Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-сайту ”Piggyback” з використанням технологій HTML, CSS та JS.

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Шинкаренко Л.М.
(підпис) (прізвище та ініціали)

Керівник _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль - 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Шинкаренко Лесі Максимівні
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка веб-сайту "Piggyback" з використанням технологій HTML, CSS та JS.

Керівник роботи Литвиненко Ярослав Володимирович, д.т.н., професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*7*» *травня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 28 червня 2025 р.
3. Вихідні дані до роботи Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Постанова задачі виконання кваліфікаційної роботи, 1.1 Аналіз предметної області, 1.2 Формування вимог розробки веб-сайту "Piggyback", 1.3 Пошук акторів та варіантів використання веб-сайту, 1.4 Опис ключових варіантів використання веб-сайту, 1.5 Вибір технологій розробки, 1.6 Висновок до першого розділу, РОЗДІЛ 2. Проєктування та реалізація веб-сайту "Piggyback", 2.1 Аналіз архітектури веб-сайту, 2.2 Вибір інструментів та бібліотек для розробки, 2.3 Проєктування структури веб-сайту, 2.4 Висновок до другого розділу, РОЗДІЛ 3. Реалізація та тестування веб-сайту "Piggyback", 3.1 Реалізація інтерфейсу, 3.2 Реалізація інтерактивних елементів, 3.3 Тестування та оптимізація, 3.4 Висновок до третього розділу, РОЗДІЛ 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел.

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин Віктор Степанович		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	08.02.2025-15.05.2025	
3.	Виконання дослідження щодо реалізації веб-сайту	19.02.2025-23.05.2025	
	Розроблення веб-сайту Piggyback		
4.	Оформлення розділу «Постанова задачі виконання кваліфікаційної роботи»	26.05.2025-02.06.2025	
5.	Оформлення розділу «Проєктування та реалізація веб-сайту Piggyback»	26.05.2025-02.06.2025	
6.	Оформлення розділу «Тестування веб-сайту Piggyback	26.05.2025-02.06.2025	
	»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	09.06.2025-13.06.2025	
8.	Виконання завдання до підрозділу «Основи охорони	09.06.2025-13.06.2025	
	праці»		
9.	Оформлення кваліфікаційної роботи	10.06.2025-12.06.2025	
10.	Нормоконтроль	13.06.2025-16.06.2025	
11.	Перевірка на плагіат	18.06.2025	
12.	Попередній захист кваліфікаційної роботи	20.06.2025	
13.	Захист кваліфікаційної роботи	29.06.2025	

Студент _____ Шинкаренко Л.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-сайту ”Piggyback” з використанням технологій HTML, CSS та JS. // Кваліфікаційна робота освітнього рівня «Бакалавр» // Шинкаренко Леся Максимівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-43 // Тернопіль, 2025 // С. 50, рис. – 15, табл. – 3, додат. – 12, бібліогр. – 30.

Ключові слова: веб-розробка, javascript, frontend, інтерфейс користувача, соціальна мережа, стрімінговий сервіс, адаптивний дизайн, тестування.

Кваліфікаційна робота присвячена розробці веб-сайту "Piggyback", платформи, що інтегрує соціальні функції, музичний і відео стрімінг. Метою дослідження є створення зручного й безпечного веб-сайту для аудиторії з використанням HTML, CSS і JavaScript.

В першому розділі було проведено аналіз потреб користувачів, сформовано вимоги до функціональності та інтерфейсу, визначено акторів і сценарії використання, а також виконано порівняння з аналогами.

В другому розділі було розроблено архітектуру багатосторінкового застосунку, обрано технології та інструменти, спроєктовано структуру сторінок із інтерактивними елементами, такими як каруселі й плеєри, забезпечивши адаптивний дизайн.

В третьому розділі було реалізовано інтерфейс та інтерактивні компоненти, проведено тестування авторизації, відео та адаптації.

ANNOTATION

Development of the "Piggyback" Website Using HTML, CSS, and JS// Qualification work of the educational level «Bachelor» // Shynkarenko Lesia Maksymivna // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2025 // P. 50, fig. – 15 , tabl. – 3, annexes. – 12, references – 30.

Keywords: web development, javaScript, frontend, user interface, responsive design, social network, streaming service, testing.

The qualification work is dedicated to the development of the "Piggyback" website, a platform that integrates social features, music, and video streaming. The aim of the study is to create a convenient and safe website for the audience using HTML, CSS, and JavaScript.

In the first section, an analysis of user needs was conducted, requirements for functionality and interface were formulated, actors and usage scenarios were defined, and a comparison with analogs was performed.

In the second section, the architecture of a multi-page application was developed, technologies and tools were selected, the structure of pages with interactive elements such as carousels and players was designed, ensuring an adaptive design.

In the third section, the interface and interactive components were implemented, and testing of authorization, video, and adaptation was carried out.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

HTML (англ. HyperText Markup Language) – мова гіпертекстової розмітки.

JS (англ. JavaScript) – мова сценаріїв для динамічної взаємодії з веб-сторінками.

MPA (англ. Multi-Page Application) – багатосторінковий застосунок.

UI (англ. User Interface) – інтерфейс користувача.

Адаптивний дизайн – підхід до розробки інтерфейсу, який забезпечує коректне відображення на різних пристроях.

Веб-розробка – процес створення та підтримки веб-сайтів.

Інтерфейс користувача – елементи, з якими взаємодіє користувач у веб-застосунку.

Клієнтська частина – частина веб-додатку, яка виконується у браузері користувача.

Стрімінговий сервіс – онлайн-сервіс для передачі мультимедійного контенту в реальному часі.

Фронтенд – частина веб-додатку, яка відповідає за відображення і взаємодію з користувачем.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок швидкого розвитку інформаційних технологій, сучасні користувачі стали частіше використовувати різноманітні веб-платформи для соціальної взаємодії, прослуховування музики чи перегляду відеоконтенту. Однак необхідність використання кількох окремих сервісів створює незручності, пов’язані з перемиканням між додатками, різними інтерфейсами та управлінням кількома обліковими записами. Розробка єдиної платформи, яка об’єднує соціальні функції, музичний та відео стрімінг з можливістю синхронізованого перегляду з іншими користувачами, є актуальним напрямком для досліджень у галузі веб-розробки. Такий підхід спрощує взаємодію користувачів із контентом та відповідає сучасним тенденціям до інтеграції та крос-платформеності, що забезпечує зручність і доступність для широкої аудиторії.

Мета і задачі дослідження. Метою кваліфікаційної роботи освітнього рівня «Бакалавр» є розробка веб-сайту «Piggyback» з використанням технологій HTML, CSS і JavaScript для створення зручної платформи, яка об’єднує функціонал соціальних мереж, музичного та відео стрімінгу. Для досягнення поставленої мети потрібно виконати ряд завдань:

– Проаналізувати предметну область та потреби користувачів.

– Сформувати вимоги до функціональності та інтерфейсу веб-сайту.

– Визначити акторів і ключові сценарії використання платформи.

– Обґрунтувати вибір для реалізації проєкту.

– Розробити та реалізувати адаптивний інтерфейс із підтримкою інтерактивних елементів.

– Провести тестування веб-сайту для забезпечення його функціональності та зручності використання.

Практичне значення одержаних результатів. Розроблений веб-сайт забезпечує користувачам зручний доступ до інтегрованих функцій соціальних мереж, прослуховування музики та перегляду відео в єдиному інтерфейсі.

Платформа дозволяє створювати та редагувати пости і сторіс, формувати плейлисти, переглядати фільми, зокрема у синхронізованому режимі з іншими користувачами, що сприяє підвищенню соціальної взаємодії та зручності. Результати роботи можуть бути використані для створення комерційних або некомерційних веб-платформ, які об’єднують різноманітний функціонал, а також як основа для подальшого розвитку крос-платформених додатків із використанням сучасних веб-технологій.

РОЗДІЛ 1. ПОСТАНОВА ЗАДАЧІ ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Аналіз предметної області

Сучасний розвиток інформаційних технологій сприяє інтеграції різноманітних сервісів у єдині платформи, що забезпечують зручність і ефективність взаємодії користувачів із цифровим контентом.

