить саме вимогам проєкту, наприклад, є спеціальні теми для ресторанів, охорони здоров'я, малого бізнесу, гастрономічних блогів тощо;

– спільнота завжди готова допомогти – WordPress має велику світову спільноту і дуже корисний форум підтримки. Якщо виникнуть якісь питання або проблеми, можлово швидко знайти рішення;

– відкритий вихідний код на відміну від інших CMS, не потрібно платити за програмне забезпечення WordPress.

Недоліки:

– проблеми з безпекою оскільки ця система управління контентом лежить в основі понад 43% вебсайтів в Інтернеті, її дуже часто зламують, однак встановлення плагіна безпеки може значно зменшити ризик.

– сторонній контент багато плагінів і тем WordPress створюються сторонніми розробниками і можуть містити помилки, завжди читайте інструкції та коментарі перед встановленням нового плагіна або теми, а також запитуйте у спільноти;

– час завантаження сторінки, занадто багато плагінів може уповільнити завантаження сайту, встановлення плагіна кешування зазвичай вирішує цю проблему.

WordPress – неймовірно універсальна платформа. Ця CMS дозволяє створювати практично будь-які типи вебсайтів. Ви хочете створити блог – WordPress є чудовим рішенням. Ваш малий бізнес потребує присутності в Інтернеті. Знову ж таки, WordPress – ідеальне рішення для цього. Ви плануєте інтернет-магазин. Звичайно, WordPress. Варіантів незліченна безліч. Більше того, з ним легко почати роботу, а за потреби, за допомогою спільноти, ви зможете швидко знайти рішення та відповіді на свої запитання.

1.2.2 Система керування контентом Drupal

CMS Drupal [9] (див. рис. 1.6) використовується вже понад 20 років, а розробку цієї платформи розпочав у 2000 році бельгійський програміст Дріс Бейтарт. Він і досі є відповідальним за проєкт. Двигун був випущений на сайті drop.org у 2001 році; назва Drupal походить від голландського слова drupel, що перекладається як "крапля".

Drupal написаний на PHP і використовує реляційні бази даних (наприклад, MySQL, PostgreSQL).

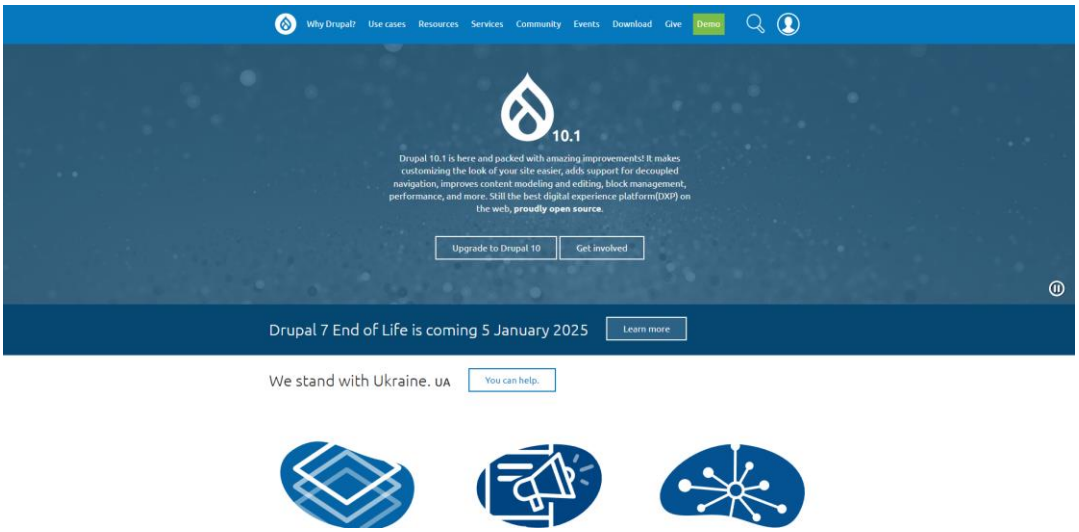


Рисунок 1.6 – Головна сторінка Drupal

Переваги:

– розповсюджується як вільне програмне забезпечення (ліцензія GPL), це означає, що платформу можна розвивати і налаштовувати, і немає прямої залежності від автора двигуна, проєкти з відкритим кодом також вважаються більш

захищеними, ніж проєкти з закритим кодом;

– має зручну для користувача архітектуру і дозволяє дуже гнучко налаштовувати вебсайти, це означає, що Drupal можна використовувати як платформу для всіх видів вебсайтів, від джерел новин, форумів та інших соціальних компонентів до інтернет-магазинів;

– можна використовувати як фреймворк, а також як CMS;

– як і багато інших платформ, має велику спільноту розробників, які створюють нові патчі, модулі та оновлення, а також документацію та посібники;

– має велику кількість модулів, які можуть значно розширити його функціональність і додати практично будь-яку можливість Друпал – це платформа, керована розробниками, а не користувачами (як це часто буває в інших CMS) це полегшує розробникам розуміння движка та налаштування потрібних їм функцій;

– має вбудовану систему кешування, яка зменшує навантаження на сервер і час завантаження сторінок;

– хуки дозволяють маніпулювати даними або виконувати інші дії під час їх обробки. Хуки пов'язані з процедурними точками, а не окремими об'єктами, і в Drupal ця модель дозволяє змінювати функціональність платформи без необхідності додавати або переписувати великі обсяги коду;

– розробники пишуть код в єдиному стилі, тому вони можуть швидко зрозуміти раніше написані модулі.

Недоліки:

– це платформа для досвідчених користувачів і розробників, налаштування модулів вимагає навичок програмування, тому потрібно витратити час на вивчення Drupal, перш ніж почати налаштовувати сайт;

– не найскладніший, але й не найпростіший або найлегший у використанні;

– після встановлення Drupal містить мінімальний набір інструментів, необхідних для правильної роботи сайту, таким чином, Drupal – це не готове рішення "встанови і працюй", а готовий фреймворк рішення, який потрібно розвивати за допомогою модулів;

– модулі використовуються для розширення функціональності двигуна – Drupal поставляється з декількома модулями при установці, але їх часто буває недостатньо для реалізації всіх побажань.

Популярність Drupal пояснюється, в тому числі, великою кількістю модулів, доступних на цій платформі. Щоб не загубитися серед величезної кількості модулів, існують фільтри за статусом розробки, категорією, версією ядра та іншими параметрами.

Drupal – це потужна платформа, яка дуже легко налаштовується. Вона не підходить для початківців, але досвідченим користувачам з навичками програмування вона сподобається. Двигун ідеально підходить для сайтів зі складною структурою даних (форуми, корпоративні сайти тощо).

1.2.3 Система керування контентом Joomla

За статистикою W3Techs на Joomla[10] працює близько 1,8% усіх сайтів світу. Щоб застосовувати CMS у своїх проєктах не потрібно нічого оплачувати або купувати. Почати можна абсолютно безкоштовно. Для багатьох завдань існують безкоштовні плагіни. У мережі навіть можна знайти безкоштовні шаблони від відомих розробників. Але робота сайту – це не тільки двигун. Це ще хостинг, домен, послуги розробників, витрати на просування і рекламу тощо. І якщо більшу частину завдань можна реалізувати своїми силами, то від оплати хостингу ніяк не уникнути.

Перший реліз CMS відбувся 2005 року, як мова програмування використовується PHP, але частина коду базується на фреймворку Symfony.

Joomla ідеально підходить для обслуговування великих новинних порталів, ЗМІ та інших корпоративних сайтів. За правильного компонування плагінів на Joomla можна запустити також блоги, інтернет-магазини і навіть форуми.

Двигун важко назвати доброзичливим. Він своєрідний. Передбачається, що користувачі мають достатній досвід і необхідні профільні знання, а також розуміння архітектури та знання особливостей устрою Joomla. Іншими словами – потрібен час для освоєння. Просто взяти і почати працювати швидше за все не вийде.

Joomla з коробки має велику кількість функцій. Спочатку інтерфейс панелі управління двигуна здається занадто складним і викликає багато запитань, але згодом цей ефект минає, і всі елементи управління виглядають більш ніж логічно. Це скоріше питання досвіду і звички.

Зараз Joomla – це скоріше CMF-система, на базі якої можна побудувати повноцінний web-сервіс, як і у випадку з Drupal. Тут сильний бекенд, який забезпечує роботу як повноцінної Headless (Decoupled) CMS.

Joomla не вимагає обов'язкового встановлення плагінів, щоб почати роботу з публікацією. Наприклад, з коробки є все необхідне для роботи з SEO (мета-теги, ЧПУ тощо), кешування сторінок, налаштування зовнішнього вигляду теми та виведення блоків контенту, для переходу на захищену HTTPS – версію тощо.

Joomla за своїми можливостями схожа на CMS для web-агентств – Drupal, тільки з більш "людським обличчям". Інакше кажучи, це комплексний фреймворк із готовою панеллю керування (CMF), у якому багато функцій можна перевизначити або відключити без необхідності програмування.

Joomla вміє перевіряти наявність нової стабільної версії ядра системи і розширень, присутніх в офіційному каталозі. Але CMS не оновлює код автоматично, обов'язково потрібна участь власника сайту. Тільки він може запустити процедуру оновлення.

У налаштуваннях сайту можна вибрати потрібний рівень стабільності – від Stable до Development (проміжні стадії включають, Alpha, Beta і Release Candidate).

Якихось вбудованих блокових конструкторів або спеціальних інструментів для роботи з шаблонами у Joomla немає. Тільки вбудований редактор вихідного коду (для обізнаних користувачів). Але разом з тим, зовнішній вигляд сайту на CMS Joomla можна дуже тонко налаштувати. Це реалізується за рахунок модулів і компонентів.

Переваги:

– модульна структура і можливість розширення за рахунок плагінів;
– багатомовність сайту з коробки;
– готове API;

– ЧПУ та SEO-атрибути без будь-яких плагінів;

– високий рівень безпеки;

– Joomla можна встановити практично на будь-який тип хостингу.

Недоліки:

– складні налаштування і публікація;

– немає готових мобільних додатків;

– з кожною новою версією двигуна втрачається сумісність з масою популярних плагінів і компонентів;

– багато розширень платні;

– без хостингу Joomla не працює;

– в офіційному каталозі повністю відсутні шаблони;

– з коробки немає набору перекладів.

Але все це мінуси для новачків. Професійним вебмайстрам Joomla точно сподобається, адже на цій системі сайт може розростися до вражаючих розмірів.

Joomla – крута і продумана система управління контентом, підходить для сайтів різної тематики, добре масштабується, має багато якісних інструментів із коробки. Але весь більш-менш важливий функціонал винесено в розширення, без них навіть блог не запустити. Самі розширення здебільшого платні або мають обмежену безкоштовну версію. Продаж плагінів – це ключове джерело доходу розробників.

Через особливості архітектури освоїти двигун новачкам буде складно, але цілком реально – є докладна документація, вбудована довідка, маса допоміжних матеріалів у мережі.

Останнім часом Joomla змістила акцент на користь більш просунутих користувачів, додавши Rest API і організувавши роботу за принципами web-сервісів. Разом з тим, система, як і раніше, невимоглива до хостингу.

1.2.4 Система керування контентом Magento

Magento [11] CMS має відкритий вихідний код. Вона написана на PHP (а конкретно використовує Zend Framework), а як базу даних використовує MySQL. Поширюється за ліцензією OSL 3.0.

Magento CMS має адаптивний вебдизайн, а отже, інтернет-магазин на цій платформі зручно переглядати на будь-яких пристроях, зокрема портативних. Це значно економить ресурси власника інтернет-магазину (як часові, так і матеріальні).

Її головною сильною стороною є велика кількість вбудованих функцій: йдеться про валюту, мови, знижки та купони, звіти та багато іншого. Крім вбудованого функціоналу Magento CMS має безліч модулів (або розширень) і різних шаблонів.

Magento CMS задумувалася як гнучка і багатогранна платформа. Конкуренцію можуть скласти хіба що WordPress з плагіном WooCommerce, PrestaShop і 1С-Бітрікс, але WordPress є скоріше блоговою платформою, PrestaShop складна в налаштуванні, а для використання 1С-Бітрікс необхідно купувати ліцензію. Але їхні вбудовані можливості не йдуть ні в яке порівняння з Magento CMS. Тому вибір цієї платформи для створення інтернет-магазину здається найбільш логічним і правильним, особливо для тих, хто має намір працювати із західною аудиторією, оскільки Magento CMS підтримує мультивалютність.

Magento CMS уможливлює створення одразу кількох інтернет-магазинів, а потім управління ними із загального центру, однієї адміністративної панелі, що, безумовно, дуже зручно. Ця можливість є жирним плюсом, який схиляє багатьох власників інтернет-магазинів до вибору саме цієї платформи.

У Magento CMS легко працювати з цінами – можна на певний час знизити вартість своїх товарів, дати знижку постійним клієнтам, влаштувати акцію. Іншими словами, під рукою буде все, що дасть змогу підвищити популярність магазину, а також збільшити продажі.

У Magento CMS реалізовано зручне управління, яке дає змогу розмежувати права в адміністративній панелі для різних людей: менеджерів, бухгалтерів, програмістів та інших. Також в інтернет-магазині на платформі Magento CMS багато функцій доступні і користувачам. Наприклад, вони можуть писати відгуки на товари і виставляти оцінки. Для просування в пошукових системах існує можливість оптимізації сторінки кожного товару. Magento CMS дає змогу створити XML-карту сайту, яка міститиме посилання на всі доступні сторінки інтернет-магазину. Ця карта необхідна для роботів пошукових систем для правильної індексації ресурсу.

Взагалі слід зазначити, що в CMS Magento є доступ до HTML і PHP-коду, тому можливостей для налаштування досить багато.

Порадує ця платформа і тих, хто любить уважно стежити за діями своїх товарів і клієнтів, оскільки дає змогу складати різні звіти: що відбувається з товарами (їхній рух, залишок), що приваблює відвідувачів на сторінках, яка сторінка є останньою перед тим, як вони підуть, і багато іншого.

До особливостей Magento CMS слід віднести те, що ця платформа досить вимоглива до хостингу, тобто для її використання обов'язково потрібен якісний хостинг-провайдер.

Під час роботи з Magento CMS пересічний користувач може зіткнутися з деякими складнощами (наприклад, якщо захоче зробити справді унікальний дизайн або додати певні платіжні системи) – у такому разі доведеться наймати сторонніх спеціалістів. Оскільки цей движок вимогливий до ресурсів, немає сенсу використовувати Magento CMS для зовсім невеликих інтернет-магазинів із кількома товарами.

Magento CMS – добротна і якісна платформа, яка чудово підходить для створення інтернет-магазину. У чомусь вона може бути складною для недосвідченого користувача (на відміну від конструктора сайтів), проте її можливості набагато ширші, ніж у інших схожих платформ. А якщо якихось функцій немає в базовій комплектації, то можна під'єднати додаткові модулі – наразі налічується понад чотири тисячі різних розширень.

Таким чином, проаналізувавши розглянуті вище конструктори сайтів можна виділити переваги та недоліки.

Перевагами є необмежені можливості. Можна реалізувати фактично будь-який ексклюзивний дизайн, створити яку завгодно структуру, інтегрувати різноманітні застосунки та плагіни. Відсутній ліміт на трафік та кількість сторінок – повна свобода дій. Також CMS дають повноцінні права. Сайт, всі дані на ньому, контент – це все належить вам повністю, а за необхідності ви можете без проблем змінити хостинг у будь-який час. Якщо на більшості конструкторів сайтів встановлена певна плата за користування, то тут можна вибрати повністю безплатну систему. Єдине за що все ж доведеться платити окремо – хостинг. Але на CMS вам будуть доступні всі можливості та відсутня стороння реклама. Сайти,

створені за допомогою CMS краще ранжуються пошуковими системами завдяки різноманітним можливостям SEO: метатеги, аналітика, оптимізація тощо.

Недоліками таких систем є необхідність спеціальних навичок. Для роботи з CMS у будь-якому випадку знадобиться хоча б початкове розуміння принципів роботи HTML/CSS, або ж допомога веброзробника. Навіть за умови, що ви вже володієте достатньою кількістю навичок у програмуванні, щоб створити сайт за допомогою CMS доведеться витратити більше часу, аніж, наприклад, на конструкторі. Зазвичай з усіма труднощами, які виникають на сайті, створеному на CMS, доводиться справлятися самостійно. Звісно, у мережі є досить інформації та інструкцій, але це все ж складніше, ніж просто написати у чат підтримки. Зазвичай сайти, створені за допомогою CMS потребують багато ресурсів, і завдання адміністратора сайту – слідкувати за тим, чи витримає обраний хостинг певне навантаження, щоб ненароком не зупинити роботу сайту.

1.3 Фреймворки

Фреймворк – це набір інструментів або бібліотека, що використовується для створення користувацького інтерфейсу для вебсайту або програми. Фреймворки допомагають веброзробникам створювати динамічні та інтерактивні користувацькі інтерфейси, які відповідають сучасним стандартам.

Фреймворки зазвичай використовують такі технології, як HTML, CSS та JavaScript. Вони також можуть включати такі функції:

– компонентний підхід;
– керування станом;
– маршрутизація;
– впровадження залежностей.

Компонентний підхід дозволяє розробникам створювати користувацькі інтерфейси з окремих компонентів багаторазового використання. Керування

станом дозволяє розробникам зберігати та керувати станом інтерфейсу користувача. Маршрутизація дозволяє розробникам створювати багатосторінкові вебсайти та додатки. Впровадження залежностей дозволяє розробникам легко імпортувати та використовувати сторонні бібліотеки та код.

Фронтенд-фреймворки корисні для веброзробників, які хочуть створювати складні та масштабовані користувацькі інтерфейси. Вони не тільки економлять час і зусилля розробників, але й забезпечують більшу функціональну сумісність і масштабованість. Нижче наведено порівняльну таблицю найпопулярніших фреймворків (табл. 1.3).

Переваги фреймворків:

– пропонують широкий спектр функцій і можливостей, які можуть бути використані для створення складних і масштабованих інтерфейсів користувача;

– легко масштабуються, що дозволяє створювати вебсайти, які можуть підтримувати великі обсяги трафіку;

– як правило, є швидкими і продуктивними, що забезпечує кращий досвід користувача;

– мають велике та активне співтовариство, яке надає підтримку та ресурси.

Таблиця 1.3 – Порівняльна таблиця фреймворків

Критерій	**React**	**Angular**	**Vue.js**	**Svelte**
Популярність	Найпопулярніший	Популярний	Популярний	Популярний
Функціональність	Широкий спектр	Широкий спектр	Широкий спектр	Широкий спектр
Легкість використання	Середня	Середня	Легка	Легка
Ціна	Безкоштовний	Безкоштовний	Безкоштовний	Безкоштовний
Підтримка	Є	Є	Є	Є
Складність	Середня	Середня	Легка	Легка
Масштабованість	Гарна	Гарна	Гарна	Гарна
Підтримка мобільних пристроїв	Так	Так	Так	Так
Швидкість і продуктивність	Гарна	Гарна	Гарна	Гарна
Безпека	Гарна	Гарна	Гарна	Гарна

Проте недоліками фреймворків є складність і їхня достатньо висока вартість. Фронтенд-фреймворки можуть бути складними у використанні для початківців і можуть бути дорогими, якщо ви використовуєте платні бібліотеки та інструменти.

Порівняльна таблиця інструментів створення сайтів представлена у табл.1.4.

Таким чином, можна зробити наступні висновки.

Конструктор вебсайтів є хорошим вибором для початківців, які хочуть створити вебсайт без необхідності вивчати кодування. Вони пропонують широкий спектр шаблонів і функцій, які можна використовувати для створення привабливих і функціональних вебсайтів. Однак конструктори вебсайтів можуть бути обмежені в функціональності і масштабованості.

CMS є більш потужними інструментами, ніж вебсайт-будівники. Вони пропонують широкий спектр шаблонів і функцій, а також можливість розширення за допомогою плагінів. CMS часто використовуються для створення корпоративних вебсайтів та вебсайтів електронної комерції. Однак CMS вимагають деяких навичок кодування.

Таблиця 1.4 – Порівняльна таблиця інструментів створення сайтів

Критерій	**Вебсайт-будівники**	**CMS**	**Фреймворки**
Легкість використання	Легкі у використанні, навіть для початківців	Досвід у кодуванні може бути корисним	Вимагають досвіду в кодуванні
Функціональність	Зазвичай пропонують широкий спектр шаблонів та функцій	Зазвичай пропонують більш широкий спектр шаблонів та функцій, а також можливість розширення за допомогою плагінів	Пропонують широкий спектр функцій та можливостей
Ціна	Багато вебсайт-будівників безкоштовні або пропонують безкоштовний пробний період	Багато CMS безкоштовні або пропонують безкоштовний пробний період, але можуть вимагати додаткових витрат на плагіни та розширення	Не завжди безкоштовні, але можуть бути доступні за низькою ціною
Підтримка	Багато вебсайт-будівників пропонують підтримку користувачів	Багато CMS пропонують підтримку користувачів	Зазвичай пропонують підтримку користувачів

Продовження таблиці 1.4

Критерій	**Вебсайт-будівники**	**CMS**	**Фреймворки**
Складність	Прості у використанні	Більш складні, ніж вебсайт-будівники	Найскладніші з трьох типів інструментів
Масштабованість	Можуть бути обмежені в масштабованості	Могуть бути масштабованими, але вимагають додаткових зусиль	Можуть бути масштабованими
Підтримка мобільних пристроїв	Зазвичай підтримують мобільні пристрої	Зазвичай підтримують мобільні пристрої	Зазвичай підтримують мобільні пристрої
Швидкість і продуктивність	Можуть бути повільнішими, ніж CMS або фреймворки	Можуть бути повільнішими, ніж фреймворки	Можуть бути швидкими і продуктивними
Безпека	Можуть бути менш безпечними, ніж CMS або фреймворки	Можуть бути менш безпечними, ніж фреймворки	Можуть бути безпечними
Приклади	Wix, Squarespace, Webflow, Shopify	WordPress, Joomla, Drupal, Magento	React, Angular, Vue.js

Фреймворки є найпотужнішими інструментами для створення вебсайтів і вебдодатків. Вони пропонують широкий спектр функцій і можливостей, які дозволяють розробникам створювати складні і масштабовані вебсайти. Однак фреймворки вимагають досвіду в кодуванні.

2 ПРОЄКТУВАННЯ ВЕБСАЙТА

2.1 Аналіз предметної області

Наразі гімназія №55 має застарілий сайт із мінімальним наповненням про заклад освіти. Тож сайт потребує оновлення, а можливо і повної переробки для задоволення сучасних потреб освітнього процесу.

Розробка вебсайту гімназії №55 із інтелектуальним чат-ботом дозволить покращити якість взаємодії між гімназією та учасниками освітнього процесу, а

також розширити можливості для дистанційного навчання та спілкування.

Для досягнення поставленої мети роботи вебсайт гімназії №55 повинен відповідати наступним вимогам:

– мати зрозумілий і логічний дизайн, систему навігації, яка дозволяє користувачам легко знаходити потрібну інформацію, кнопки та інші елементи керування повинні бути інтуїтивно зрозумілими та легкими в користуванні;

– оптимізований для швидкого завантаження, важлива інформація повинна бути легко доступна на головній сторінці або в інших видимих місцях;

– розроблений з використанням адаптивного дизайну, щоб добре виглядав і працював на різних пристроях з можливістю масштабування, щоб він міг адаптуватися до різних розмірів екранів;

– використовувати сучасні протоколи безпеки, такі як https, щоб захистити дані користувачів, мати систему виявлення та запобігання атакам;

– чат-бот повинен бути навчений на великому наборі даних, щоб він міг відповідати на широкий спектр запитань, здатний зрозуміти контекст запитань користувачів.

2.2 Структура вебсайту

Перед початком роботи над сайтом слід ретельно продумати його структуру. Це безпосередньо впливає на ранжування ресурсу в пошукових системах і його сприйняття користувачами. Розуміння правильної структури дозволить зрозуміти, чому два схожі за багатьма параметрами сайти матимуть дуже різні показники.

Структура сайту – це розташування сторінок, категорій, підкатегорій і товарів. Це своєрідний план, який показує логічні зв'язки між сторінками. З технічної точки зору, навігація по сайту – це набір URL-адрес, розташованих у певному порядку. Вона нерозривно пов'язана з семантичним ядром. Воно визначає [12] , які папки та документи мають бути на сайті.

Важливо розрізняти зовнішню і внутрішню структуру сайту. Перша стосується макета сторінок і розташування блоків на них. Друга відображає категорії та посилання на конкретні сторінки і документи. Про них ми поговоримо більш детально пізніше.

Якщо є семантичне ядро і є розуміння, який матеріал повинен бути на сайті, можливо побачити структуру ресурсів у вигляді схеми. Іншими словами, ієрархію, логічний і послідовний ланцюжок створення та подання інформації.

Тип діаграми залежить від призначення сайту. Виходячи з цього, можливо обрати тип структури, який найкраще відповідає потребам:

– алфавітна структура;
– хронологічна;
– деревоподібна структура;
– тематична;
– лінійна;
– сітка;
– павутинка;
– гібридна.

Алфавітна структура – інформація організована в алфавітному порядку. Це корисно, коли клієнти знають точну назву продукту, який вони шукають.

Хронологічна – найчастіше використовується для новин та прес-релізів і дозволяє здійснювати навігацію за датою публікації потрібного матеріалу.

Деревоподібна структура – коли є головна сторінка, а від неї вже відходять різні категорії, картки товарів тощо. Це популярна структура інтернет-магазинів, яку потрібно розробляти дуже ретельно.

Тематична – зручна навігація, якщо вся інформація організована за темами.

Лінійна – вся інформація організована в ланцюжок зі своїми складовими. При цьому немає можливості вільно переміщатися між сторінками, все йде за певним маршрутом. Це типовий варіант для візиток.

Сітка – всі компоненти розташовані в окремих гілках. Часто зустрічається в інформаційних каталогах статей.

Павутинка – композитна структура, яка поєднує в собі кілька типів. Однак через свою складність використовується рідко.

Гібридна – також комбінація декількох типів структури, яка дозволяє користувачам легше і швидше знаходити потрібну інформацію.

У кожного типу побудови можна знайти переваги і недоліки. Однак розробка ресурсу для максимальної ефективності повинна базуватися на наступних правилах:

Навігація повинна бути максимально зрозумілою для клієнта: всі сторінки каталогу повинні бути максимально оптимізовані та логічні, сайти повинні бути простими у використанні.

Загалом, правильно оформлена таблиця Excel дасть вам приблизне уявлення про структуру сайту. Однак для більшої наочності необхідно намалювати візуальну схему структури.

Правильна структура вебсайту (багато в чому!) гарантує його ефективність та успіх. Вона має безпосередній вплив на користувацький досвід і в поєднанні з семантичним ядром може досягти дивовижних результатів з точки зору SEO.

2.2.1 Основні сторінки сайту

Основні сторінки сайту – це розділи, які є найважливішими для користувачів і містять основну інформацію про сайт. Вони повинні бути добре видимі та легко доступні.

Сайт гімназії №55 є важливим інструментом для комунікації з учнями та їхніми батьками. Сайт повинен бути інформативним, зрозумілим і легкодоступним. Сторінка повинна містити інформацію, яка є важливою для користувачів і відповідає їхнім потребам. Сторінка повинна відвідуватися користувачами часто.

Тож визначити основні сторінки сайту можливо визначити з наступних критеріїв:

– важливість для користувачів;

– частота відвідування.

Гімназія може використовувати свій вебсайт для надання інформації про свої вимоги до вступу, результати навчання, інформацію про проведені заходи тощо. Це

може допомогти залучити потенційних учнів та їхніх батьків до гімназії.

Також вебсайт можна використовувати для збору зворотного зв'язку від користувачів. Це може допомогти гімназії покращити якість освіти. На основі цих критеріїв, можна визначити наступні основні сторінки сайту гімназії №55:

На основі цих критеріїв, були визначені наступні сторінки (див. рис. 2.1):

– головна;
– новини;
– контакти;
– особистий кабінет.

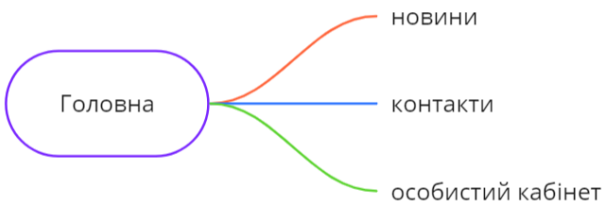


Рисунок 2.1 – Основні сторінки сайту

Головна – це основна сторінка сайту і містить загальну інформацію про гімназію, її історію, фотогалерею, останні новини, партнери та контакти. Ця сторінка є важливим для користувачів, оскільки він містить основну інформацію про гімназію. Вона також відвідується користувачами часто, оскільки це перша сторінка, яку користувачі бачать при відвідуванні сайту.

Новини – містить інформацію про новини та події, пов'язані з гімназією, її учнями, педагогічним колективом, подіями. Ця сторінка важлива для користувачів, оскільки вона дозволяє бути в курсі останніх новин та подій.

Контакти – ця сторінка містить інформацію про контактні дані гімназії, її розташування, години роботи. Вона дозволяє користувачам зв'язатися з гімназією.

Особистий кабінет – на цій сторінці користувачі можуть отримувати доступ до особистих даних такі як ім’я, аватар і зміна паролю.

Особливим буде те, що нові інформаційні сторінки та нові записи в новини

будуть додаватися адміністратором сайту. Це було зроблено з метою забезпечення точності та актуальності інформації, а також контролю контенту, який буде доступний користувачам.

2.2.2 Визначити структуру навігаційної панелі

Структура навігації сайту повинна бути простою та інтуїтивно зрозумілою для користувачів. Вона повинна дозволяти користувачам легко знаходити потрібну інформацію та взаємодіяти з сайтом.

На основі вищесказаної інформації сторінки розділилися на декілька груп і було вирішено розділити на наступну структуру навігації:

– головна – буде доступною з головного меню сайту;

– новини – буде доступною з головного меню сайту;

– контакти – буде доступною з головного меню сайту;

– особистий кабінет – ця сторінка призначена для користувачів, які зареєстровані на сайті. Вона буде доступною з головного меню сайту для користувачів, які авторизовані на сайті та попередньо ввійшли інакше буде запропоновано зареєструватися.

Крім того, у головному меню сайту буде присутній такий елемент навігації як «пошук». Цей елемент навігації дозволить користувачам швидко знайти потрібну інформацію на сайті. А починатися панель навігації буде з логотипу гімназії. (див. рис. 2.2)

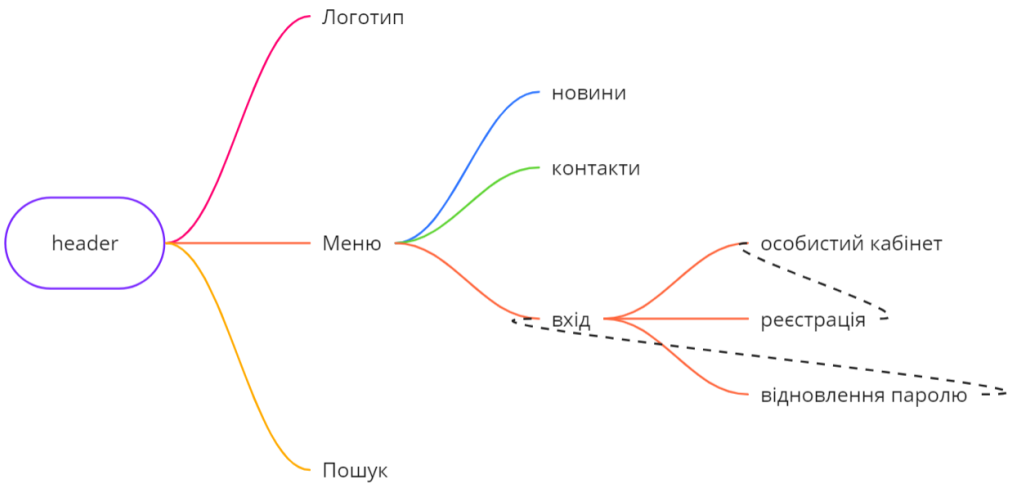


Рисунок 2.2 – Структура навігаційної панелі

Така структура навігації є простою та інтуїтивно зрозумілою для користувачів. Вона дозволить користувачам легко знаходити потрібну інформацію та взаємодіяти з сайтом.

2.2.3 Визначити структуру відображення контенту

Головна сторінка повинна містити основну інформацію про гімназію, а саме (див. рис. 2.3):

– про гімназію;
– фотогалерея;
– останні новини;
– наші партнери;
– контакти.

Сторінка "Новини" повинен містити інформацію про поточні події, пов'язані з гімназією. Ця інформація буде представлена у вигляді статей(див. рис. 2.4).

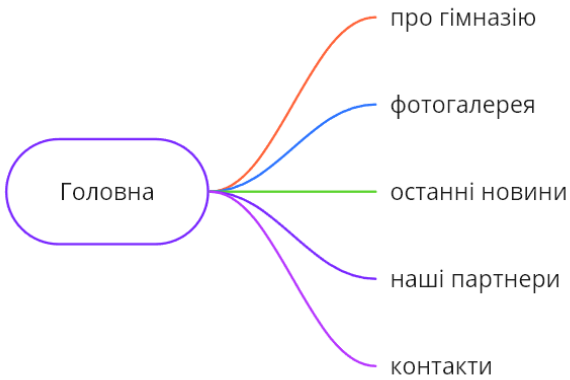


Рисунок 2.3 – Структура головної сторінки

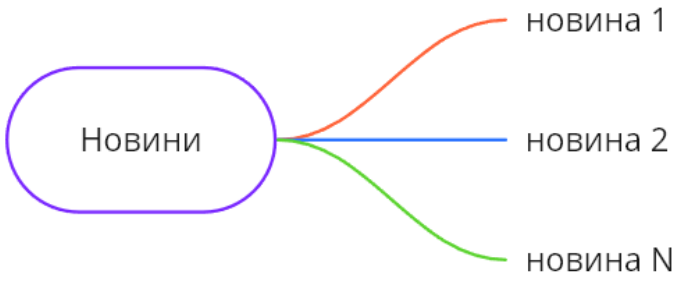


Рисунок 2.4 – Структура сторінки «новини»

Сторінка "Контакти" повина містити інформацію про контактні дані гімназії, її розташування, години роботи та інше (див. рис. 2.5).

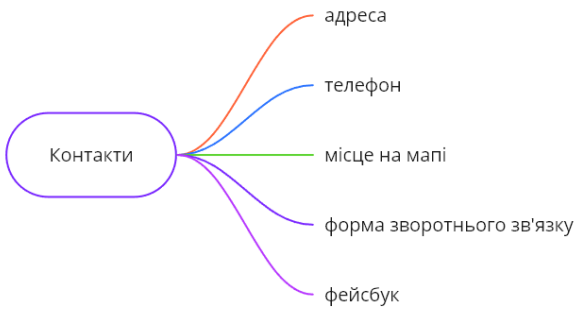


Рисунок 2.5 – Структура сторінки «контакти»

Сторінка "Особистий кабінет" Призначена для користувачів сайту. У цьому розділі користувачі можуть отримувати доступ до особистих даних такі як ім’я, аватар і зміна паролю. А адміністратори з особистого кабінету зможуть адмініструвати сайт. Тож особистий кабінет для користувачів буде мати такі дані (див. рис. 2.6) як: ім’я, аватар, зміна паролю.