Предметна область дослідження охоплює створення веб-платформ, які інтегрують соціальні, мультимедійні та інтерактивні функції. Основною проблемою, яку вирішує "Piggyback", є розрізненість сучасних сервісів, що змушує користувачів переходити між різними додатками для виконання повсякденних завдань, таких як спілкування, прослуховування музики чи перегляд фільмів. Однак використання кількох платформ ускладнює взаємодію, підвищує витрати часу на перемикання між інтерфейсами та вимагає управління кількома обліковими записами. "Piggyback" пропонує вирішення цієї проблеми шляхом створення єдиного адаптивного веб-сайту, який об’єднує зазначені функції та забезпечує швидке перемикання між ними.

Аналіз потреб користувачів [1] показує, що цільова аудиторія включає молодь і дорослих, які активно використовують соціальні мережі, слухають музику та переглядають відео. Ключовими вимогами до платформи є:

– Інтуїтивно зрозумілий дизайн із можливістю швидкого перемикання між соціальними функціями, музичним і відео плеєром.

– Крос-платформенність для забезпечення широкого охоплення аудиторії.

– Підтримка створення та редагування постів та сторіс, формування плейлистів, а також синхронізованого перегляду відео з іншими користувачами.

– Можливість авторизації для збереження особистих даних, таких як улюблені треки, плейлисти чи історія переглядів.

Для порівняння, такі платформи, як Instagram, Spotify і Netflix, які є лідерами у своїх сегментах, мають обмеження. Instagram не підтримує музичний

чи відеострімінг, Spotify зосереджений виключно на музиці, а Netflix не має соціальних функцій. "Piggyback" усуває ці недоліки, пропонуючи інтегроване рішення, яке поєднує переваги всіх трьох платформ. Наприклад, функція синхронізованого перегляду фільмів із друзями, реалізована за допомогою JavaScript, додає соціальний аспект до відеострімінгу, що є унікальною особливістю проєкту.

Отже, предметна область розробки веб-сайту "Piggyback" охоплює створення інтегрованої платформи, яка відповідає сучасним потребам користувачів у зручному доступі до соціальних, музичних і відеофункцій. Аналіз аналогів і потреб цільової аудиторії підтверджує актуальність проєкту та необхідність його реалізації з використанням сучасних веб-технологій.

1.2 Формування вимог розробки веб-сайту "Piggyback"

На основі проведеного аналізу предметної області сформовано вимоги до функціональності та характеристик системи, які поділяються на функціональні та нефункціональні. Ці вимоги враховують потреби цільової аудиторії та сучасні тенденції веб-розробки, забезпечуючи зручність, доступність і ефективність взаємодії користувачів із платформою.

Функціональні вимоги визначають основні можливості [2], які має надавати веб-сайт "Piggyback". До них належать:

– Реалізація системи входу та створення облікового запису для збереження персональних даних, таких як історія переглядів, плейлисти чи створений контент.

– Можливість створювати, редагувати та видаляти пости й сторіс, подібно до функціоналу Instagram, із підтримкою текстового, графічного та мультимедійного контенту.

– Доступ до музичних треків, створення та редагування плейлистів, а також управління відтворенням (пауза, перемотування, регулювання гучності).

– Можливість перегляду фільмів і серіалів із функцією синхронізованого перегляду, що дозволяє кільком користувачам дивитися відео одночасно з синхронізацією відтворення.

– Підтримка коментарів, лайків до постів і сторіс, а також можливість ділитися контентом із іншими користувачами.

– Реалізація інтуїтивного інтерфейсу для швидкого переходу між соціальними функціями, музичним плеєром і відеоплеєром.

Нефункціональні вимоги визначають якісні характеристики системи [3], які забезпечують її ефективність і зручність використання:

– Веб-сайт має коректно відображатися на різних пристроях (ПК, планшети, смартфони) із роздільною здатністю екрану від 320 до 1920 пікселів.

– Час відгуку інтерфейсу не повинен перевищувати 2 секунд для основних операцій (завантаження сторінки, перемикання між модулями).

– Підтримка сучасних веб-браузерів (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) для забезпечення широкого охоплення аудиторії.

– Дизайн має бути простим і зрозумілим, із чіткою навігацією та мінімальною кількістю дій для виконання основних функцій (наприклад, створення посту чи запуску синхронізованого перегляду).

Сформовані вимоги до веб-сайту "Piggyback" відображають потреби цільової аудиторії та сучасні стандарти веб-розробки. Вони слугують основою для подальшого проєктування та реалізації платформи, забезпечуючи її функціональність, зручність і доступність для користувачів.

1.3 Пошук акторів та варіантів використання веб-сайту

Для розробки веб-сайту "Piggyback" необхідно визначити акторів, які взаємодіятимуть із системою, та основні сценарії використання [4], що відображають їхню взаємодію з платформою. На основі аналізу предметної області та сформованих вимог ідентифіковано акторів і ключові варіанти використання, які є основою для подальшого проєктування системи.

Актори представляють ролі користувачів або зовнішніх систем, що взаємодіють із веб-сайтом [4]. Для "Piggyback" визначено таких акторів:

Авторизований користувач — основний користувач із зареєстрованим обліковим записом, який має доступ до всіх функцій платформи, включаючи створення контенту, прослуховування музики, перегляд відео та соціальну взаємодію.

Адміністратор — користувач із розширеними правами, відповідальний за модерацію контенту (видалення невідповідних постів) і управління обліковими записами користувачів.

Щоб краще зрозуміти, як користувачі взаємодіють із сайтом, ми розглянули кілька типових сценаріїв використання. Для веб-сайту "Piggyback" визначено такі основні сценарії:

Реєстрація та авторизація — дозволяє авторизованим користувачам і гостям створювати обліковий запис або входити в систему для доступу до персоналізованих функцій.

Створення та редагування контенту — авторизований користувач може створювати, редагувати або видаляти пости та сторіс із текстовим, графічним чи мультимедійним вмістом.

Пошук і прослуховування музики — авторизований користувач може шукати треки, відтворювати їх і створювати плейлисти.

Перегляд відеоконтенту — авторизований користувач переглядає фільми чи серіали та може ініціювати синхронізований перегляд із іншими користувачами.

Соціальна взаємодія — авторизований користувач може коментувати, лайкати або ділитися контентом.

Модерація контенту — адміністратор переглядає та видаляє невідповідний контент або керує обліковими записами.

Перемикання між модулями — усі актори можуть швидко переходити між соціальними, музичними та відеофункціями через інтуїтивний інтерфейс.

Для візуалізації взаємодії акторів із системою розробимо діаграму варіантів використання (рис. 1.1).

Рисунок 1.1 – Діаграма варіантів використання веб-сайту "Piggyback"

Зв’язки між акторами та сценаріями позначені стрілками, що вказують на їхню взаємодію із системою.

Визначення акторів і варіантів використання веб-сайту забезпечує чітке розуміння ролей користувачів і сценаріїв їхньої взаємодії з платформою. Ці дані стануть основою для більш деталізованого опису ключових варіантів використання та подальшого проєктування системи.

1.4 Опис ключових варіантів використання веб-сайту

Підключення ключових варіантів використання веб-сайту "Piggyback" дозволяє деталізувати взаємодію користувачів із платформою. На основі акторів і сценаріїв, визначених у попередньому підрозділі, обрано основні варіанти використання, що відображають ключовий функціонал: реєстрація та авторизація, створення та редагування контенту, перегляд відеоконтенту з синхронізацією та пошук і прослуховування музики. Ці сценарії відповідають

функціональним вимогам [5] і демонструють, як авторизовані користувачі взаємодіють із системою. Для узагальнення варіантів використання наведено таблицю 1.1, яка підсумовує їхні характеристики.

Таблиця 1.1 – Ключові варіанти використання веб-сайту "Piggyback"

Назва сценарію	**Актор**	**Мета**	**Передумова**	**Результат**
Реєстрація та авторизація	Авторизований користувач	Отримання доступу до персоналізованих функцій	Користувач не авторизований	Доступ до всіх функцій платформи
Створення та редагування контенту	Авторизований користувач	Створення/редагування постів і сторіс	Користувач авторизований	Опублікований або відредагований контент
Перегляд відеоконтенту з синхронізацією	Авторизований користувач	Перегляд відео з синхронізацією	Користувач у відеомодулі	Синхронізований перегляд відео
Пошук і прослуховування музики	Авторизований користувач	Прослуховування музики, створення плейлистів	Користувач у музичному модулі	Відтворення музики, збереження плейлиста

Наведені в таблиці ключові варіанти використання веб-сайту "Piggyback" узагальнюють основні сценарії взаємодії авторизованих користувачів із платформою. Ці дані є основою для проєктування та реалізації системи, забезпечуючи її відповідність потребам користувачів і функціональним вимогам.