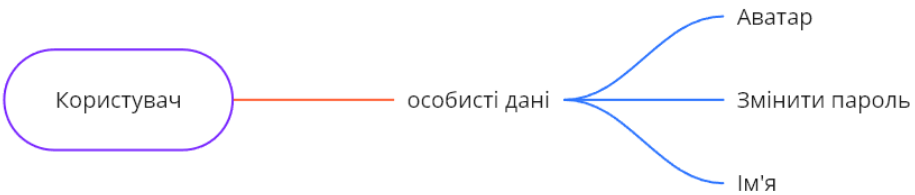


Рисунок 2.6 – Структура сторінки «особистий» від користувача

А особистий кабінет для адміністраторів буде мати наступні розділи:

– сторінки;
– новини;
– коментарі;
– альбоми;
– користувачі;
– особисті дані.

Детальніше на схемі (див. рис. 2.7)

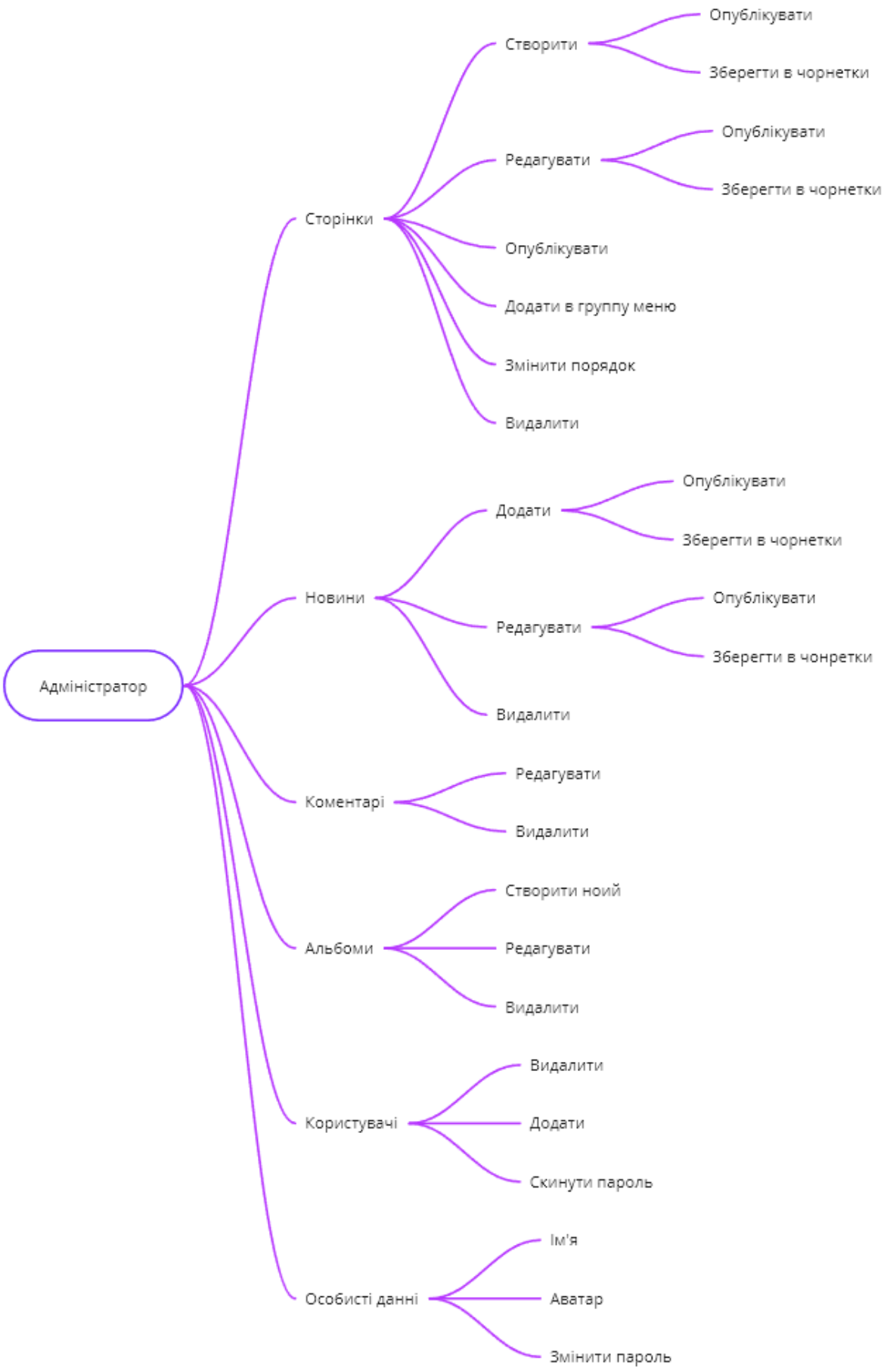


Рисунок 2.7 – Структура сторінки «особистий» від адміністратора

Інтелектуальний чат-бот є новим елементом вебсайту гімназії №55. Він призначений для того, щоб допомогти користувачам знаходити потрібну інформацію на сайті. Чат-бот може відповідати на запитання користувачів, надавати інформацію про гімназію, тощо.

Така структура відображення контенту забезпечує інформативність та зрозумілість вебсайту для всіх користувачів. Вона також дозволяє гімназії досягти своїх цілей, таких як інформування про діяльність гімназії, залучення потенційних учнів та батьків.

2.3 Створення макету сайту

2.3.1 Каркас

Каркас – це 2D-візуалізація цифрових продуктів, від найпростіших ескізів олівцем до повністю інтерактивних цифрових дизайнів.

Каркас можна розділити на 3 типи: Ескіз з низькою точністю, ескіз з середньою точністю і ескіз з високою точністю.

Ескізи сайту низької точності(див. рис. 2.8) використовуються на ранніх етапах розробки вебсайту. Вони використовуються для того, щоб швидко і легко створити загальний вигляд і структуру сайту. Ескізи низької точності не містять детальної інформації про дизайн, шрифти, кольори та інші елементи. За браком часу цей етап було пропущено.

Ескізи сайту середньої точності(див. рис. 2.9) використовуються на середньому етапі розробки вебсайту. Вони містять більш детальну інформацію про дизайн, шрифти, кольори та інші елементи, ніж ескізи низької точності. Однак, вони все ще не є точними, як макет сайту високої точності. Також цей етап зачасту пропускають із за браку часу на проєкт. На цьому етапі схематично було розміщено елементи на сторінці.

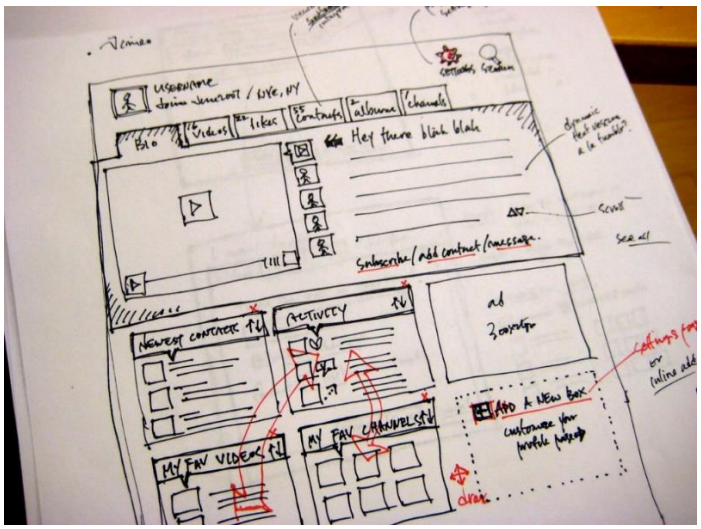


Рисунок 2.8 – Ескіз сайту низької точності

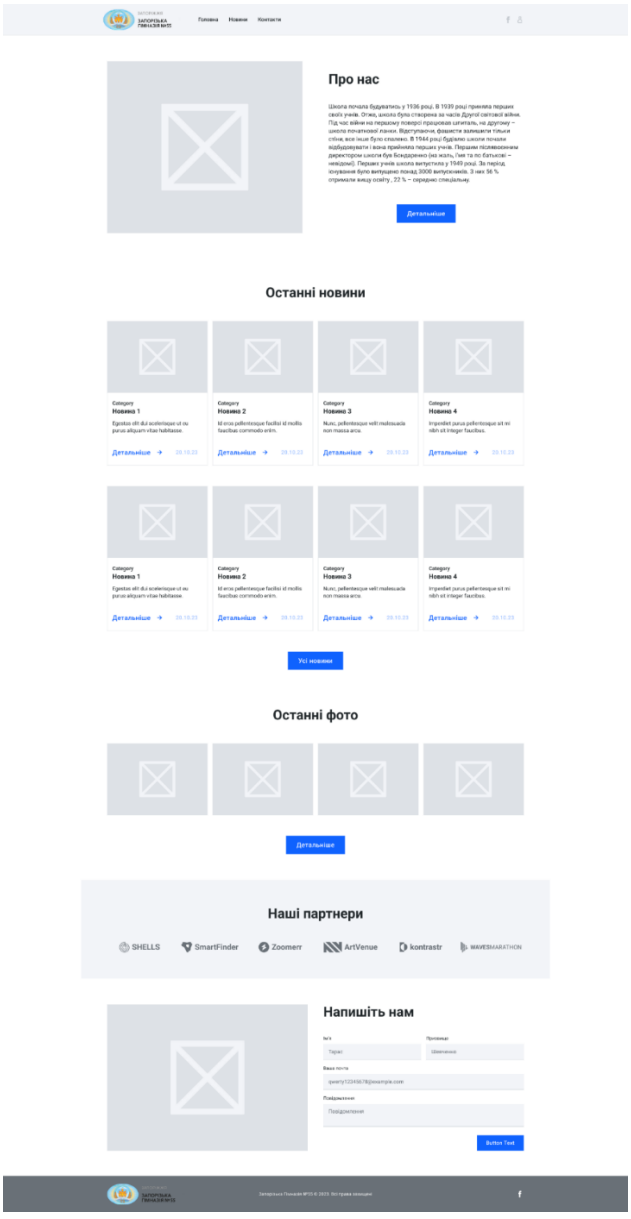


Рисунок 2.9 – Ескіз сайту середньої точності

Ескіз сайту високої точності – це детальний план дизайну сайту, який включає в себе всі елементи сайту, такі як текст, зображення, відео, кнопки, меню та інші.

Ескіз високої точності, який було розроблено, відповідає вимогам технічного завдання і містить у собі всі необхідні елементи для розробки вебсайту. Ось деякі конкретні приклади того, що можна побачити на ескізі високої точності:

На головній сторінці сайту можна побачити логотип компанії, навігаційне меню, а також основні блоки інформації, такі як новини, галерея, партнери, контакти та інше (рис 2.10).

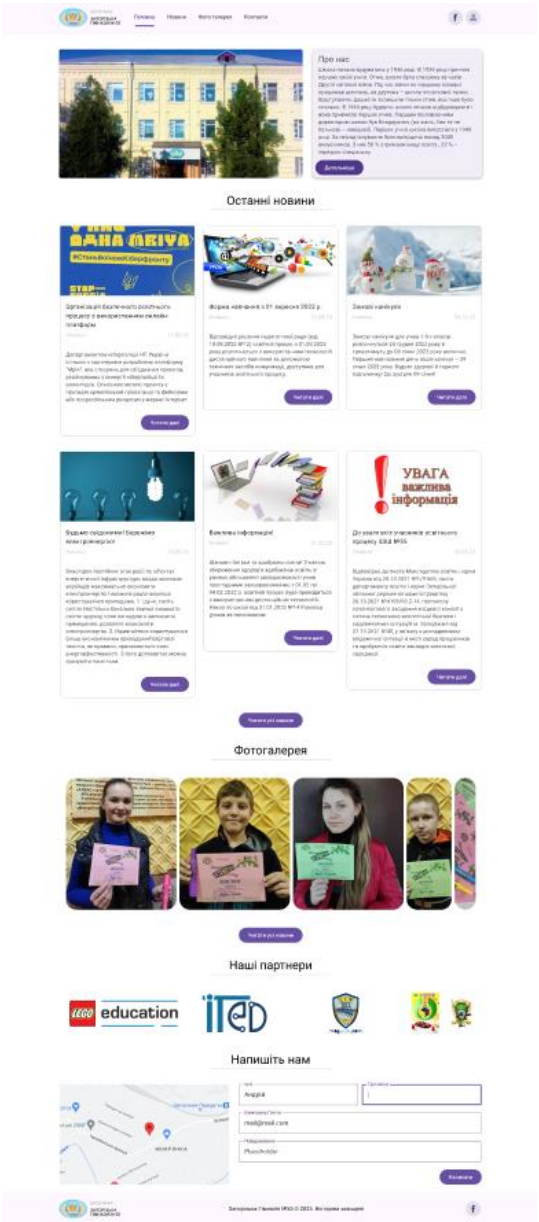


Рисунок 2.10 – Ескіз сайту високої точності

Ескіз високої точності – це важливий документ, який дозволяє точно визначити, як буде виглядати вебсайт. Він є основою для розробки вебсайту і дозволяє уникнути неприємних сюрпризів на пізніх етапах розробки.

2.3.2 Прототип

На етапі прототипування (див. рис. 2.11) ескізи перетворюються на інтерактивні демонстрації, які точно імітують зовнішній вигляд і поведінку продукту. Дизайнери використовують прототипи для проведення користувацького тестування та збору цінних відгуків про зручність використання продукту.

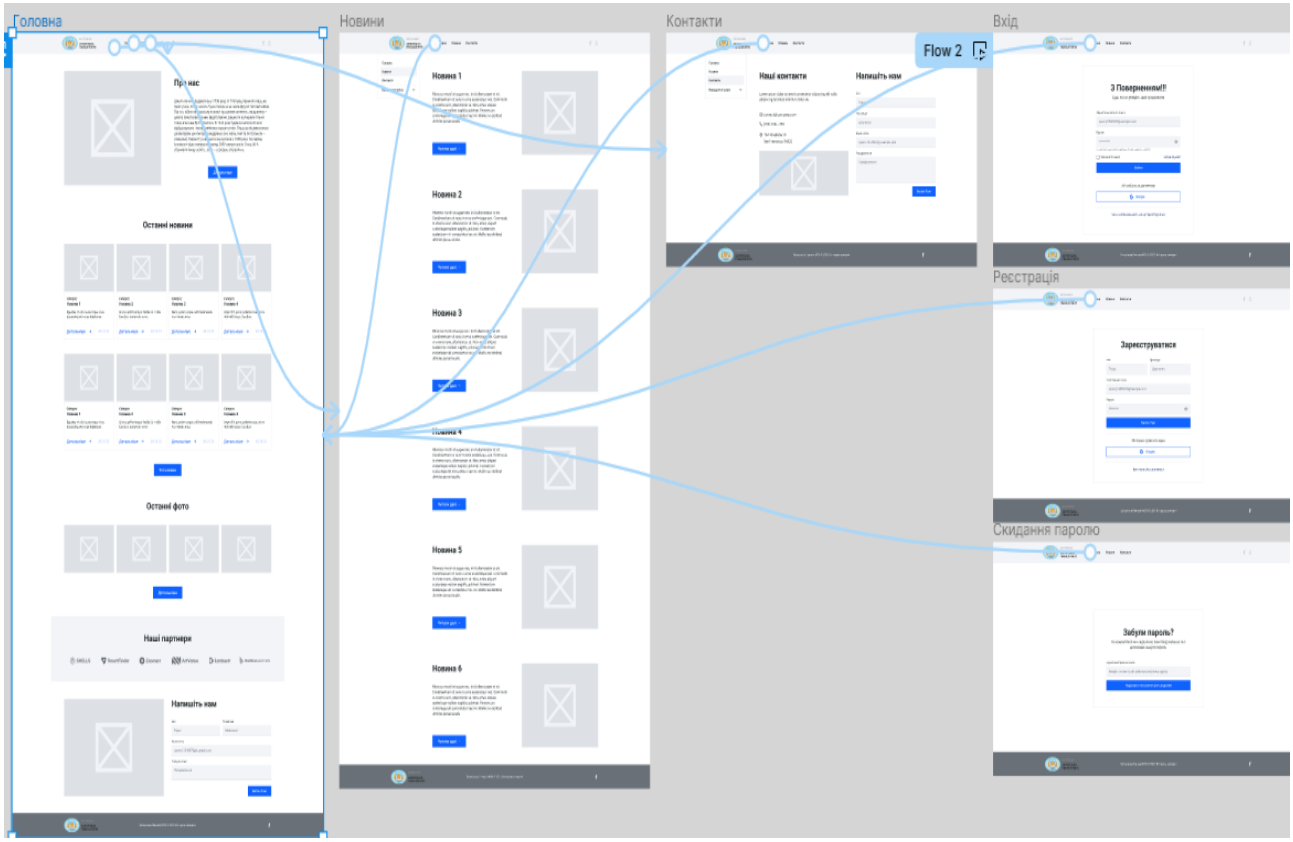


Рисунок 2.11 – Прототипування сайту

2.3.3 Візуальні елементи

Набір візуальних елементів – це набір елементів дизайну, таких як кнопки, форми та інші компоненти інтерфейсу, які дозволяють дизайнерам та розробникам ефективніше взаємодіяти з користувачами.

Мова візуального дизайну в основному складається з 4 категорій, і потрібно врахувати роль, яку кожен із цих елементів дизайну відіграє у кожному компоненті

на екрані:

– колір;
– шрифти;
– розміри та інтервали;
– зображення.

Вибір дизайнерського фреймворка – це важливе рішення, яке може вплинути на успіх вебсайту. Material Design – це хороший вибір для вебсайтів, які повинні бути сучасними, інтуїтивно зрозумілими і адаптивними.

Material Design – це сучасний дизайн, який відповідає останнім тенденціям у вебдизайні. Він використовує прості і зрозумілі форми, а також яскраві кольори. Цей дизайн робить сайт більш привабливими і зручними для використання. Він також і адаптивний дизайн, який добре виглядає на різних пристроях. Він використовує гнучкі макети, які можуть адаптуватися до розміру екрану пристрою. Це дозволяє користувачам отримувати доступ до інформації на вебсайті з будь-якого пристрою. Цей фреймворк – це відкритий фреймворк, який можна безкоштовно використовувати для розробки вебсайтів. Це означає, що ви можете використовувати цей фреймворк для розробки власних вебсайтів.

3 РЕАЛІЗАЦІЯ ВЕБСАЙТУ

Реалізація вебсайту складається з двух частин: це Fontend і Backend.

Frontend – це публічна частина web-додатків (вебсайтів), з якою користувач може взаємодіяти і контактувати напряму. У Frontend входить відображення функціональних завдань призначеного для користувача інтерфейсу, що виконуються на стороні клієнта, а також обробка запитів користувачів. По суті, фронтенд – це все те, що бачить користувач при відкритті web-сторінки.

Backend – це програмно-апаратна частина проєкту. Frontend ж є клієнтською стороною призначеного для користувача інтерфейсу до програмно-апаратної

частини проєкту, тобто до бекенду. Іншими словами бекенд – це все те, що відбувається на стороні сервера і що залишається невидимим користувачеві (сам сервер теж є частиною бекенду, тільки апаратного) [13]. Звідси і назва front – це видиме спереду, back – це те, що приховано позаду, невидиме [14].

Для реалізації Backend було обрано наступні технології:

– Node.js (ES6);
– Express.js + Validator;
– Mongo DB / Mongoose;
– JSON Web Token;
– Multer;
– BCrypt.

Для реалізації Frontend було обрано наступні технології:

– React.js;
– Redux Toolkit;
– React Hook Form;
– React Router;
– React Marcdown / Simpl Editor;
– Axios.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання. Платформа Node.js перетворила JavaScript на мову загального використання з великою спільнотою розробників.

Express – програмний каркас розробки серверної частини вебзастосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення під ліцензією MIT[15]. Він спроєктований для створення вебзастосунків і API.

Validator – бібліотека призначена для валідації даних які приходять на сервер.

MongoDB – система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД,

MongoDB не потрібні таблиці, схеми або окрема мова запитів. Інформація зберігається у вигляді документів або колекцій.

Mongoose – це ORM (Object Relational Mapping – об'єктно-реляційне відображення або зв'язування) для MongoDB. Mongoose надає в розпорядження розробників просте засноване на схемах рішення для моделювання даних застосунку, що містить вбудовану перевірку типів, валідацію, формування запитів і хуки, які відповідають за реалізацію додаткової логіки обробки запитів.

JSON Web Token – потрібен для авторизації та аутентифікації.

Біблеотека Multer знадобиться для завантаження файлів та зображень на сервер.

BCrypt – адаптивна криптографічна хеш-функція формування ключа, яка використовується для захищеного зберігання паролів.

React – це декларативна, ефективна і гнучка JavaScript-бібліотека, призначена для створення інтерфейсів користувача. Вона дозволяє компонувати складні інтерфейси з невеликих окремих частин коду — “компонентів” [16].

Redux Toolkit – це бібліотека, яка спрощує використання Redux. Він надає набори функцій і інструментів, які допомагають розробникам створювати і підтримувати Redux додатки.

React Hook Form в проєкті потрібен для реалізації форм таких як авторизації, реєстрацції та інших.

React Router – це бібліотека для реалізації маршрутизації на стороні клієнта в React-додатках.

React Markdown – це бібліотека, яка дозволяє відображати Markdown-код в React додатках. Він надає простий синтаксис для створення компонентів React, які відображають Markdown-код. Вона потрібна для рендеру статей.

Simple-Editor – це редактор для розробки засобів візуалізації та керування технологічними процесами. Вона треба для рендеру редактору в якому будуть писатися статті.

Axios – це бібліотека JavaScript, яка дозволяє виконувати HTTP-запити з JavaScript-коду. Він є простим у використанні і забезпечує багато функцій, які

роблять його потужним інструментом для розробки вебдодатків. З її допомоги будуть створюватися запити на Backend.

Любий вебсервер вміє приймати http-запити. HTTP-запити – це запити, які використовуються для взаємодії з HTTP-серверами. Вони є основою для вебдодатків і вебсервісів. Він складається з двох частин: заголовка і тіла. Заголовок містить інформацію про тип запиту, метод запиту, адресат запиту, і інші дані. Тіло містить дані, які відправляються на сервер.

Є всього 4 типи запитів:

– GET;
– POST;
– PUT;
– DELETE.

GET використовується для отримання ресурсу з сервера. POST використовується для створення нового ресурсу на сервері. PUT використовується для оновлення існуючого ресурсу на сервері. DELETE використовується для видалення ресурсу з сервера.

При відправкі HTTP-запиту сервер повинен надати код відповіді. Якщо все виконано правильно код відповіді має бути 2XX. Якщо користувача треба перенаправити на другий розділ код відповіді буде 3XX. При невірному запиті сервер відповість кодом 4XX це означає що проблема на стороні клієнта. При несправностях на стороні сервера він відправить код 5XX наприклад кола не може з’єднатися з базою даних.

Розглянемо на прикладі (див. рис. 3.1). Наприклад є користувач, якому потрібно отримати список статтей. В цей момент клієнт надає get запит до серверу, де вказується що потрібно від сервера. Сервер опрацьовує запит збирає список усіх статтей перетворює в формат JSON, додає код статусу і повертає користувачу.

Рисунок 3.1 – Робота HTTP протоколу

Є таке поняття як CRUD – акронім, що позначає чотири базові функції, створення (create), читання (read), модифікація (update), видалення (delete).

Автентифікація – це процес перевірки достовірності пред'явленого користувачем ідентифікатора. Ідентифікатор може бути будь-яким унікальним значенням, яке може бути використано для ідентифікації користувача, наприклад: логін, пароль, біометричні дані. Для реалізації такого функціоналу буде використано JSON Web Token. За допомогою цього протоколу буде генеруватися спеціальний токен для кожного користувача. З таким токеном користувач буде звертатися до сервера, і якщо сервер зможе його розшифрувати і при цьому токен був не просрочений, тоді аунтентифікація вважається успішною, і користувач може виконати якусь захищену дію на сервері. Наприклад видалити статтю.

3.1 Backend

3.1.1 Планування

Визначити вимоги до бекенду важливий етап у проєктуванні будь-якої системи. На цьому етапі необхідно визначити, що система повинна робити, а також які дані вона повинна зберігати, або передавати.

Для нашого проєкту необхідно визначити функціональні (що система повинна робити) та нефункціональні (як система повинна працювати) вимоги.

Функціональні вимоги нашого проєкту:

– користувачам можна реєструватися, авторизуватися, читати новини, переглядати фотогалерею та коментувати новини;

– адміністраторам дозволено створювати, редагувати, видаляти новини, альбоми та коментарі.

Нефункціональні вимоги:

– система доступна 24/7;

– система захищена від несанкціонованого доступу;

– система легка у використанні.

На сайті визначено два типи прав користувачів: простий користувач і адміністратор.

Не в залежності від типу користувача, існують вимоги до даних, які будуть зберігатися в базі даних. Для кожного користувача зберігається: тип, ім'я та прізвище користувача, адреса електронної пошти, пароль, посилання на аватар.

Для кожної новини яка створюється на сайті, зберігаються дані: заголовок, статус (затверджена, відхилена, в процесі редагування), дата створення та опублікування, текст і зображення.

Для кожного альбому зберігається назва, опис і список зображень.

Для кожного коментаря необхідно зберігати такі дані: автор, новина, текст коментаря, дата створення коментаря.

3.1.2 Створеня сервера

Написання коду Backend розпочалася з встановлення Node.js (див. рис. 3.2). Для цього з офіційного сайту[17] завантажено LTS версію, так як вона вважається най стабільнішою. Після встановлення перевірив в терміналі встановлену версію Node.js.

Наступний крок проініціалізував проєкт в поточній папці де буде розроблятися проєкт. Далі в package.json вказано що використовуються модулі – це надає змогу використовувати JS ES6 в проєкті.

В корні проєкту створено index.js (див. рис. 3.3) який буде головним документом в проєкті. В ньому проініціалізував «експрес» додаток і створив «слухача» для перевірки функціональності сервера, який буде прослуховувати запити за каталогом що вказано і передавати 2 параметра: req і res. Він слухає запити до кореневої папки, отримує request, опрацьовує цей запит і відповідає на цей запит за допомогою response. Наступний код запускає вебсервер на порту, визначеному константою PORT. Якщо запуск сервера не вдався, обробник помилок виводить повідомлення про помилку на консоль. Якщо запуск сервера вдався, обробник успіху виводить повідомлення про успіх на консоль.

Рисунок 3.2 – Головна сторінка офіційного сайту Node.js

```
import express from "express";

const PORT = 5000;

const app = express();

app.get("/", (req, res) => {
    res.send("Сервер Працює");
});

app.listen(PORT, (err) => {
    if (err) {
        return console.log(err);
    }
    console.log("Server started on port: " + PORT);
});
```

Рисунок 3.3 – Перевірка найпростішого серверу на Express

Далі було написано тестову авторизацію. Для цього написав тестовий маршрут «/auth/login» (див. рис. 3.4) який приймає POST-запит. В запиті містяться дані про користувача. Для хешування використалась бібліотека JWT яка створює вебтокен для безпечної передачі між клієнтом і сервером. Після хешування даних від клієнта вона відправляється назад до клієнта у вигляді зашифрованого вебтокена(див. рис. 3.5).

```
app.post("/auth/login", (req, res) => {
    console.log(req.body);

    const token = jwt.sign(
        {
            email: req.body.email,
            fullName: req.body.FullNeme,
        },
        "Valhalla4486"
    );

    res.json({
        success: true,
        token,
    });
});
```

Рисунок 3.4 – Код авторизації

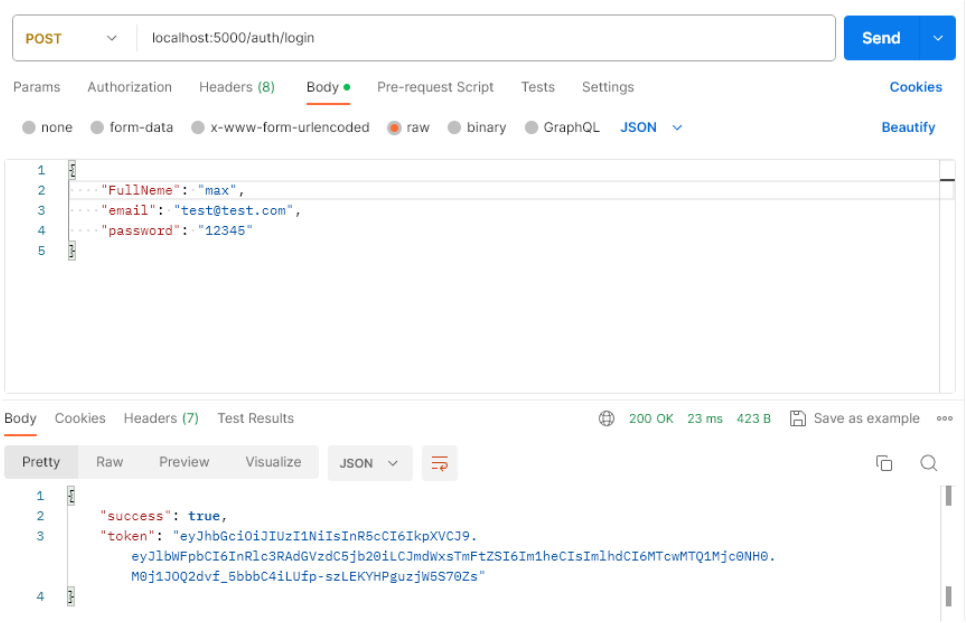


Рисунок 3.5 – Відповідь сервера в вигляді зашифрованого вебтокена

Наступним кроком підключив базу даних MongoDB. Для цього зареєстрував акаунт на офіційному сайті та створено новий кластер для проєкту. В проєкт імпортував бібліотеку mongoose та підключився до бази даних (див. рис. 3.6).

```
mongoose
    .connect("mongodb+srv://grigoryshvets:wqfdsgfgwfefdvDW32@cluster0.xthnm39.mongodb.net/?retryWrites=true&w=majority")
    .then(() => console.log("DataBase OK"))
    .catch((err) => console.log("DataBase Error", err));
```

Рисунок 3.6 – Підключення до БД

3.1.3 Розробка моделей

На цьому етапі було розроблено моделі даних:

– користувачів;

– новин;

– коментарів;

– альбомів.

Модуль користувачів зберігає такі атрибути моделі, які мають певний тип. Тип атрибута визначає, які значення може приймати атрибут (табл. 3.1).

Таблиця 3.1 – Тип і атрибут користувача

Атрибут	**Тип**
Ідентифікатор користувача	ObjectId
Ім'я	string
Прізвище	string
Адреса електронної пошти	string
Пароль	string
Тип	string
Фотографія	string
Дата народження	date

Ця модель використовується для зберігання даних про користувачів системи. Також для ідентифікації та авторизації користувачів, надання користувачам доступу до ресурсів системи.

Модуль новин зберігає такі атрибути моделі, які мають певний тип. Тип атрибута визначає, які значення може приймати атрибут (табл. 3.2).

Таблиця 3.2 – Тип і атрибут новини

Атрибут	**Тип**
Ідентифікатор новини	ObjectId
Заголовок новини	string
Статус новини	string
Дата створення новини	date
Дата публікування новини	date
Текст новини	string
Зображення новини	string

Ця модель використовується для зберігання даних про новини системи. Також для пошуку, сортування та відображення новин.

Альбом - це сукупність зображень, об'єднаних однією темою або метою. Модуль альбомів зберігає такі атрибути моделі, які мають певний тип. Тип атрибута визначає, які значення може приймати атрибут (табл. 3.3).

Таблиця 3.3 – Тип і атрибут альбома

Атрибут	**Тип**
Ідентифікатор альбому	ObjectId
Назва альбому	String
Опис альбому	String
Список зображень в альбомі	Array

Ця модель використовується для зберігання даних про альбоми системи. Також для пошуку, сортування та відображення альбомів.

Коментарі є важливим елементом будь-якого веб-сайту або додатка, який дозволяє користувачам спілкуватися один з одним. Моделі даних для коментарів описують структуру даних, які будуть зберігатися в базі даних.

Таблиця 3.4 – Тип і атрибут коментаря

Атрибут	**Тип**
Ідентифікатор коментаря	ObjectId
Ідентифікатор новини	int
Ідентифікатор автора коментаря	int
Текст коментаря	text
Дата створення коментаря	date

3.1.4 Розробка контролерів

Наступним етапом була розробка контролерів, які відповідають за обробку запитів до сервера. Контролери є частиною архітектури MVC і відповідають за взаємодію між представленим шаром (View) і шаром даних (Model).

Функція авторизації користувача дозволяє перевірити, чи ім'я користувача та пароль є дійсними. Якщо вони є дійсними, функція надає доступ до ресурсів системи.

Загалом наступний код (див. рис. 3.2) реалізує функціональність авторизації користувачів. Він приймає запит POST на адресу /auth і повертає токен доступу, якщо ім'я користувача та пароль є дійсними. Користувач надсилає запит POST на адресу /auth з ім'ям користувача та паролем у тілі запиту. Контролер отримує ім'я користувача та пароль від запиту.

```
// Імпортуємо необхідні модулі
const express = require("express");
const bcrypt = require("bcrypt");
const crypto = require("crypto");

// Створюємо новий контролер
const AuthController = express.Router();

// Обробляємо запит POST /auth
AuthController.post("/auth", (req, res) => {
    // Отримуємо ім'я користувача та пароль від користувача
    const username = req.body.username;
    const password = req.body.password;

    // Перевіряємо, чи ім'я користувача та пароль є дійсними
    const user = UserModel.findOne({ username }, (err, user) => {
        if (err) {
            res.send(err);
        } else if (!user) {
            res.send({ message: "Користувач не знайдений" });
        } else {
            const isValid = bcrypt.compareSync(password, user.password);
            if (!isValid) {
                res.send({ message: "Неправильний пароль" });
            } else {
                // Надаємо доступ до ресурсів системи
                const token = crypto.randomBytes(32).toString("hex");
                user.token = token;
                user.save();
                res.send({
                    message: "Авторизація успішна",
                    token,
                });
            }
        }
    });
});

// Функція для генерації токена
function generateToken(user) {
    // Генеруємо рандомний строк
    const token = crypto.randomBytes(32).toString("hex");

    // Зберігаємо токен у базі даних
    user.token = token;
    user.save();

    // Повертаємо токен
    return token;
}

// Функція для генерації рандомного строку
function generateRandomString(length) {
    // Створюємо масив символів
    const characters =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    // Генеруємо рандомний строк
    let randomString = "";
    for (let i = 0; i < length; i++) {
        randomString += characters[Math.floor(Math.random() * characters.length)];
    }

    // Повертаємо строк
    return randomString;
}

// Повертаємо контролер
module.exports = AuthController;
```

Рисунок 3.2 – Код контролера авторизації

Далі перевіряє, чи ім'я користувача та пароль є дійсними. Для цього контролер використовує модель даних користувачів для пошуку користувача з

ім'ям, яке вказав користувач. Якщо користувач знайдений, контролер використовує функцію bcrypt.compareSync() для порівняння паролів користувача з бази даних та пароля, який вказав користувач. Якщо паролі збігаються, то користувач авторизований.

Якщо користувач авторизований, тоді він генерує токен доступу. Токен доступу – це унікальний код, який використовується для ідентифікації користувача при доступі до захищених ресурсів системи. Для генерації токена доступу контролер використовує функцію crypto.randomBytes() для генерації випадкового рядка з 32 символів. Далі повертається токен доступу користувачеві.