1.5 Вибір технологій розробки

Розробка веб-сайту спрямована на створення платформи, яка об’єднує функції соціальних мереж, музичного та відео стрімінгу, відповідно до функціональних і нефункціональних вимог та сценаріїв використання. Сайт

забезпечує створення постів і сторіс, прослуховування музики, формування плейлистів і перегляд відео з можливістю синхронізації для авторизованих користувачів, а також модерацію контенту адміністратором. Для реалізації проєкту обрано технології HTML, CSS і JavaScript, які відповідають потребам фронтенд-розробки [6], забезпечуючи адаптивність, інтерактивність і зручність інтерфейсу. Цей підрозділ обґрунтовує їх вибір і описує їх роль у проєкті.

HTML обрано для створення структури веб-сайту, що є основою для відображення контенту. HTML забезпечує семантичну розмітку сторінок, що сприяє зрозумілості коду та доступності платформи. У контексті "Piggyback" HTML використовується для:

– Створення структури модулів (соціального, музичного, відеомодуля).

– Розмітки форм для авторизації, створення постів і сторіс.

– Організації елементів інтерфейсу, таких як плеєри для музики та відео.

CSS відповідає за стилізацію та адаптивність інтерфейсу, що є ключовою нефункціональною вимогою. CSS дозволяє створювати візуально привабливий і адаптивний дизайн, який коректно відображається на пристроях із роздільною здатністю від 320 до 1920 пікселів. Для "Piggyback" CSS використовується для:

– Реалізації адаптивного макета з використанням Flexbox і Grid для розташування елементів.

– Стилізації інтерактивних компонентів (кнопок, форм, плеєрів).

– Адаптації дизайну до вимог платформи.

JavaScript забезпечує інтерактивність і динамічне оновлення контенту, що необхідно для реалізації складних функцій, таких як синхронізований перегляд відео чи обробка подій. JavaScript дозволяє створювати динамічний і зручний інтерфейс без використання серверної логіки. Для "Piggyback" JavaScript використовується для:

– Обробки подій користувача.

– Імітації авторизації та збереження даних.

– Реалізації синхронізованого перегляду відео.

Використовувались додаткові бібліотеки для прискорення розробки [7]. Наприклад, Bootstrap, який спростив створення адаптивного дизайну, а CSS для гнучкості та оптимізації.

Для узагальнення обраних технологій наведено таблицю, яка описує їх призначення та роль у проєкті (таб. 1.2).

Таблиця 1.2 – Обрані технології розробки веб-сайту "Piggyback"

Технологія	**Призначення**	**Роль у проєкті**
HTML	Створення структури сторінок	Розмітка модулів, форм, елементів інтерфейсу
CSS	Стилізація та адаптивність	Адаптивний дизайн, стилізація компонентів
JavaScript	Інтерактивність і динамічне оновлення	Обробка подій, імітація авторизації, синхронізація

Основний вибір HTML, CSS і JavaScript для розробки веб-сайту "Piggyback" обґрунтований їхньою відповідністю вимогам фронтенд-розробки, простотою реалізації та підтримкою сучасних стандартів. Ці технології є основою для проєктування і реалізації платформи, забезпечуючи її функціональність і зручність для користувачів.

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи проведено комплексний аналіз і підготовку до розробки веб-сайту "Piggyback", який об’єднує функції соціальних мереж, музичного стрімінгу та перегляду відеоконтенту з використанням технологій HTML, CSS і JavaScript. На основі поставлених завдань виконано аналіз предметної області, що підтвердив актуальність створення інтегрованої платформи, яка усуває незручності використання кількох окремих сервісів. Сформовано функціональні та нефункціональні вимоги, які

включають підтримку авторизації, створення контенту, музичного та відеострімінгу, соціальної взаємодії, адаптивності й інтуїтивності інтерфейсу.

Визначено основних акторів системи — авторизованого користувача та адміністратора, а також ключові сценарії використання, такі як реєстрація, створення постів і сторіс, пошук і прослуховування музики, перегляд відео з синхронізацією та модерація контенту. Ці сценарії узагальнено в діаграмі варіантів використання і таблиці, що забезпечує чітке розуміння взаємодії користувачів із платформою. Вибір технологій HTML, CSS і JavaScript обґрунтовано їхньою відповідністю вимогам фронтенд-розробки, зокрема для створення структури, адаптивного дизайну та інтерактивних функцій.

Проведений аналіз і підготовка створюють міцну основу для подальшого проєктування та реалізації веб-сайту "Piggyback". Отримані результати підтверджують можливість створення зручної та функціональної платформи, яка відповідає потребам цільової аудиторії та сучасним стандартам веб-розробки.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ "PIGGYBACK"

2.1 Аналіз архітектури веб-сайту

Для "Piggyback" обрано архітектуру багатосторінкового веб-застосунку, яка передбачає використання окремих HTML-сторінок для кожного модуля платформи (соціального, музичного, відеомодуля). MPA [8] є оптимальним вибором для структурованої організації контенту, де кожен модуль представлений окремою сторінкою, що забезпечує чітке розмежування функціоналу та полегшує навігацію для користувачів. У контексті "Piggyback" MPA реалізується через HTML для структури сторінок, CSS із Flexbox, Grid і медіа-запитами для адаптивного дизайну та JavaScript для обробки подій і забезпечення інтерактивності.

Архітектура веб-сайту складається з чотирьох основних компонентів. Модуль інтерфейсу користувача (UI), який відповідає за відображення контенту та взаємодію з користувачем. Кожен модуль (соціальний, музичний, відео) представлений окремим HTML-файлом, який забезпечує семантичну розмітку сторінок, включаючи форми авторизації, пости, сторіс і мультимедійні плеєри. CSS-файли, посилені бібліотекою Bootstrap, реалізують адаптивний дизайн із використанням Flexbox і Grid, що гарантує коректне відображення на пристроях із роздільною здатністю від 320 до 1920 пікселів. Дизайн базується на макетах, створених у Figma, які адаптовано до вимог платформи.

Модуль обробки подій реалізований через JavaScript-скрипти для управління інтерактивними елементами, такими як кнопки, форми та плеєри. Цей модуль обробляє події користувача (клацання, введення тексту, запуск відео) на кожній сторінці. Наприклад, створення посту чи запуск синхронізованого перегляду відео обробляється через JavaScript-події, з використанням jQuery для спрощення DOM-маніпуляцій.

У відсутність бекенду дані, такі як пости, плейлисти чи історія переглядів, зберігаються в localStorage [9] для імітації персистентності в межах сесії користувача. Статичні ресурси (зображення, аудіо, відео, шрифти) використовуються для імітації мультимедійного контенту, забезпечуючи функціонал без серверної логіки.

Модуль інтерактивностей забезпечує роботу елементів, таких як музичний і відеоплеєри, форми створення контенту та синхронізований перегляд. JavaScript-скріпти координують ці функції, використовуючи Bootstrap для стилізації форм і jQuery для спрощення обробки подій.

Для розробки та оптимізації використано інструменти, такі як VS Code для редагування коду та Gulp [10] для збирання проєкту. Gulp забезпечує швидке збирання сторінок і підтримку модульної структури, що спрощує управління окремими HTML-файлами.

Для візуалізації архітектури розроблено схему (рис. 2.1), яка ілюструє взаємодію компонентів.