Наступний код (див. рис. 3.3) реалізує функціональність реєстрації користувачів.

```
// Імпортуємо необхідні модулі
import express from "express";
import bcrypt from "bcrypt";
import UserModel from "./models/User";

// Створюємо новий контролер
const AuthController = express.Router();

// Обробляємо запит POST /register
AuthController.post("/register", async (req, res) => {
    // Отримуємо ім'я користувача та пароль від користувача
    const username = req.body.username;
    const password = req.body.password;

    // Перевіряємо, чи ім'я користувача є унікальним
    const existingUser = await UserModel.findOne({ username });
    if (existingUser) {
        res.status(400).send({ message: "Ім'я користувача вже зайняте" });
        return;
    }

    // Хешуємо пароль
    const hashedPassword = await bcrypt.hash(password, 10);

    // Створюємо нового користувача
    const newUser = new UserModel({
        username,
        password: hashedPassword,
    });

    // Зберігаємо нового користувача в базі даних
    await newUser.save();

    // Надаємо доступ до ресурсів системи
    const token = generateToken(newUser);
    res.status(201).send({
        message: "Реєстрація успішна",
        token,
    });
});

// Створюємо функцію для генерації токена
function generateToken(user) {
    const payload = {
        id: user._id,
        username: user.username,
    };
    const secret = process.env.SECRET;
    return jwt.sign(payload, secret, { expiresIn: "1d" });
}

// Експортуємо контролер
export default AuthController;
```

Рисунок 3.3 – Код контролера реєстрації

Він приймає запит POST на адресу /register з ім'ям користувача та паролем у тілі запиту. Якщо ім'я користувача є унікальним, то користувач реєструється та отримує токен доступу. Для хешування пароля контролер використовує функцію

bcrypt.hash(). Використовуючи модель даних користувачів, створюється новий користувач. Контролер зберігає нового користувача в базі даних. Для зберігання нового користувача контролер використовує метод save() моделі даних користувачів. Контролер генерує токен доступу, використовуючи функцію generateToken(). В кінці повертає повідомлення про успіх та токен доступу користувачеві.

Наступний код (див. рис. 3.4) реалізує функціональність відновлення пароля користувачів. Він приймає запит POST на адресу /reset-password з адресою електронної пошти користувача у тілі запиту. Якщо адреса електронної пошти є дійсною, то для користувача генерується новий пароль і відправляється йому електронним листом.

```
// Імпортуємо необхідні модулі
const express = require("express");
const bcrypt = require("bcrypt");
// Створюємо новий контролер
const AuthController = express.Router();
// Обробляємо запит POST /reset-password
AuthController.post("/reset-password", (req, res) => {
    // Отримуємо адресу електронної пошти від користувача
    const email = req.body.email;
    // Перевіряємо, чи адреса електронної пошти є дійсною
    const user = UserModel.findOne({ email }, (err, existingUser) => {
        if (err) {
            res.send(err);
        } else if (!existingUser) {
            res.send({ message: "Адреса електронної пошти не знайдена" });
        } else {
            // Генеруємо новий пароль
            const newPassword = generateRandomString(12);
            // Хешуємо новий пароль
            const hashedNewPassword = bcrypt.hashSync(newPassword, 10);
            // Змінюємо пароль користувача в базі даних
            existingUser.password = hashedNewPassword;
            existingUser.save((err, savedUser) => {
                if (err) {
                    res.send(err);
                } else {
                    // Відправляємо новий пароль користувачеві
                    sendEmail(email, newPassword);
                    res.send({
                        message: "Новий пароль надіслано на вашу адресу електронної пошти",
                    });
                }
            });
        }
    });
});
// Функція для генерації рандомного строку
function generateRandomString(length) {
    // Створюємо масив символів
    const characters =
        "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    // Генеруємо рандомний строк
    let randomString = "";
    for (let i = 0; i < length; i++) {
        randomString += characters[Math.floor(Math.random() * characters.length)];
    }
    // Повертаємо строк
    return randomString;
}
// Функція для відправки електронного листа
function sendEmail(email, newPassword) {
    // Створюємо об'єкт електронного листа
    const mail = {
        from: "no-reply@example.com",
        to: email,
        subject: "Новий пароль для вашого облікового запису",
        text: `Ваш новий пароль: ${newPassword}`,
    };
    // Відправляємо електронний лист
    mail.send((err) => {
        if (err) {
            console.error(err);
        }
    });
}
```

Рисунок 3.4 – Код контролера відновлення паролю

Наступний код (див. рис. 3.5) реалізує функціональність зміни аватара користувача. Він приймає запит POST на адресу /profile/avatar з аватарним файлом у тілі запиту. Якщо аватар успішно завантажений, то аватар користувача в базі даних змінюється на новий аватар.

Функція multer() використовується для обробки запитів з файлами. Ця функція використовується для отримання аватара від запиту.

```
// Імпортуємо необхідні модулі
const express = require("express");
const multer = require("multer");

// Створюємо новий контролер
const ProfileController = express.Router();

// Обробляємо запит POST /profile/avatar
ProfileController.post("/profile/avatar", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо аватар від користувача
    const avatar = req.files.avatar;

    // Якщо аватар не був переданий, повертаємо помилку
    if (!avatar) {
        res.send({ message: "Аватар не був переданий" });
        return;
    }

    // Зберігаємо аватар у базі даних
    const fileName = avatar.filename;
    avatar.mv(`./avatars/${fileName}`, (err) => {
        if (err) {
            res.send(err);
        } else {
            // Змінюємо аватар користувача в базі даних
            user.avatar = fileName;
            user.save((err, savedUser) => {
                if (err) {
                    res.send(err);
                } else {
                    // Відправляємо користувачеві повідомлення про успіх
                    res.send({ message: "Аватар успішно змінено" });
                }
            });
        }
    });
});

// Повертаємо контролер
module.exports = ProfileController;
```

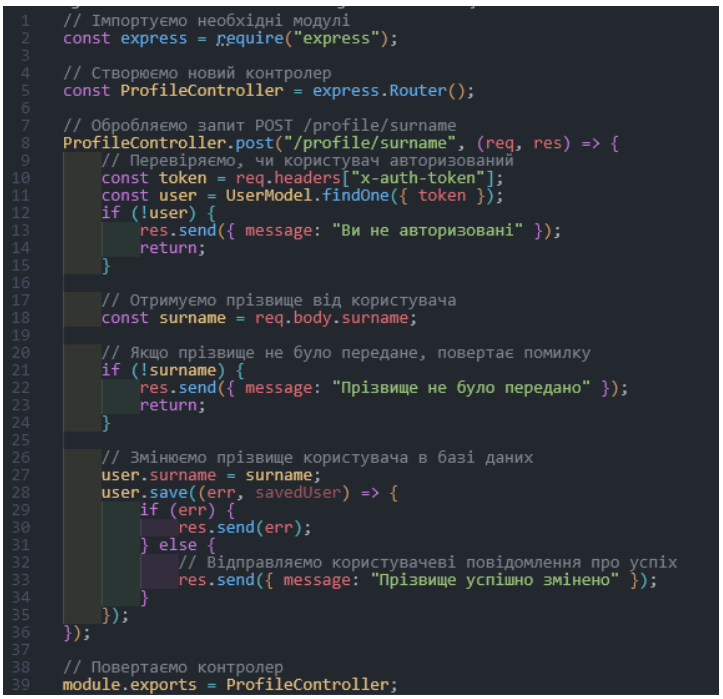
Рисунок 3.5 – Код контролера зміни аватара користувача

Наступний код (див. рис. 3.6) реалізує функціональність зміни імені користувача. Він приймає запит POST на адресу /profile/name з ім'ям користувача у тілі запиту. Якщо ім'я успішно змінено, то ім'я користувача в базі даних змінюється на нове ім'я.

```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const ProfileController = express.Router();

// Обробляємо запит POST /profile/name
ProfileController.post("/profile/name", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо ім'я від користувача
    const name = req.body.name;

    // Якщо ім'я не було передане, повертає помилку
    if (!name) {
        res.send({ message: "Ім'я не було передано" });
        return;
    }

    // Змінюємо ім'я користувача в базі даних
    user.name = name;
    user.save((err, savedUser) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Ім'я успішно змінено" });
        }
    });
});

// Повертаємо контролер
module.exports = ProfileController;
```

Рисунок 3.6 – Код контролера зміни імені користувача

Цей код (див. рис. 3.7) реалізує функціональність зміни прізвища користувача. Він приймає запит POST на адресу /profile/surname з прізвищем користувача у тілі запиту. Якщо прізвище успішно змінено, то прізвище користувача в базі даних змінюється на нове прізвище.



```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const ProfileController = express.Router();

// Обробляємо запит POST /profile/surname
ProfileController.post("/profile/surname", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо прізвище від користувача
    const surname = req.body.surname;

    // Якщо прізвище не було передане, повертає помилку
    if (!surname) {
        res.send({ message: "Прізвище не було передано" });
        return;
    }

    // Змінюємо прізвище користувача в базі даних
    user.surname = surname;
    user.save((err, savedUser) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Прізвище успішно змінено" });
        }
    });
});

// Повертаємо контролер
module.exports = ProfileController;
```

Рисунок 3.7 – Код контролера зміни призвища користувача

Процес зміни пароля проходить у кілька кроків (див. рис. 3.8), користувач надсилає запит POST на адресу /profile/password зі старим паролем і новим паролем у тілі запиту. Контролер отримує старий пароль і новий пароль від запиту. Та якщо старий пароль не було передано, контролер повертає помилку. А якщо новий пароль не було передано, контролер повертає помилку. Також він перевіряє, чи старий пароль є правильним. Далі хешує новий пароль і змінюється в базі даних. У відповідь контролер відправляє користувачеві повідомлення про успіх.

```
// Імпортуємо необхідні модулі
const express = require("express");
const bcrypt = require("bcrypt");

// Створюємо новий контролер
const ProfileController = express.Router();

// Обробляємо запит POST /profile/password
ProfileController.post("/profile/password", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо старий пароль від користувача
    const oldPassword = req.body.oldPassword;

    // Отримуємо новий пароль від користувача
    const newPassword = req.body.newPassword;

    // Якщо старий пароль не був переданий, повертає помилку
    if (!oldPassword) {
        res.send({ message: "Старий пароль не було передано" });
        return;
    }

    // Якщо новий пароль не був переданий, повертає помилку
    if (!newPassword) {
        res.send({ message: "Новий пароль не було передано" });
        return;
    }

    // Перевіряємо, чи старий пароль є правильним
    const isValid = bcrypt.compareSync(oldPassword, user.password);
    if (!isValid) {
        res.send({ message: "Неправильний старий пароль" });
        return;
    }

    // Хешуємо новий пароль
    const hashedNewPassword = bcrypt.hashSync(newPassword, 10);

    // Змінюємо пароль користувача в базі даних
    user.password = hashedNewPassword;
    user.save((err, savedUser) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({
                message: "Пароль успішно змінено",
                token: generateToken(savedUser),
            });
        }
    });
});

// Повертаємо контролер
module.exports = ProfileController;
```

Рисунок 3.8 – Код контролера зміни пароля користувача

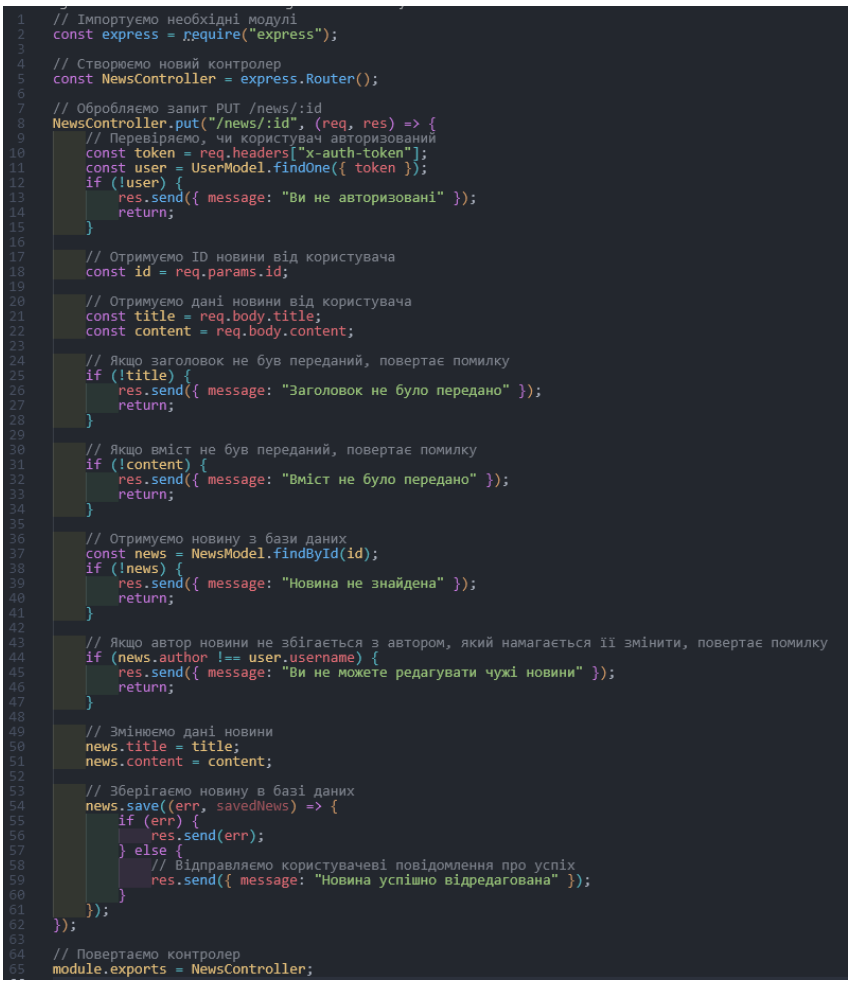
Наступний контролер – це додавання новин (див. рис. 3.9). Він приймає запит POST на адресу /news з заголовком і змістом новини у тілі запиту. Якщо новина успішно додана, то вона додається до бази даних.

```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const NewsController = express.Router();

// Обробляємо запит POST /news
NewsController.post("/news", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо дані новини від користувача
    const title = req.body.title;
    const content = req.body.content;

    // Якщо заголовок не був переданий, повертає помилку
    if (!title) {
        res.send({ message: "Заголовок не було передано" });
        return;
    }

    // Якщо вміст не був переданий, повертає помилку
    if (!content) {
        res.send({ message: "Вміст не було передано" });
        return;
    }

    // Створюємо нову новину
    const news = new NewsModel({
        title,
        content,
        author: user.username,
    });

    // Зберігаємо новину в базі даних
    news.save((err, savedNews) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Новина успішно додана" });
        }
    });
});

// Повертаємо контролер
module.exports = NewsController;
```

Рисунок 3.9 – Код контролера додавання новини

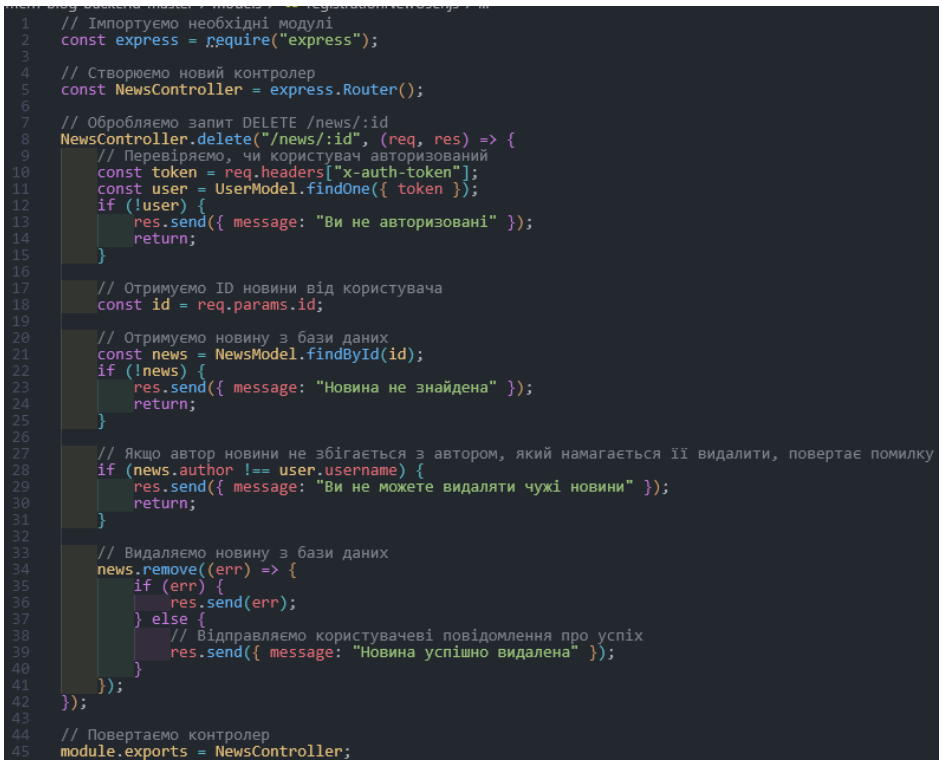
Процес редагування новини (див. рис. 3.10) проходить у кілька кроків. користувач надсилає запит PUT на адресу /news/:id з новими даними новини у тілі запиту. Контролер отримує ID і нові дані новини від запиту. Якщо заголовок або вміст не було передано, контролер повертає помилку. Далі він отримує новину з бази даних. Якщо новина не знайдена, або автор новини не збігається з автором, який намагається її змінити, контролер повертає помилку. Потім змінює та зберігає в базі даних дані новини. У відповідь користувачу відправляє повідомлення про успіх.

Цей код (див. рис. 3.11) реалізує функціональність видалення новин. Він приймає запит DELETE на адресу /news/:id. Якщо новина успішно видалена, то вона видаляється з бази даних.



```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const NewsController = express.Router();

// Обробляємо запит PUT /news/:id
NewsController.put("/news/:id", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо ID новини від користувача
    const id = req.params.id;

    // Отримуємо дані новини від користувача
    const title = req.body.title;
    const content = req.body.content;

    // Якщо заголовок не був переданий, повертає помилку
    if (!title) {
        res.send({ message: "Заголовок не було передано" });
        return;
    }

    // Якщо вміст не був переданий, повертає помилку
    if (!content) {
        res.send({ message: "Вміст не було передано" });
        return;
    }

    // Отримуємо новину з бази даних
    const news = NewsModel.findById(id);
    if (!news) {
        res.send({ message: "Новина не знайдена" });
        return;
    }

    // Якщо автор новини не збігається з автором, який намагається її змінити, повертає помилку
    if (news.author !== user.username) {
        res.send({ message: "Ви не можете редагувати чужі новини" });
        return;
    }

    // Змінюємо дані новини
    news.title = title;
    news.content = content;

    // Зберігаємо новину в базі даних
    news.save((err, savedNews) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Новина успішно відредагована" });
        }
    });
});

// Повертаємо контролер
module.exports = NewsController;
```

Рисунок 3.10 – Код контролера редагування новини



```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const NewsController = express.Router();

// Обробляємо запит DELETE /news/:id
NewsController.delete("/news/:id", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо ID новини від користувача
    const id = req.params.id;

    // Отримуємо новину з бази даних
    const news = NewsModel.findById(id);
    if (!news) {
        res.send({ message: "Новина не знайдена" });
        return;
    }

    // Якщо автор новини не збігається з автором, який намагається її видалити, повертає помилку
    if (news.author !== user.username) {
        res.send({ message: "Ви не можете видаляти чужі новини" });
        return;
    }

    // Видаляємо новину з бази даних
    news.remove((err) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Новина успішно видалена" });
        }
    });
});

// Повертаємо контролер
module.exports = NewsController;
```

Рисунок 3.11 – Код контролера видаленя новини

Наступний функціонал (див. рис. 3.12) додавання альбомів. Він приймає запит POST на адресу /albums з заголовком і описом альбому у тілі запиту. Якщо альбом успішно доданий, то він додається до бази даних.

```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const AlbumController = express.Router();

// Обробляємо запит POST /albums
AlbumController.post("/albums", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо дані альбому від користувача
    const title = req.body.title;
    const description = req.body.description;

    // Якщо заголовок не було передано, повертає помилку
    if (!title) {
        res.send({ message: "Заголовок не було передано" });
        return;
    }

    // Якщо опис не було передано, повертає помилку
    if (!description) {
        res.send({ message: "Опис не було передано" });
        return;
    }

    // Створюємо новий альбом
    const album = new AlbumModel({
        title,
        description,
        author: user.username,
    });

    // Зберігаємо альбом в базі даних
    album.save((err, savedAlbum) => {
        if (err) {
            res.send(err);
        } else {
            // Відправляємо користувачеві повідомлення про успіх
            res.send({ message: "Альбом успішно додано" });
        }
    });
});

// Повертаємо контролер
module.exports = AlbumController;
```

Рисунок 3.12 – Код контролера додавання альбому

Наступний контролер (див. рис. 3.13) реалізує функціональність редагування альбомів. Він приймає запит PUT на адресу /albums/:id з новими даними альбому у тілі запиту. Якщо альбом успішно відредагований, то нові дані альбому зберігаються в базі даних.

Наступний це функціонал видалення альбомів (див. рис. 3.14). Він приймає запит DELETE на адресу /albums/:id. Якщо альбом успішно видалений, то він видаляється з бази даних.

```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const AlbumController = express.Router();

// Обробляємо запит PUT /albums/:id
AlbumController.put("/albums/:id", (req, res) => {
    // Перевіряємо, чи користувач авторизований
    const token = req.headers["x-auth-token"];
    const user = UserModel.findOne({ token });
    if (!user) {
        res.send({ message: "Ви не авторизовані" });
        return;
    }

    // Отримуємо ID альбому від користувача
    const id = req.params.id;

    // Отримуємо альбом з бази даних
    const album = AlbumModel.findById(id);
    if (!album) {
        throw new Error("Альбом не знайдено");
    }

    // Якщо автор альбому не збігається з автором, який намагається його змінити, повертає помилку
    if (album.author !== user.username) {
        throw new Error("Ви не можете редагувати чужі альбоми");
    }

    // Змінюємо дані альбому
    album.title = req.body.title;
    album.description = req.body.description;

    // Зберігаємо альбом в базі даних
    album.save();

    // Відправляємо користувачеві повідомлення про успіх
    res.send({ message: "Альбом успішно відредаговано" });
});

// Повертаємо контролер
module.exports = AlbumController;
```

Рисунок 3.13 – Код контролера редагування альбому

```
// Імпортуємо необхідні модулі
const express = require("express");

// Створюємо новий контролер
const AlbumController = express.Router();

// Обробляємо запит DELETE /albums/:id
AlbumController.delete("/albums/:id", (req, res) => {
  // Перевіряємо, чи користувач авторизований
  const token = req.headers["x-auth-token"];
  const user = UserModel.findOne({ token });
  if (!user) {
    res.send({ message: "Ви не авторизовані" });
    return;
  }

  // Отримуємо ID альбому від користувача
  const id = req.params.id;

  // Отримуємо альбом з бази даних
  const album = AlbumModel.findById(id);
  if (!album) {
    res.send({ message: "Альбом не знайдено" });
    return;
  }

  // Якщо автор альбому не збігається з автором, який намагається його видалити, повертає помилку
  if (album.author !== user.username) {
    res.send({ message: "Ви не можете видаляти чужі альбоми" });
    return;
  }

  // Видаляємо альбом з бази даних
  album.remove();

  // Відправляємо користувачеві повідомлення про успіх
  res.send({ message: "Альбом успішно видалено" });
});

// Повертаємо контролер
```

Рисунок 3.14 – Код контролера видаленя альбому

Наступний контролер (див. рис. 3.15) спочатку перевіряє, чи користувач авторизований і має доступ до цієї дії. Якщо немає, то користувачеві відправляється помилка. Якщо користувач авторизований, то контролер отримує ID користувача,

який потрібно видалити. Потім контролер отримує користувача з бази даних за цим ID. Якщо користувача не знайдено, то користувачеві відправляється помилка. Але якщо користувач знайдено, то контролер видаляє його з бази даних. Якщо видалення не вдалося, то користувачеві відправляється помилка, або відправляє користувачеві повідомлення про успіх якщо вдалося видаити.

```
// Імпортуємо необхідні модулі
const express = require("express");
const UserModel = require("./models/user");

// Створюємо новий контролер
const UsersController = express.Router();

// Обробляємо запит DELETE /users/:id
UsersController.delete("/users/:id", (req, res) => {
  // Перевіряємо, чи користувач авторизований
  const token = req.headers["x-auth-token"];
  const user = UserModel.findOne({ token });
  if (!user || user.role !== "admin") {
    res.send({ message: "Ви не маєте доступу до цієї дії" });
    return;
  }

  // Отримуємо ID користувача від адміністратора
  const id = req.params.id;

  // Отримуємо користувача з бази даних
  const userToDelete = UserModel.findById(id);
  if (!userToDelete) {
    res.send({ message: "Користувач не знайдено" });
    return;
  }

  // Видаляємо користувача з бази даних
  userToDelete.remove((err, deletedUser) => {
    if (err) {
      res.send(err);
      return;
    }

    // Відправляємо адміністратору повідомлення про успіх
    res.send({ message: "Користувач успішно видалено" });
  });
});

// Повертаємо контролер
module.exports = UsersController;
```

Рисунок 3.15 – Код контролера видаленя альбому

3.2 Frontend

Для початку, треба ініціалізувати проєкт в корневій папці проєкту. Після ініціалізації в папці проєкту з’явиться файл package.json? в якому в ходы проэкту будуть зявлятися усі залежності проєкту. Далі було виконано команду: npm install react react-dom redux react-hook-form react-router react-markdown @simpl/editor axios. Це встановить усі необхідні для проєкта бібліотеки.

На зображенні представлена структура проєкту React, яка відповідає описаним принципам. Папка src організована за функціональністю компонентів. Папка public містить файли HTML, CSS та JavaScript, які необхідні для відображення сторінок веб-сайту. Папка node_modules містить бібліотеки та модулі, які використовуються проєктом. В папкі src знаходиться папка components, pages та файли App.js, index.js, index.scss, theme.js.

Файл App.js є основним файлом веб-сайту, розробленого за технологією MERN (MongoDB, Express, React, Node.js). Цей файл відповідає за запуск веб-сайту і ініціалізацію всіх компонентів і сторінок.

Файл index.js є основним файлом сторінки в веб-сайті, розробленому за технологією React. Цей файл відповідає за ініціалізацію компонентів і сторінки.

Файл index.scss є основним файлом стилів для веб-сайту, розробленого за технологією React. Цей файл відповідає за визначення всіх стилів, які використовуються на веб-сайті.

Файл theme.js є файлом стилів, який використовується для визначення тем для веб-сайту, розробленого за технологією React. Цей файл містить код для визначення стилів, які будуть використовуватися на веб-сайті в залежності від обраної теми.

Папка pages в файловій структурі веб-сайту, розробленого за технологією React, містить файли для сторінок веб-сайту. Кожна сторінка веб-сайту має свій власний каталог в папці pages.

Папка components в файловій структурі веб-сайту, розробленого за технологією React, містить файли для компонентів веб-сайту. Компоненти є основними будівельними блоками React. Вони використовуються для відображення окремих елементів інтерфейсу користувача.

3.2.1 Верстання

Gulp – це таск-менеджер, написаний на мові програмування JavaScript, для автоматизації завдань розробки веб-сайтів і додатків. Його може бути використаний для будь-якого типу веб-сайту або додатка. Однак він особливо

корисний для веб-сайтів і додатків, які використовують складні препроцесори, такі як SASS і TypeScript, або які потребують оптимізації для швидкості завантаження.

Він дозволяє автоматизувати завдання розробки веб-сайтів і додатків, що звільняє час для інших завдань. Підвищіти ефективність розробки веб-сайтів і додатків, зменшуючи кількість часу, який витрачається на виконання повторюваних завдань. А також повторно використовувати код для виконання завдань, що полегшує розробку і масштабування веб-сайтів і додатків.

В верстанні було використано наступні модулі: browser-sync, del, gulp-sass, gulp, gulp-autoprefixer, gulp-concat, gulp-imagemin, gulp-uglify-es, sass.

В самому файлі gulp.js (див. рис. 3.16), імпортуються усі необхідні функції, які використовується при верстанні сайту.

Рисунок 3.16 – Імпорт модулів для верстання

Функція браузерної синхронізації (див. рис. 3.17) відповідає за ініціалізацію екземпляра BrowserSync і налаштування його на обслуговування файлів з каталогу програми. BrowserSync – це інструмент, який допомагає розробляти веб-додатки, дозволяючи бачити зміни в коді у реальному часі без необхідності вручну перезавантажувати браузер.

Рисунок 3.17 – Функція синхронізації браузера

Функція scripts (див. рис. 3.18) відповідає за компіляцію, мінімізацію та конкатенацію файлів JavaScript, розташованих у каталозі app/js. Вона також запускає перезавантаження BrowserSync, щоб відобразити будь-які зміни в JavaScript-коді.

```
function scripts() {
  return src(["app/js/**/*.js", "app/js/main.js"])
    .pipe(concat("main.min.js"))
    .pipe(uglify())
    .pipe(dest("app/js"))
    .pipe(browserSync.stream(""));
}
```

Рисунок 3.18 – Функція компіляції JS

Функція img (див. рис. 3.19) відповідає за оптимізацію всіх файлів зображень, розташованих у каталозі app/img. Вона використовує бібліотеку imagemin для виконання різних методів оптимізації зображень і зберігає оптимізовані зображення до каталогу dist/img.

```
function img() {
  return src("app/img/*.*")
    .pipe(
      imagemin([
        imagemin.gifsicle({ interlaced: true }),
        imagemin.mozjpeg({ quality: 75, progressive: true }),
        imagemin.optipng({ optimizationLevel: 5 }),
        imagemin.svgo({
          plugins: [{ removeViewBox: true }, { cleanupIDs: false }],
        }),
      ])
    )
    .pipe(dest("dist/img"));
}
```

Рисунок 3.19 – Функція img

Функція build (див. рис. 3.20) відповідає за створення розповсюджуваної збірки веб-додатку. Вона копіює всі необхідні файли з каталогу app до каталогу dist, забезпечуючи можливість розгортання програми на веб-сервері.

Кінцевий код gulp.js експортує визначені функції як експортовані об'єкти і визначає дві задачі, "build" і "default".

Завдяки цій збірці gulp.js можна зверстати усі необхідні компоненти сайту.

```
function build() {
  return src(
    [
      "app/css/style.min.css",
      "app/fonts/**/*",
      "app/js/main.min.js",
      "app/*.html",
    ],
    { base: "app" }
  ).pipe(dest("dist"));
}

function cleanDist() {
  return del("dist");
}
```

Рисунок 3.20 – Функція build

```
exports.styles = styles;
exports.watching = watching;
exports.browsersync = browsersync;
exports.scripts = scripts;
exports.img = img;

exports.build = series(cleanDist, img, build);
exports.default = parallel(scripts, browsersync, watching);
```

Рисунок 3.21 – Експорт функцій та задач

В якості CSS-фреймворку використовувався material design. Material Design – це дизайн-система, розроблена компанією Google. Вона пропонує набори елементів інтерфейсу, шрифтів, кольорів і стилів, які можна використовувати для створення інтерфейсів, які є одночасно красивими, інтуїтивно зрозумілими і доступними.

3.2.2 Розробка компонентів

Щоб створити компонент, потрібно створити новий файл JavaScript з розширенням .jsx. У файлі слід визначити клас, який розширює клас React.Component і визначити метод render(), який повертає HTML-код, який потрібно відобразити.

Компоненти які були розроблені в для проєкту:

– головний компонент;
– блок про нас;
– галерея;
– форма зворотнього зв'язку;
– список новин;

– новина;
– форма коментаря;
– форма редагування профілю;
– інформація про контакти.

Головний компонент (App) – це основний компонентом, який буде містить всі інші компоненти. У блоку «про нас» відображається інформація про гімназію. У галареї відображаються фотографії або відео, які стосуються тематики гімназії. На формі зворотньогозвязку надається можливість користувачам залишати коментарі або запитання.

В спискі новин відображається список новин. Кожен елемент списку містить заголовок, дату публікації та короткий опис новини.

Новина – це компонент в якому відображається уся новина цілком.

Список новин – відображатє список новин, які адміністратор може відредагувати або видалити.

Форма коментаря в цьому компоненті надається можливість користувачам писати коментарі до новин.

Форма редагування профілю дає можливість користувачам редагувати свій профіль.

Інформація про контакти – відображає інформацію про те, як зв'язатися з адміністрацією гімназії.

Компонент, який буде відображати навігаційне меню: Цей компонент може бути використаний для всіх сторінок.

У проєкті компоненти зберігаються в папці components. Компоненти для головної сторінки, сторінки з новинами, особистого кабінету та сторінки з контактами зберігаються в папкі peges.

3.2.3 Управління станом

Управління станом – це процес зберігання і керування даними, які змінюються протягом часу. У React управління станом можна здійснювати за допомогою хуків useState або useReducer.

Хук useState дозволяє зберігати і керувати одним виразом стану. Так є

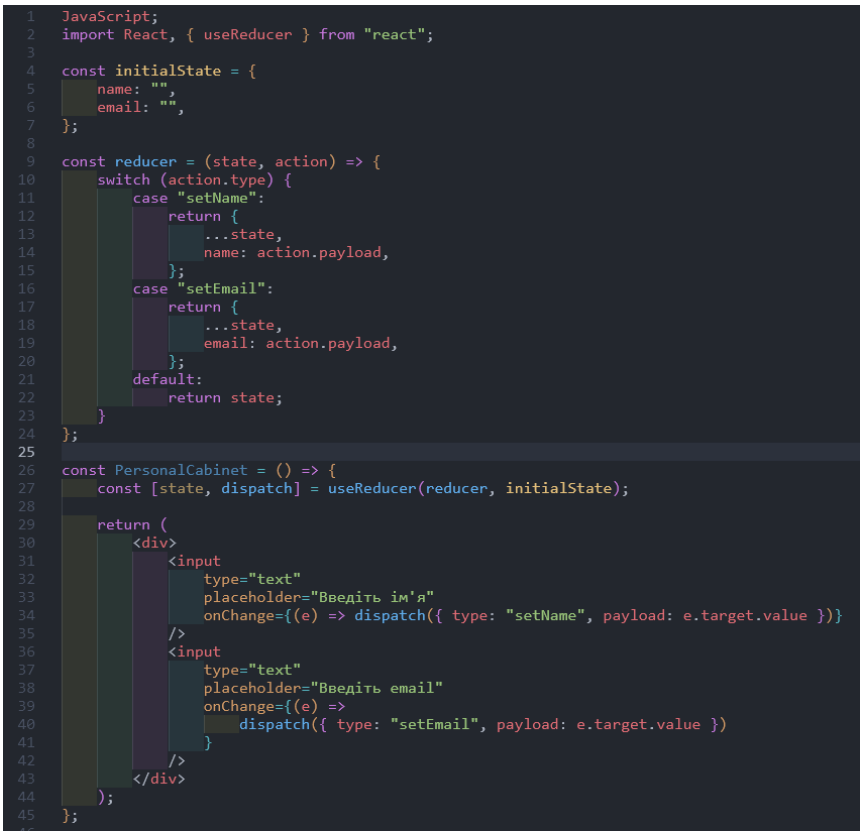
можливість використовувати хук useState для зберігання списку новин у компоненті NewsList.js (див. рис. 3.22).

```
import React, { useState } from "react";

const NewsList = () => {
    const [news, setNews] = useState([]);

    return (
        <ul>
            {news.map((newsItem) => (
                <li key={newsItem.id}>{newsItem.title}</li>
            ))}
        </ul>
    );
};
```

Рисунок 3.22 – Використання хука useState

У цьому коді змінна news зберігає список новин. Змінна setNews використовується для оновлення списку новин.