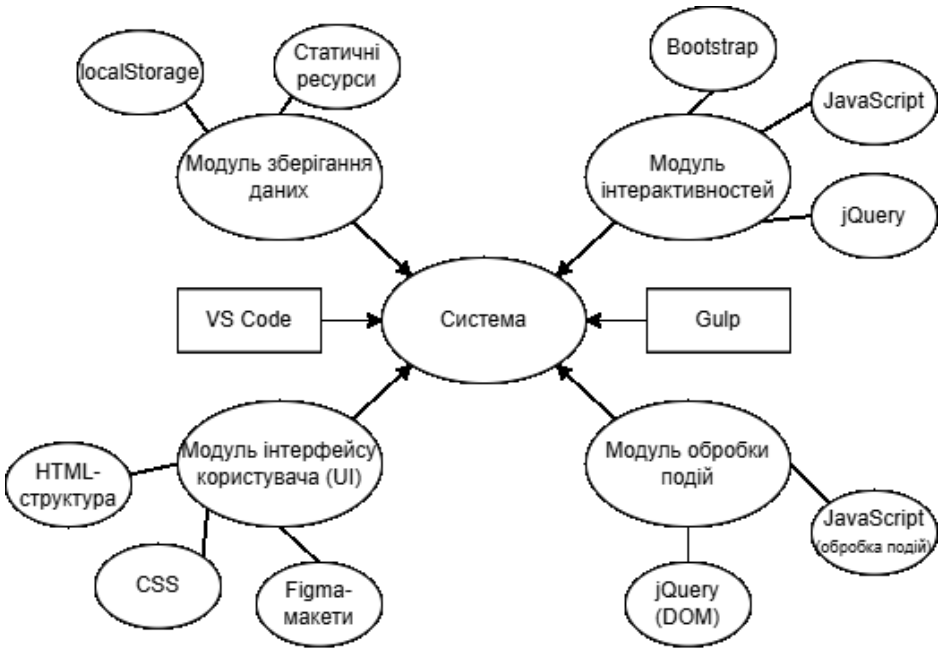

Рисунок 2.1 – Архітектурна схема веб-сайту "Piggyback"

Схема відображає, як модуль UI (HTML, CSS, Bootstrap, Figma-макети) для кожної сторінки взаємодіє з модулем обробки подій (JavaScript, jQuery), який звертається до модуля управління даними (localStorage, статичні ресурси). Наприклад, при створенні плейлиста на сторінці музичного модуля JavaScript записує дані в localStorage, а CSS і Bootstrap стилізують відображення списку треків.

Переваги MPA-архітектури для "Piggyback" включають:

– Кожен модуль представлений окремою сторінкою, що полегшує організацію коду та навігацію для користувачів.

– Використання окремих HTML-файлів спрощує створення та тестування модулів за допомогою VS Code і Gulp.

– MPA забезпечує стабільну роботу в різних браузерах (Chrome, Firefox, Safari, Edge) завдяки стандартам HTML5, CSS3 і ES6.

Недоліки MPA, такі як повільніше перемикання між модулями через перезавантаження сторінок, мінімізуються через оптимізацію статичних ресурсів і JavaScript-коду, а також використання Bootstrap і jQuery для прискорення розробки інтерфейсу. Для зменшення часу завантаження сторінок застосовуються інструменти збирання, які оптимізують збірку та оптимізацію контенту.

Альтернативний підхід — односторінковий застосунок (SPA) [11] — відхилено через складність реалізації динамічного рендерингу всіх модулів на одній сторінці та більший обсяг JavaScript-коду, що могло б ускладнити розробку в межах проєкту.

Архітектура MPA для веб-сайту "Piggyback" є оптимальною для створення структурованої платформи, що відповідає вимогам до авторизації, створення контенту, стрімінгу та синхронізації. Визначені компоненти (UI, обробка подій, управління даними, інтерактивність) та інструменти (VS Code, Gulp, Figma, Bootstrap, jQuery) створюють основу для вибору додаткових інструментів і проєктування структури.

2.2 Вибір інструментів та бібліотек для розробки

Для написання та налагодження коду застосовується Visual Studio Code [12], текстовий редактор із підтримкою синтаксичного підсвічування HTML, CSS і JavaScript. VS Code спрощує керування файлами проєкту, включаючи сторінки модулів, стилі, скрипти та статичні ресурси, такі як зображення, аудіо, відео й шрифти. Збирання проєкту автоматизується через Gulp, який обробляє файли, оптимізує їх і забезпечує розгортання. Gulp компілює стилі з препроцесорів за допомогою gulp-sass і gulp-less, додає вендорні префікси через gulp-autoprefixer для сумісності з браузерами (Chrome, Firefox, Safari, Edge) і зменшує обсяг CSS за допомогою gulp-clean-css [13]. Зображення оптимізуються плагіном gulp-imagemin, що прискорює завантаження сторінок. Плагіни gulp-newer, gulp-rename, gulp-empty, gulp-sass-glob, gulp-less-glob і gulp-rsync полегшують керування файлами, імпорт стилів і синхронізацію з віддаленим сервером. Для тестування змін у реальному часі використовується BrowserSync, який оновлює сторінки при змінах у коді, прискорюючи перевірку адаптивності.

Інтерфейс сторінок реалізовано з використанням Bootstrap [14], який забезпечує адаптивний дизайн форм авторизації, кнопок і компонентів соціального модуля, таких як пости й сторіс. Його сітка гарантує коректне відображення на екранах із роздільною здатністю від 320 до 1920 пікселів. Інтерактивність досягається завдяки jQuery, що спрощує обробку подій і маніпуляції з DOM для створення контенту, навігації між модулями та керування плеєрами. У соціальному модулі Chart.js застосовується для відображення статистики користувача, наприклад, кількості постів чи прослуханих треків. Slick Carousel створює слайдери для постів, сторіс і плейлистів у соціальному та музичному модулях, забезпечуючи зручну навігацію. У відеомодулі Video.js реалізує відеоплеєри, що підтримують відтворення та імітацію синхронізованого перегляду.

Сумісність JavaScript-коду з різними браузерами забезпечує Babel [15] із модулями babel/core, babel/preset-env і babel-loader, які обробляють синтаксис

ES6+. Плагіни babel-plugin-module-resolver і babel-plugin-root-import оптимізують імпорт модулів. gulp-sourcemaps генерує sourcemaps для полегшення налагодження CSS і JavaScript. webpack-stream інтегрує Webpack із Gulp для збирання модульних JavaScript-скриптів. Макети дизайну, створені в Figma, стали основою для інтерфейсу, із них експортовано іконки та візуальні елементи, які адаптовано до стилів Bootstrap і власних CSS-файлів.

Обрані інструменти та бібліотеки узагальнено в таблиці, яка ілюструє їх призначення та роль у проєкті.

Таблиця 2.1 – Обрані інструменти та бібліотеки для розробки веб-сайту "Piggyback"

Інструмент/Бібліотека	**Призначення**	**Роль у проєкті**
VS Code	Редагування коду	Написання та налагодження HTML, CSS, JS
Gulp	Збирання проєкту	Автоматизація обробки файлів, оптимізація
BrowserSync	Синхронізація змін	Тестування в реальному часі
gulp-sass, gulp-less	Компіляція препроцесорів	Створення CSS-стилів
gulp-autoprefixer	Сумісність CSS	Додавання вендорних префіксів
gulp-clean-css	Мініфікація CSS	Оптимізація стилів
gulp-imagemin	Оптимізація зображень	Зменшення обсягу статичних ресурсів
Bootstrap	Стилізація	Адаптивний дизайн форм, кнопок, сітки

Продовження таблиці 2.1

jQuery	Інтерактивність	Обробка подій, DOM-маніпуляції
Chart.js	Візуалізація даних	Відображення статистики користувача
Slick Carousel	Слайдери	Навігація постами, сторіс, плейлистами
Video.js	Відеоплеєр	Відтворення відео, синхронізований перегляд
Babel	Транскомпіляція JS	Сумісність із браузерами
gulp-sourcemaps	Налагодження	Генерація sourcemaps для CSS і JS
webpack-stream	Збирання JS	Модульна структура скриптів
Figma	Дизайн UI	Експорт іконок і візуальних елементів

Обрані інструменти та бібліотеки забезпечують ефективну розробку "Piggyback", відповідаючи вимогам до структури, продуктивності та зручності інтерфейсу.

2.3 Проєктування структури веб-сайту

З урахуванням архітектури багатосторінкового застосунку та інструментів збирання й дизайну, створення веб-сайту "Piggyback" вимагає чіткої організації його компонентів. Використання Gulp для автоматизації процесів і Bootstrap для адаптивного оформлення дозволяє сформувати зручну структуру, що полегшує доступ до функціоналу платформи.

Користувацький шлях розпочинається зі сторінки авторизації, де реалізовано форми для входу та реєстрації з використанням Bootstrap для адаптивного відображення та jQuery [16] для обробки подій. Після авторизації відкривається головна сторінка, що слугує центральним хабом контенту. Тут розташовано перемикач, який дозволяє користувачу обрати перегляд фільмів і серіалів із інтегрованим відеоплеєром на основі Video.js [17] або музичний модуль із плейлистами, де Slick Carousel забезпечує зручну навігацію. З головної сторінки доступні додаткові розділи через навігаційне меню, яке включає опції для перегляду стрічки, плейлистів, обміну повідомленнями та профілю. Натискання на профіль відкриває налаштування та опції зміни акаунта, розширюючи можливості користувача.

Файлова структура проєкту організовано з урахуванням обробки Gulp, де вихідні файли зберігаються в окремих папках для HTML, CSS, JavaScript і статичних ресурсів, таких як зображення, аудіо, відео та іконки, експортовані з Figma. Оптимізовані версії цих файлів розміщуються в папці призначення після збирання, що забезпечує швидке завантаження сторінок завдяки плагінам gulp-imagemin і gulp-clean-css. Така організація підтримує модульну розробку, де кожен тип контенту має власний набір стилів і скриптів.

Модулі сайту реалізовано через окремі сторінки, що відображають їхній функціонал. Соціальний модуль включає стрічку з постами та візуальними елементами, де Slick Carousel забезпечує слайдерну навігацію, а Chart.js [18] відображає статистику користувача. Музичний модуль пропонує плейлисти з інтуїтивним переглядом, а відеомодуль фокусується на відтворенні контенту через Video.js із імітацією синхронізованого перегляду. Дані, такі як профіль чи плейлисти, зберігаються в localStorage, що дозволяє імітувати персистентність без бекенду. Адаптивність досягається через Bootstrap Grid, Flexbox і медіа-запити, гарантуючи коректне відображення на екранах від 320 до 1920 пікселів.

Для наочності структури наведено схему, що ілюструє основну організацію сайту на рисунку 2.2.

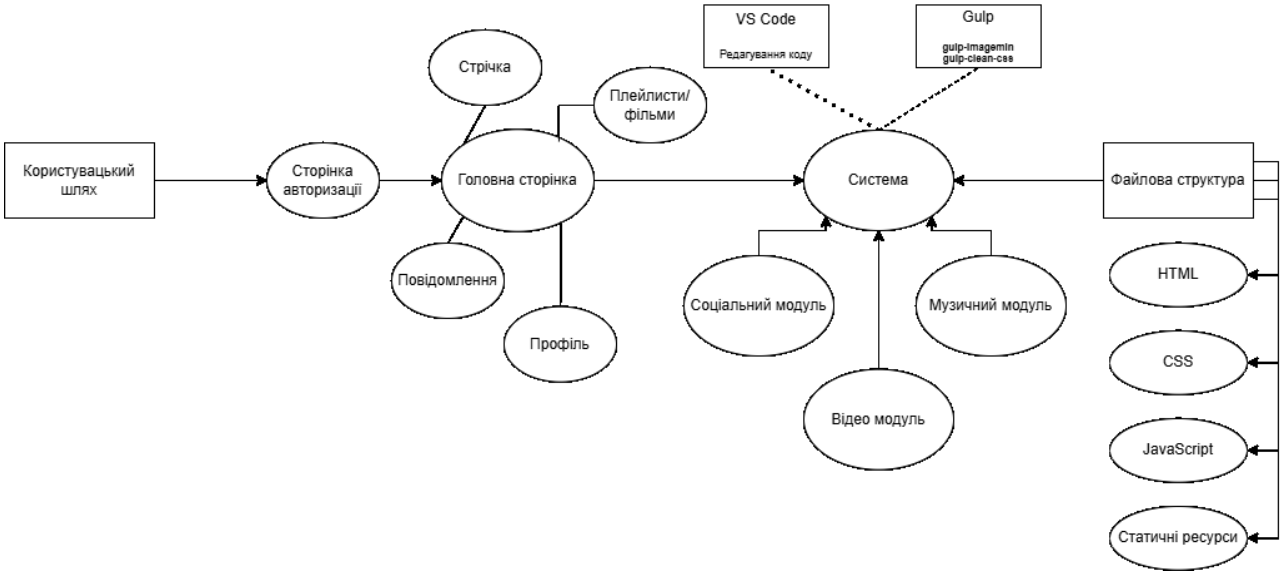

Рисунок 2.2 – Організація сторінок веб-сайту "Piggyback"

Схема відображає послідовність переходів: від сторінки авторизації до головної, де перемикач веде до музичного чи відеоконтенту, а також доступ до профілю й налаштувань через навігацію.

Спроектована структура забезпечує логічну організацію модулів і сторінок, інтеграцію інструментів Gulp і бібліотек, та адаптивний дизайн, створюючи основу для реалізації

2.4 Висновок до другого розділу

У другому розділі кваліфікаційної роботи проведено комплексне проєктування веб-сайту "Piggyback. Аналіз архітектури продемонстрував оптимальність вибору MPA для структурованої організації модулів, де кожен компонент (інтерфейс, обробка подій, управління даними) реалізовано з використанням HTML, CSS і JavaScript. Використання інструментів, таких як VS Code, Webpack і Gulp із відповідними плагінами, а також бібліотек (Bootstrap, jQuery, Video.js), забезпечило основу для адаптивного дизайну й інтерактивності.

Проєктування структури визначило логічну організацію сторінок, включаючи авторизацію, головний хаб із перемикачем між відео й музикою, а також профіль і налаштування. Модулі соціального контенту, музичного плейлиста й відеоплеєра інтегровано з використанням Slick Carousel, Chart.js і Video.js, а локальне зберігання даних через localStorage імітує персистентність. Адаптивність платформи досягнуто за допомогою Bootstrap Grid, Flexbox і медіа-запитів, що відповідає вимогам до зручності на різних пристроях.

Визначені архітектура, інструменти й структура підтверджують можливість створення інтегрованої платформи, що відповідає сучасним стандартам веб-розробки та потребам цільової аудиторії, визначеним у першому розділі.

РОЗДІЛ 3. ТЕСТУВАННЯ ВЕБ-САЙТУ "PIGGYBACK"

3.1 Реалізація інтерфейсу

Перед початком реалізації веб-сайту "Piggyback" проведемо підготовку до тестування, спрямовану на забезпечення його коректної роботи та зручності для користувачів. Тестування здійснюватиметься вручну з метою перевірки таких аспектів: авторизації, реєстрації, доступності всіх сторінок, перемикання між сторінками фільмів і музики, налаштування профілю, відтворення відео, відтворення музики та адаптації до різних пристроїв. Основна мета — виявлення можливих збоїв у функціонуванні модулів і відповідність вимогам, визначеним у першому розділі.

Для тестування підготовлено тестові дані, зокрема зразки облікових записів для авторизації та реєстрації, пости й плейлисти для перевірки сторінок, а також відеофайли й аудіофайли для відтворення контенту. Дані зберігатимуться в localStorage для імітації персистентності [19]. Критерії успішного проходження тестів включають відсутність помилок при вході/реєстрації, коректне завантаження всіх сторінок, плавне перемикання між модулями, функціональність налаштувань профілю, безперебійне відтворення відео й музики, а також правильне відображення на екранах із роздільною здатністю від 320 до 1920 пікселів. Перевірка сумісності проводитиметься в браузерах Google Chrome, Mozilla Firefox, Safari та Microsoft Edge.

Отже, підготовка до тестування веб-сайту "Piggyback" забезпечила необхідні умови для ручного оцінювання його функціоналу та якості, створюючи основу для подальшого аналізу.

3.2 Реалізація інтерактивних елементів

Веб-сайт "Piggyback" реалізовано на основі архітектури MPA із використанням HTML для базової структури сторінок, CSS для стилізації та

JavaScript для створення інтерактивних елементів. Локальне зберігання даних забезпечено через localStorage, що застосовується для авторизації, персоналізації контенту та налаштувань профілю. Основні інтерактивні компоненти розроблено в межах файлів index.html, style.css та script.js.