Хук useReducer дозволяє зберігати і керувати складнішим станом. Таким чином можливо використовувати хук useReducer для зберігання стану користувача в компоненті PersonalCabinet.js (див. рис. 3.23).



```
JavaScript;
import React, { useReducer } from "react";

const initialState = {
    name: "",
    email: "",
};

const reducer = (state, action) => {
    switch (action.type) {
        case "setName":
            return {
                ...state,
                name: action.payload,
            };
        case "setEmail":
            return {
                ...state,
                email: action.payload,
            };
        default:
            return state;
    }
};

const PersonalCabinet = () => {
    const [state, dispatch] = useReducer(reducer, initialState);

    return (
        <div>
            <input
                type="text"
                placeholder="Введіть ім'я"
                onChange={(e) => dispatch({ type: "setName", payload: e.target.value })}
            />
            <input
                type="text"
                placeholder="Введіть email"
                onChange={(e) =>
                    dispatch({ type: "setEmail", payload: e.target.value })
                }
            />
        </div>
    );
};
```

Рисунок 3.23 – Використання хука useReducer

У цьому коді змінна state зберігає стан користувача. Змінна dispatch використовується для оновлення стану. Для управління станом у React також можна використовувати сторонні бібліотеки, такі як Redux або MobX. Ці бібліотеки пропонують більш гнучкі і масштабовані можливості управління станом.

3.2.4 Інтеграція з бекендом

Інтеграція фронтенду з бекендом – це процес обміну даними між фронтендом і бекендом. У проєкті інтеграція з бекендом здійснюється за допомогою AJAX-запитів.

AJAX-запити – це HTTP-запити, які можуть виконуватися асинхронно. Це означає, що фронтенд може продовжувати працювати, поки виконується AJAX-запит. Для здійснення запиту використовується бібліотека axios (див. рис. 3.24). Бібліотека пропонує простий і зручний інтерфейс для здійснення AJAX-запитів.

Таким чином завдяки бібліотеці axios для отримання списку новин з бекенда, використовується наступний код.

```
import axios from "axios";

const NewsList = () => {
    const [news, setNews] = useState([]);

    useEffect(() => {
        axios
            .get("/api/news")
            .then((response) => {
                setNews(response.data);
            })
            .catch((error) => {
                console.error(error);
            });
    }, []);

    return (
        <ul>
            {news.map((newsItem) => (
                <li key={newsItem.id}>{newsItem.title}</li>
            ))}
        </ul>
    );
};
```

Рисунок 3.24 – Використання бібліотеки axios

У цьому коді компонент NewsList використовує бібліотеку axios для отримання списку новин з бекенда. Запит виконується асинхронно, і результат запиту зберігається у змінній news.

3.3 Створення власного чат-бота з використанням OpenAI API

ChatGPT і OpenAI API – це два різні способи доступу до моделей штучного інтелекту (ШІ) від OpenAI. ChatGPT – це веб-інтерфейс, який дозволяє користувачам спілкуватися з моделями ШІ за допомогою звичайного текстового чату. OpenAI API – це API, який дозволяє розробникам інтегрувати моделі ШІ в свої власні програми.

Основні відмінності між ChatGPT і OpenAI API можна узагальнити в наступній таблиці (табл. 3.5):

Таблиця 3.5 – Порівняльна таблиця між ChatGPT і OpenAI API

Характеристика	**ChatGPT**	**OpenAI API**
Тип інтерфейсу	Чат	API
Вбудовані функції	Інтегроване переглядання, виконання коду, плагіни	Створення тексту, переклад мов, написання творчого контенту, відповіді на запитання
Оптимізація	Для роботи в чаті	Для різних завдань
Ціна	Від безкоштовно	Від $0,025 за 1000 символів

ChatGPT краще підходить для розробників, які хочуть використовувати технології штучного інтелекту для створення чат-ботів або інших додатків, які взаємодіють з користувачами в реальному часі та яким потрібні вбудовані функції, такі як інтегроване переглядання або виконання коду.

OpenAI API краще підходить для розробників, які хочуть використовувати технології штучного інтелекту для різних завдань, таких як створення тексту, переклад мов, написання творчого контенту або відповіді на запитання та яким потрібні більш гнучкі можливості налаштування.

3.3.1 Ключ API у OpenAI

Перейдіть на веб-сайт OpenAI(див. рис. 3.25) і натисніть кнопку "Log in". та натисніть " Continue with Google", або інший обліковий запис. Введіть свої дані.

Перейдіть до розділу "Мої продукти" на веб-сайті OpenAI і натисніть

"Створити ключ API". Виберіть тип ключа "ChatGPT" і натисніть "Створити".

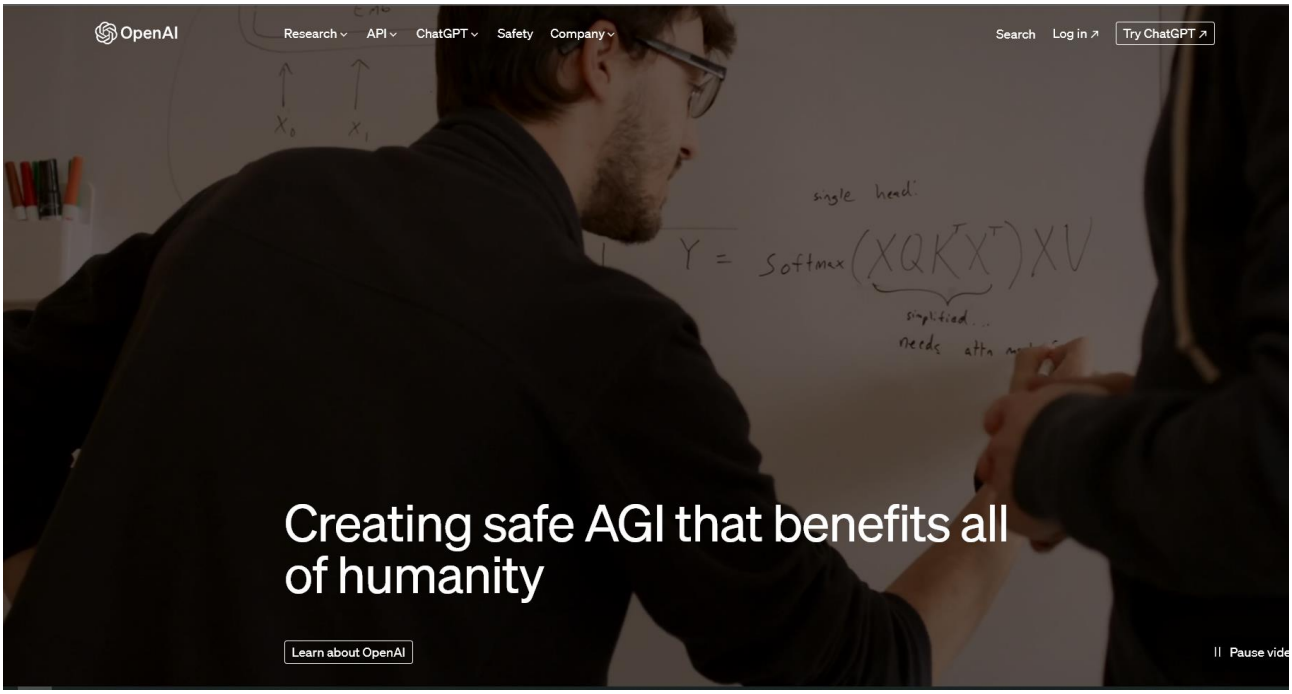


Рисунок 3.25 – Головна сторінка OpenAI

3.3.2 JavaScript-бібліотека ChatGPT

Далі треба встановити бібліотеку ChatGPT «npm і chatgpt» також можливо завантажити бібліотеку з веб-сайту OpenAI.

3.3.3 React-компонент для чат-бота.

Створивши файл з розширенням .jsx додав код чату.

Спочатку ми імпортуємо необхідні модулі. React і Component – це базові модулі React, які нам потрібні для створення компонента. ChatGPT – це модуль, який дозволяє нам взаємодіяти з ChatGPT через API.

Конструктор компонента встановлює початковий стан компонента з властивостями question та answer для зберігання питання та відповіді відповідно. Функція handleInputChange оновлює стан question при зміні значення текстового поля.

Функція handleSubmit обробляє відправку форми. Вона створює екземпляр ChatGPT, вказуючи API-ключ, базу даних та колекцію для генерування відповідей. Потім вона викликає метод generateAnswer з введеним питанням та зберігає

отриману відповідь у стані answer (див. рис. 3.26).

gpt.jsx > ...

```jsx
import React, { Component } from "react";
import ChatGPT from "chat-gpt";

class ChatBot extends Component {
    constructor(props) {
        super(props);

        this.state = {
            question: "",
            answer: "",
        };
    }

    handleInputChange = (event) => {
        this.setState({
            question: event.target.value,
        });
    };

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

        const { question } = this.state;

        const chatGPT = new ChatGPT({
            apiKey: "{
                        "id": "abcd1234-5678-90ab-cdef-012345678901",
                        "secret": "secret-key-1234567890123456789012345678901234567890"
            }",
            database: "school",
            collection: "questions",
        });

        const answer = chatGPT.generateAnswer(question);

        this.setState({
            answer,
        });
    };

    render() {
        const { question, answer } = this.state;

        return (
            <div>
                <input
                    type="text"
                    placeholder="Задайте питання"
                    value={question}
                    onChange={this.handleInputChange}
                />
                <button onClick={this.handleSubmit}>Відправити</button>
                <p>{answer}</p>
            </div>
        );
    }
}

export default ChatBot;
```

Рисунок 3.26 – Код чат-боту

Функція render повертає JSX-код для відображення інтерфейсу чат-бота. Він містить текстове поле для введення питання, кнопку для відправлення та абзац для відображення отриманої відповіді

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Опис обраних критеріїв

Мета дослідження полягала в оцінці якості розробленого вебсайту гімназії №55 (див. рис. 4.1) із інтелектуальним чат-ботом (див. рис. 4.2)., який призначений для широкого кола користувачів, включаючи учнів, батьків, вчителів та громадськість. Для досягнення цієї мети було обрано критерії, які дозволяють оцінити, чи відповідає вебсайт потребам відвідувачів і чи виконує свої функції, щоб вони були актуальні для всіх категорій користувачів.

Для дослідження якості вебсайту були використані два методи: опитування та аналіз даних про відвідування та використання сайту. Ці методи дозволяють отримати інформацію про думку відвідувачів, про його якість, а також, про його ефективність.

Тож інформація на вебсайті повинна бути легкодоступною та зрозумілою для всіх відвідувачів, незалежно від їхнього рівня знань та навичок. Цей критерій є важливим, оскільки він забезпечує можливість усім користувачам отримати необхідну інформацію про гімназію.

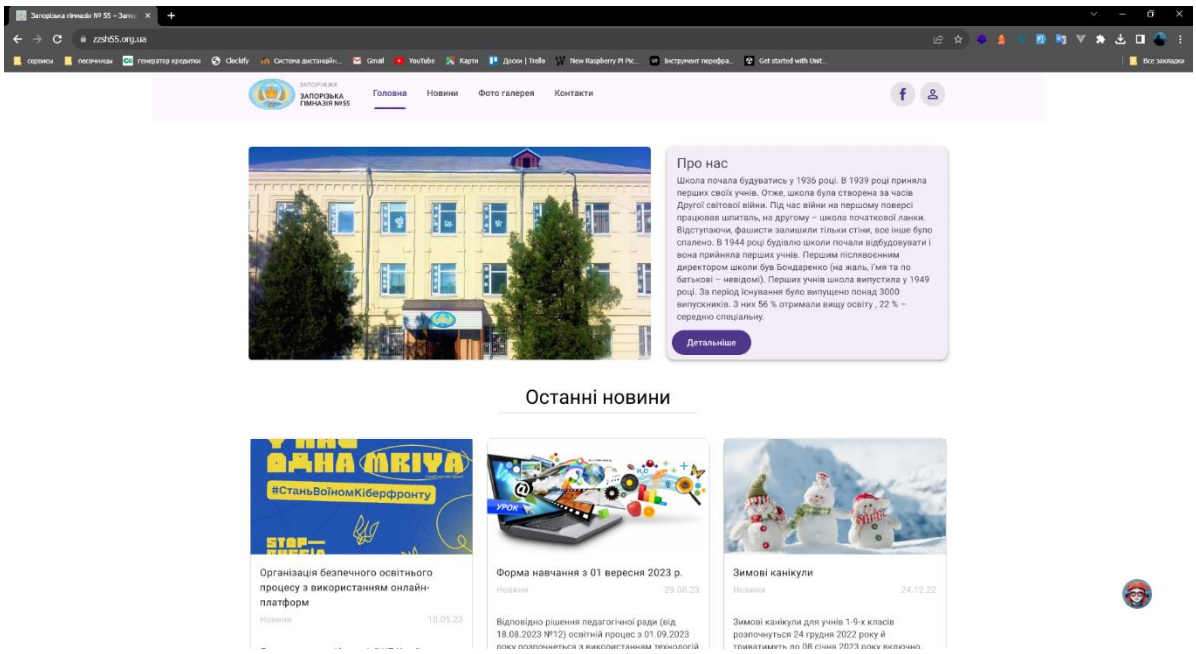


Рисунок 4.1 – Головна сторінка

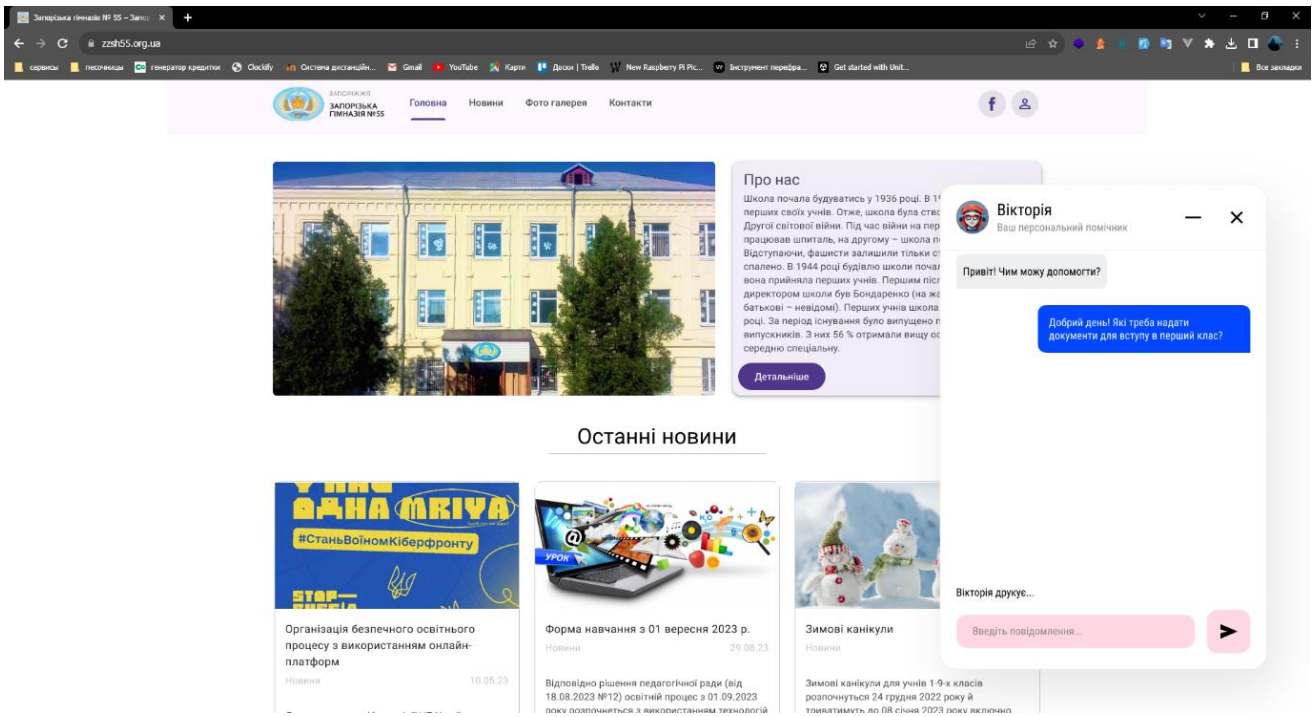


Рисунок 4.2 – Головна сторінка з відкритим чат-ботом

Вебсайт повинен містити актуальну та повну інформацію про гімназію, новини та події (див. рис. 4.3). Цей критерій є важливим, оскільки він забезпечує користувачам доступ до необхідної інформації.

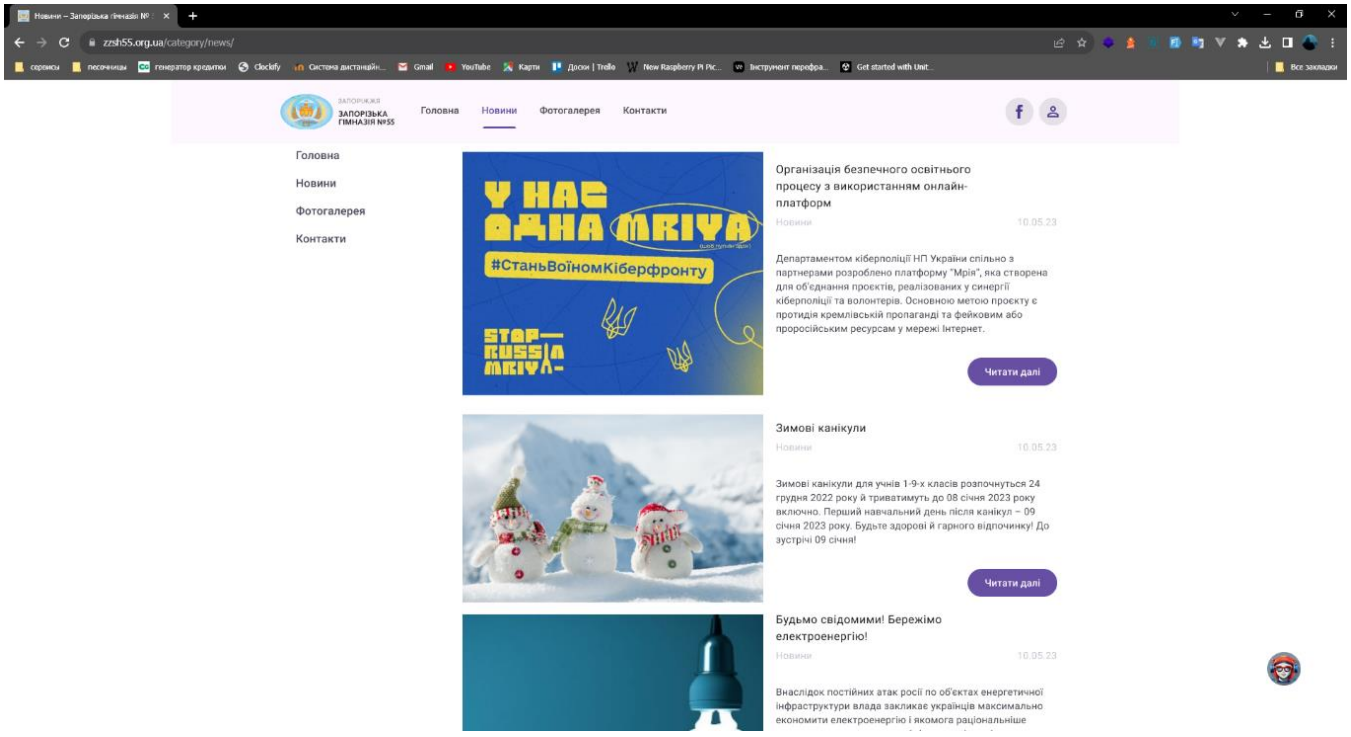


Рисунок 4.3 – Стрічка новин

Вебсайт повинен бути зручним і адаптивним (див. рис. 4.4) у використанні для всіх категорій користувачів. Цей критерій є важливим, оскільки він забезпечує користувачам комфортне та ефективне використання вебсайту.