Реалізація включала наступні елементи:

– Створено форми з валідацією (перевірка формату email, мінімальної довжини пароля) за допомогою JavaScript, із інтерактивними повідомленнями про помилки.

– Розроблено динамічні каруселі з постерами та плейлистами, що реагують на натискання стрілок і перетягування мишею, із плавною анімацією через CSS і JS.

– Інтегровано інтерактивні плеєри для відео та аудіо з елементами керування (пуск/пауза, таймлайн, гучність) на основі HTML5-елементів і JavaScript.

– Реалізовано інтерактивну сторінку з полями для редагування імені, аватара, опису та дати народження, із підтримкою завантаження файлів і збереженням у localStorage.

– Додано інтерактивний перемикач між режимами "movie" і "music", що динамічно оновлює контент сторінки.

Інтерактивність оптимізована для адаптивності через медіа-запити в CSS, хоча остаточна перевірка адаптації проведена в наступному підрозділі. Тестові дані створено для користувача "admin" із паролем, а також використані некоректні вхідні дані для реєстрації задля забезпечення коректної роботи валідації.

3.3 Тестування та оптимізація

Процес оцінки функціональності авторизації включав введення імені користувача "admin" і пароля в відповідні поля форми (рис. 3.1), після чого

натиснуто кнопку "Login". Перевірено, чи система коректно розпізнає введені дані та забезпечує доступ до основного інтерфейсу.

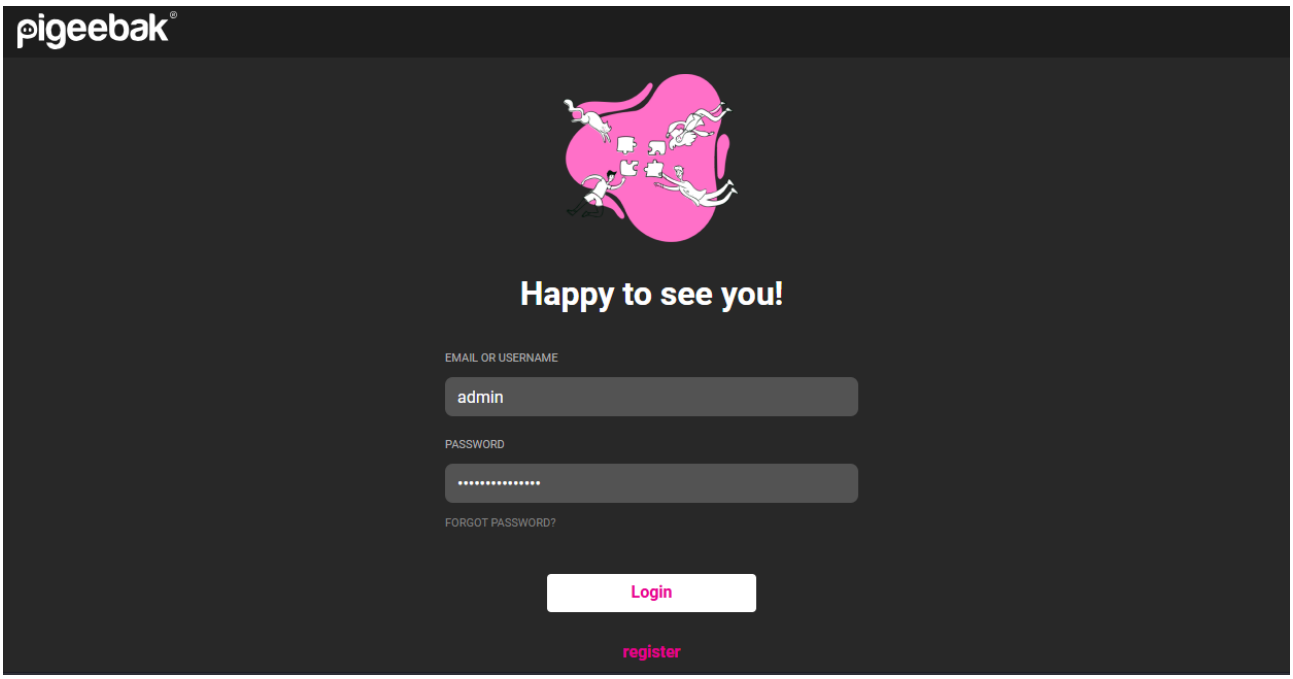

Рисунок 3.1 – Результат тестування авторизації

Результати тестування підтвердили коректну роботу авторизації. Форма завантажилася без затримок, введені дані оброблено успішно, а перехід до головної сторінки відбувся без помилок.

Наступним перевіримо реєстрацію. Повертаємося назад і натискаємо кнопку "register". Для реєстрації персонального акаунту, треба ввести свою електронну пошту, придумати назву акаунта та пароль до нього. Спробуємо ввести некоректні дані (рис. 3.2).

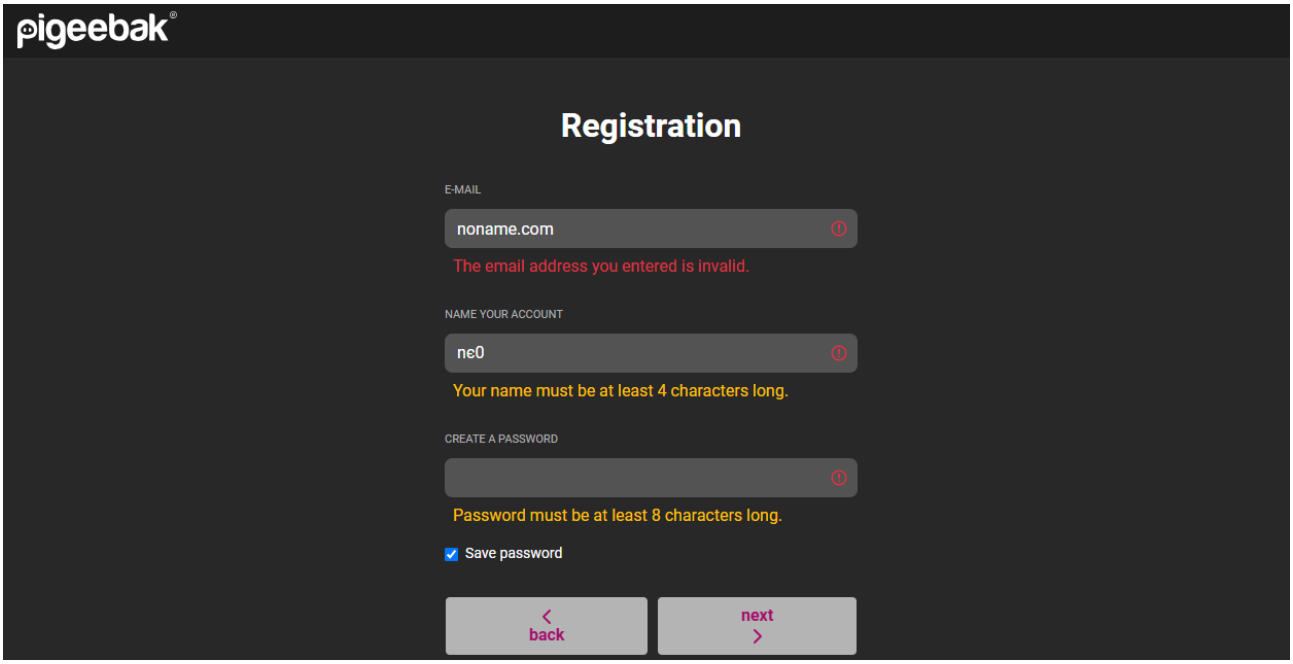

Рисунок 3.2 – Тестування реєстрації акаунту з некоректними даними

Введено "noname.com" як електронну пошту, "nє0" як назву акаунта і порожнє поле пароля, що супроводжується відповідними попередженнями.

Результати тестування показали коректну обробку вказаних даних. Форма відобразила чіткі повідомлення про помилки. Завантаження сторінки відбулося без затримок, а кнопка і "next" залишилися неактивними допоки не будуть внесені коректні дані, що дозволить продовжити або скасувати процес, що підтверджує стабільність функціоналу.

Перша сторінка, яку бачать користувачі, це сторінка з фільмами (рис.3.3). Спробуємо переключити її на музику. Для цього протестуємо функціонал перемикача, розміщеного у верхній частині інтерфейсу, який дозволяє користувачам обирати між режимами "movie" і "music".

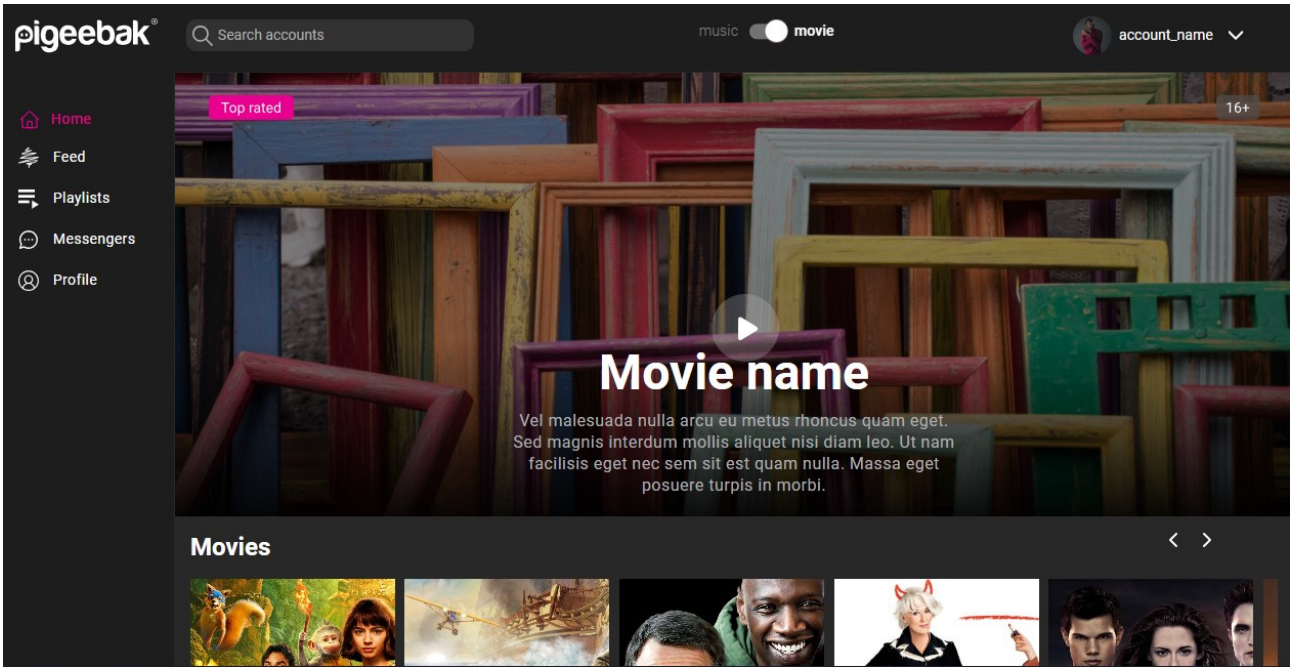

Рисунок 3.3 – Початкова сторінка користувача з фільмами

Перевіримо, чи перехід відбувається без збоїв і чи коректно відображається контент кожного модуля (рис. 3.4).

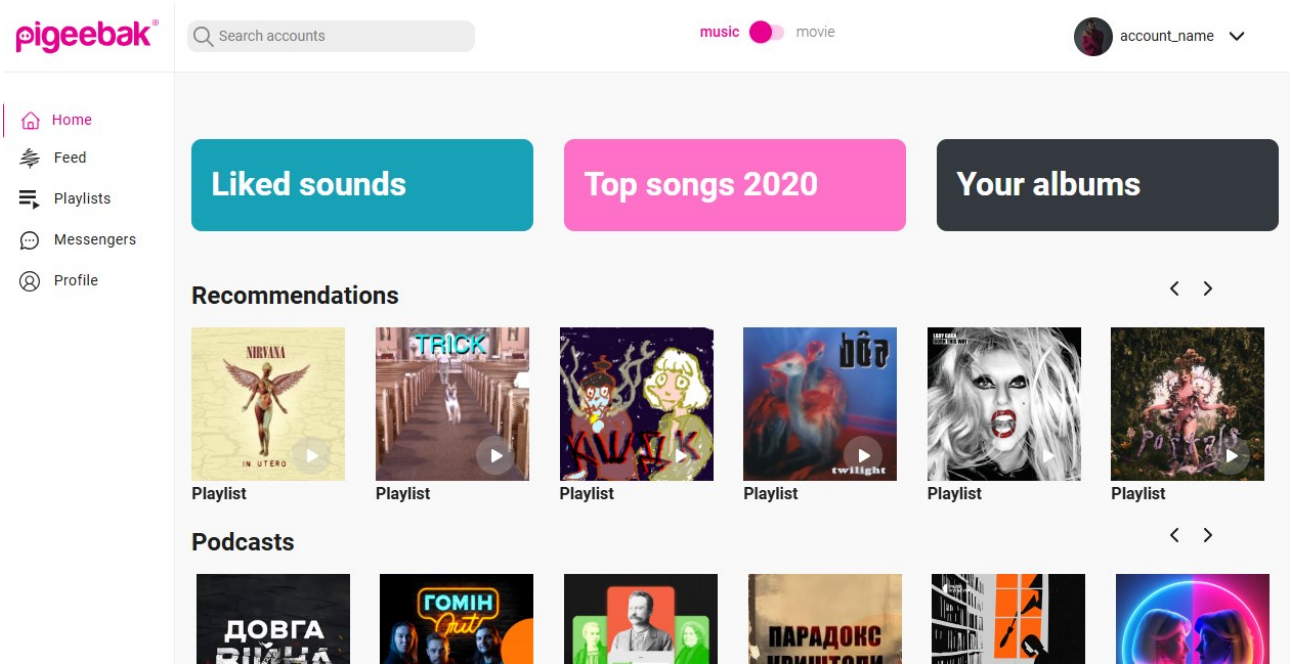

Рисунок 3.4 – Зміна початкової сторінки на музичну

Результати тестування підтвердили коректну роботу перемикача. Перехід із сторінки фільмів на сторінку музики відбувся без затримок, а контент (постери

фільмів і плейлисти) завантажився коректно. Навігаційне меню залишилося доступним, а інтерфейс зберіг адаптивність. Однак відзначено, що при швидкому перемиканні можливе короткочасне мигтіння інтерфейсу, що потребує подальшої оптимізації.

Для оцінки функціональності модулів фільмів і музики проведено тестування наявності системи каруселі. Ця система дозволяє користувачам переглядати контент, переміщаючись між елементами за допомогою стрілок або утриманням лівої кнопки миші. Перевірено, чи карусель коректно відображає постери фільмів і плейлисти музики, а також чи забезпечує плавне прокручування без збоїв.

Тестування включало натискання стрілок праворуч і ліворуч на сторінках фільмів і музики, а також утримання лівої кнопки миші для горизонтального переміщення контенту. У модулі фільмів карусель відображає постери з описами, таких як "Movie name", а в модулі музики — плейлисти, наприклад, "Nirvana – In Utero" і "Twilight". Перевірено, чи перехід між елементами відбувається без затримок і чи контент залишається чітким під час прокручування.

Результати тестування підтвердили коректну роботу каруселі в обох модулях. Натиснення стрілок і утримання миші забезпечило плавне переміщення контенту, а завантаження елементів відбувалося без помилок.

Наступним протестуємо відтворення відео та аудіо. Для перевірки функціональності відео спочатку натиснуто на один із фільмів, представлених на сторінці, щоб ініціювати відтворення. Перевірено, чи плеєр коректно запускається, чи відображається таймлайн, а також чи доступні елементи керування, такі як пауза, перемотування та регулювання гучності.