Вибір цих критеріїв дозволив провести всебічне дослідження якості розробленого вебсайту гімназії №55 із інтелектуальним чат-ботом.

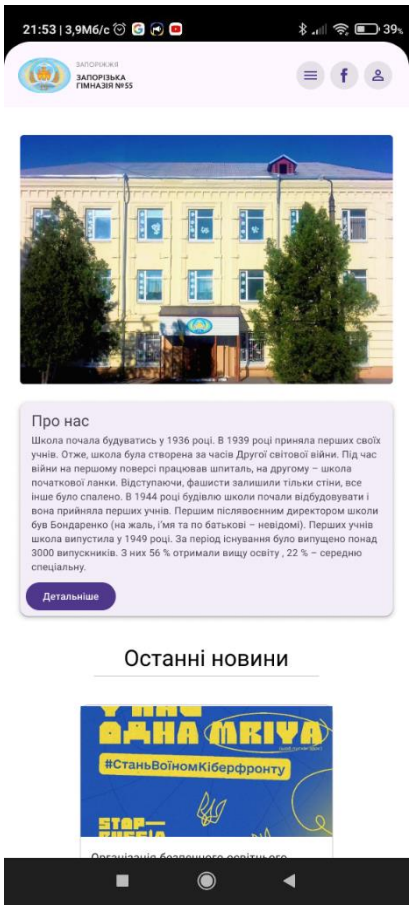


Рисунок 4.4 – Адаптована сторінка

4.2 Дослідження якості вебсайта

Для дослідження якості розробленого вебсайту гімназії №55 із інтелектуальним чат-ботом було використано два методи: опитування та аналіз даних про відвідування та використання вебсайту.

Опитування було проведено за допомогою онлайн-опитувальника. Він був

розміщений на вебсайті гімназії та доступний для всіх відвідувачів. Опитувальник містив 4 запитання, які були розроблені з урахуванням обраних критеріїв дослідження:

– чи вважаєте ви, що інформація на вебсайті є доступною та зрозумілою;
– чи є інформація на вебсайті інформативною;
– чи вважаєте ви, що вебсайт зручний у використанні;
– чи є конкретні моменти на вебсайті які можна покращити?

Дані, отримані за допомогою опитування та аналізу даних про відвідування та використання вебсайту, були проаналізовані з урахуванням обраних критеріїв дослідження (див. рис. 4.5).

Більшість користувачів (92%) вважають, що інформація на вебсайті є доступною та зрозумілою. Це означає, що вебсайт відповідає обраному критерію якості. Інформація про події та новини гімназії була оцінена як найбільш інформативна. Це означає, що вебсайт відповідає потребам користувачів у цій інформації. Вебсайт був оцінений як зручний у використанні для більшості користувачів. Це означає, що вебсайт відповідає обраному критерію якості.

Дані про відвідування та використання вебсайту були отримані за допомогою монітору ресурсів хостингу (див. рис. 4.6). Ці дані містять інформацію про такі показники, як:

– кількість запитів статичної інформації;
– кількість запитів динамічної інформації.

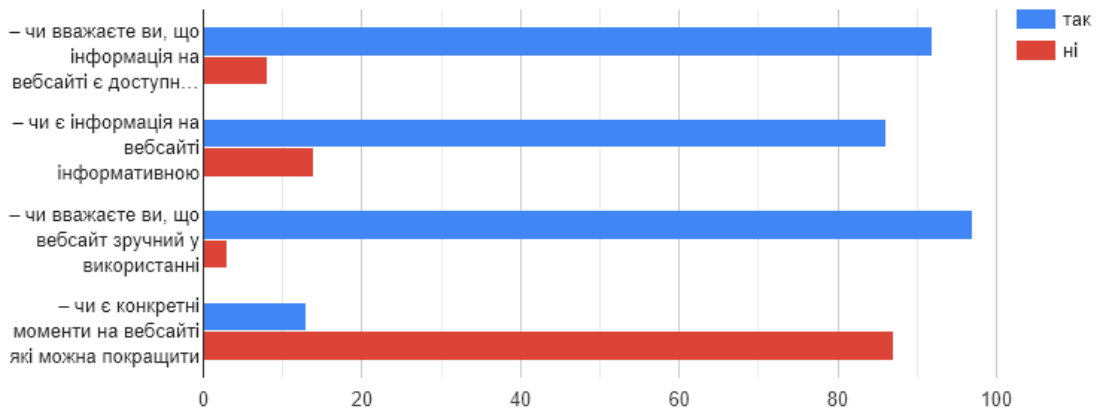


Рисунок 4.5 – Результати опитування

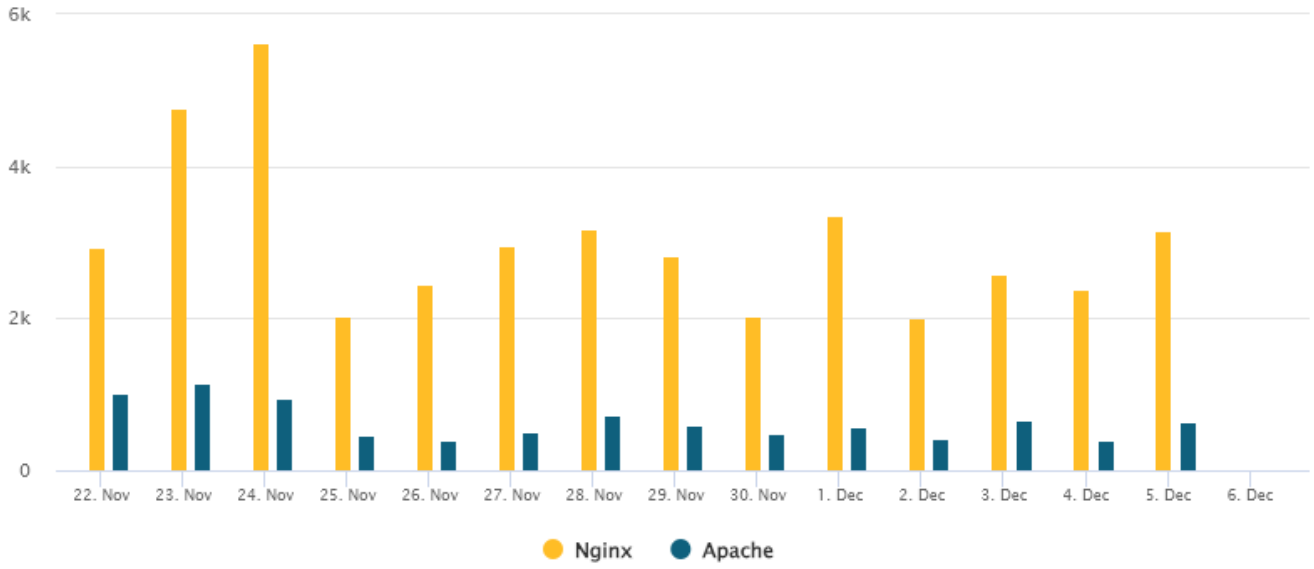


Рисунок 4.6 – Результати монітору ресурсів про кількість запитів

Опитування, проведене серед користувачів чат-ботів, показало, що більшість користувачів задоволені користуванням чат-ботом (див. рис. 4.7). Але можна покращити точність відповідей чат-бота, щоб вони були більш інформативними та корисними. Ці зміни можуть допомогти зробити чат-бот ще більш корисним та зручним для користувачів.

У цілому, результати дослідження показали, що розроблений вебсайт гімназії №55 із інтелектуальним чат-ботом відповідає обраним критеріям якості. Він доступний, інформативний та зручний у використанні, відповідає потребам більшості користувачів.

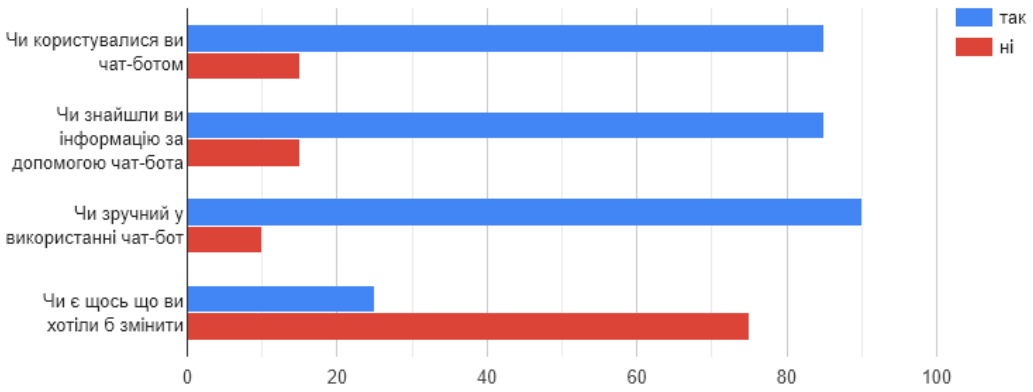


Рисунок 4.7 – Результати опитування про використання чат-бота

Однак, є деякі моменти, які можна покращити. Наприклад, можна розширити інформацію про додаткові освітні послуги, які надає гімназія. Додати кнопки "Поділитися" на сторінки вебсайту, щоб користувачі могли легко ділитися інформацією зі своїми друзями та колегами, покращити точність відповідей чат-бота.

Загалом, розроблений вебсайт є цінним інструментом для комунікації та взаємодії між гімназією та її спільнотою. Він може допомогти гімназії поліпшити якість навчання та взаємодії з її оточенням.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено вебсайт гімназії №55 із інтелектуальним чат-ботом. Мета роботи полягала в підвищенні якості навчання у гімназії №55 за рахунок розробки вебсайту гімназії із інтелектуальним чат-ботом. Для досягнення цієї мети були виконані наступні завдання:

– проаналізовано засоби розробки вебсайтів і визначено вимоги до розроблюваного вебсайту;

– спроєктовано вебсайт;

– реалізовано вебсайт і інтелектуальний чат-бот;

– проведено дослідження якості розробленого вебсайту.

У результаті виконання дипломної роботи було досягнуто поставленої мети. Вебсайт гімназії №55 із інтелектуальним чат-ботом відповідає всім вимогам, які були визначені на початку роботи. Він має зручний і інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко знаходити необхідну інформацію. Інтелектуальний чат-бот може відповідати на запитання користувачів, надавати інформацію про гімназію тощо.

Проведене дослідження якості розробленого вебсайту показало, що він відповідає очікуванням користувачів. Користувачі оцінили вебсайт високою оцінкою, зазначивши, що він зручний у використанні, містить необхідну інформацію, а інтелектуальний чат-бот є корисним помічником.

Розроблений вебсайт може бути використаний для підвищення якості навчання у гімназії №55. Він може використовуватися для надання інформації про гімназію, для організації дистанційного навчання, для допомоги у виконанні навчальних завдань тощо. Інтелектуальний чат-бот може бути використаний для надання відповідей на запитання користувачів, для допомоги у виконанні навчальних завдань тощо.

У ході виконання дипломної роботи були отримані наступні результати:

– проведено детальний аналіз інструментів розробки вебсайтів;

– розроблено вебсайт гімназії №55 із інтелектуальним чат-ботом;

– проведено дослідження якості розробленого вебсайту.

Отримані результати можуть бути використані для розробки вебсайтів гімназій інших міст України.

Для подальшого розвитку вебсайту гімназії №55 можна запропонувати наступні рекомендації:

– розширити функціональність вебсайту, додавши нові розділи та можливості;

– вдосконалити інтелектуального чат-бота, додавши йому нові можливості та збільшивши обсяг даних, на яких він навчається.

Реалізація цих рекомендацій дозволить зробити вебсайт гімназії №55 ще більш корисним та ефективним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Мар'єнко О., Іванченко А. Із досвіду: використання конструкторів сайтів в освітньому процесі. Збірник тез наукових доповідей здобувачів вищої освіти Бердянського державного педагогічного університету на Днях науки 10 листопада 2022 року. Том 3. Природничі науки. Бердянськ : БДПУ, 2022. – С. 64-66

2 Особливості розробки сайту-портфоліо фотографа. Нотатки сучасної науки : Мультидисциплінар. наук. часопис, м. м. Харків, 12 трав. 2023 р. Харків, 2023. С. 25.

3 Конструктор сайтів | Створіть сайт безкоштовно вже сьогодні | Wix.com. wix.com. URL: https://uk.wix.com (дата звернення: 03.10.2023).

4 Панджакідзе С. Т. Корабльов В. А. Використання no-code технологій у шкільному навчанні інформатики: нові можливості та ефективність. Інформатика, інформаційні системи та технології : двадцята всеукр. Конф. Студентів і молодих науковців, м. Одеса, 28 квіт. 2023 р. Одеса, 2023. С. 160–161.

5 Женченко М. І. Технології макетування і верстання інтерактивних електронних видань. Обрії друкарства. 2020. № 1(8). С. 72–81. URL: https://doi.org/10.20535/2522-1078.2020.1(8).190089 (дата звернення: 09.10.2023).

6 Конструктор сайтів або CMS: що краще? PROject SEO. Просування сайтів Львів, розкрутка сайтів і Інтернет-магазинів у Львові, Києві PROject SEO. URL: https://project-seo.net/blog-uk/kontruktor-sajtiv-vs-cms/ (дата звернення: 12.10.2023).

7 Usage Statistics and Market Share of Content Management Systems, December 2023. W3Techs - extensive and reliable web technology surveys. URL: https://w3techs.com/technologies/overview/content_management (date of access: 15.10.2023).

8 Documentation WordPress. Documentation. URL: https://wordpress.org/documentation/ (date of access: 18.10.2023).

9 Занкіна Я. В.,Зінчук М. С. Порівняльна характеристика систем управління контентом з відкритим вихідним кодом // Політ. Сучасні проблеми науки : тези доповідей XXI Міжнародної науково-практичної конференції здобувачів вищої освіти і молодих учених . – Національний авіаційний університет.

– Київ, 2021. - С. 38-39

10 Корчевський Р. Дослідження безкоштовних систем керування вмістом для створення веб-сайтів / Корчевський Р. // Збірник тез X Всеукраїнської студентської науково-технічної конференції „Природничі та гуманітарні науки. Актуальні питання“, 25-26 квітня 2017 року. — Т. : ТНТУ, 2017. — Том 1. — С. 57. — (Секція: Інформаційні технології)

11 Простяк В. розробка web-сайту інтернет-магазину для продажу спортивного одягу на основі Magento 2 / В. Простяк // Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології“, 7–8 грудня 2022 року. — Т. : ТНТУ, 2022. — С. 127. — (Програмна інженерія та моделювання складних розподілених систем).

12 Боднар Л. В. Методичні рекомендації щодо створення Інтернет-сайту освітнього закладу / Л. В. Боднар; Державний заклад «Південноукраїнський національний педагогічний університет імені К. Д. Ушинського». – Одеса, 2019. – 52 с.

13 Іщенко М. Д., Аналіз життєвого циклу та процесу розробки WEB-додатків / М. Д. Іщенко, М. Ю. Білоус // «Automation and Development of Electronic Devices» ADED-2022: Collection of Students' Scientific Paper. – Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2022. – Part 2. – P. 139-142.

14 Kotenko, N., Zhyrova, T., Chybaievskyi, V., & Desiatko, A. (2019). ДОСЛІДЖЕННЯ ОСНОВНИХ ТЕНДЕНЦІЙ СУЧАСНОЇ РОЗРОБКИ ВЕБ-САЙТІВ. Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2019. – 6–15 с.

15 Express JS – platforma dlia stvorennia storinok [Express JS is a platform for creating pages], 2022. Hlianets tm, [blog] 14 October. Available at: URL:https://glyanec.net/ua/blog/express-js-platforma-dlya-stvorennya-storinok (дата звернення: 13.11.2023)

16 Wieruch R. The Road to React: Your journey to master plain yet pragmatic React.js. Independently published, 2018. 226 p.

17 Herron, D., 2020. Node.js Web Development: Server-side web development made easy with Node 14 using practical examples. 5th ed. London: Packt Publishing.

JSON and BSON, n.d. MongoDB. [online] URL: https://www.mongodb.com/json-and-bson (дата звернення: 19.11.2023)

18 Швець Г.К., Розробка інтерфейсу терміналу самообслуговування для аптеки / Г.К. Швець, М.Ю. Тягунова // Тиждень науки-2023. Факультет комп’ютерних наук і технологій : наук.-техн. конф., 24-28 квітня 2023 р. : тези доп. – / Редкол. : Вадим ШАЛОМЄЄВ (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2023. – С. 87-88 – 1 електрон. опт. диск (DVD-ROM); 12 см. – назва з тит. екрана