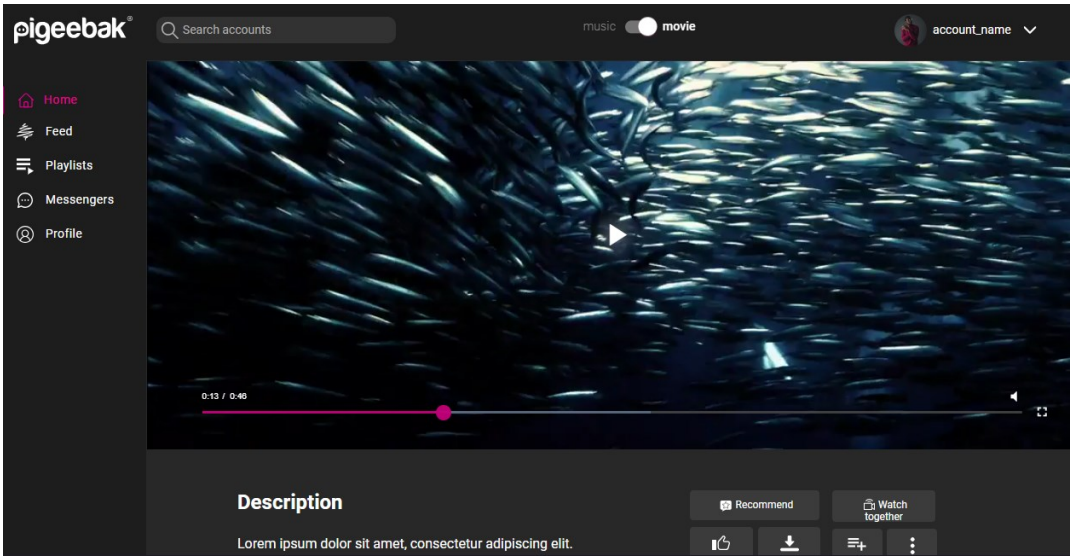

Рисунок 3.5 – Результат тестування відтворення відео

Результати тестування підтвердили коректну роботу відеоплеєра. Відтворення розпочалося без затримок після натискання, таймлайн коректно відображає прогрес, а елементи керування доступні для використання.

Наступним протестовано відтворення аудіо контенту на веб-сайті "Piggyback" після перемикання на модуль музики. За задумом, користувач має отримати доступ до власних плейлистів, таких як "Liked sounds" або "Top songs 2020", і мати змогу відтворювати аудіофайли. Перевірено, чи плеєр коректно запускається та чи відображається відповідний контент для звичайного користувача.

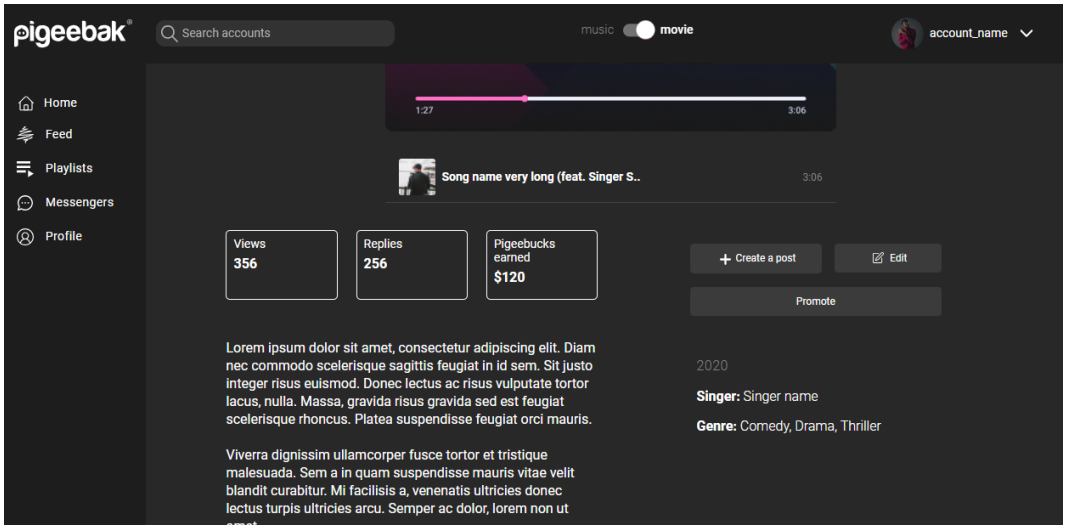

Рисунок 3.6 – Результат тестування відтворення аудіо

На зображенні видно інтерфейс модуля музики, але відтворення не відбулося, а відображено дані, призначені для адміністратора.

Результати тестування виявили неправильне функціонування аудіомодуля. При спробі відтворити аудіо відкривається контент, який належить адміністраторові (наприклад, статистика), а не звичайному користувачу, як передбачено. Крім того, відтворення аудіо виявилося неможливим: після натискання кнопки пуску плеєр не реагує. Ця проблема порушує вимоги до персоналізації і потребує подальшого аналізу в оптимізації.

Наступним протестуємо функціональність налаштувань. Перевірка включатиме доступ до сторінки, оцінку доступних опцій для редагування (ім'я користувача, аватар, опис та сповіщень). Також буде перевірено збереження змін, стабільність відображення після оновлення сторінки та адаптивність інтерфейсу на різних розмірах екранів. Спочатку відкриємо сторінку профілю та знайдемо пункт для доступу до налаштувань (рис. 3.7).

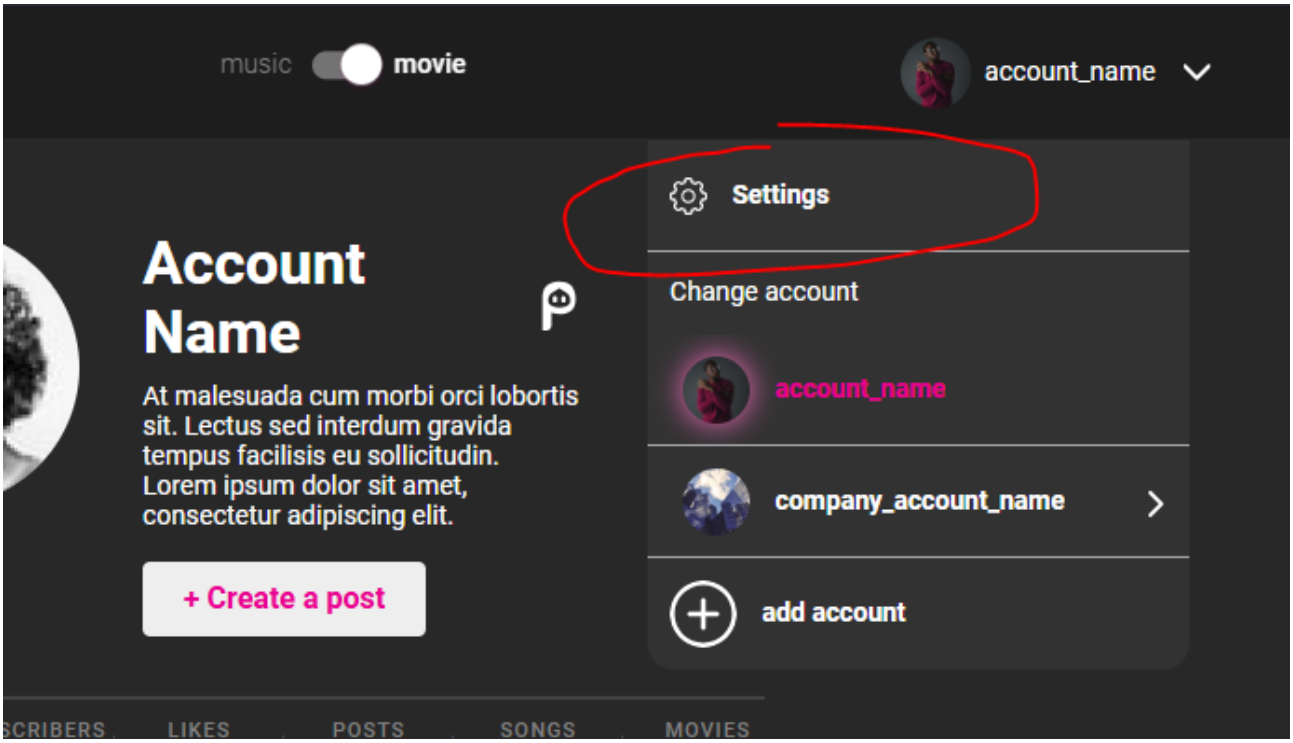

Рисунок 3.7 – Доступ до налаштувань профілю

Після натискання на "Settings" перейдемо до відповідного розділу для редагування параметрів (рис. 3.8). На зображенні відображено сторінку налаштувань, де доступні розділи "Notifications", "Edit account", "Confidentiality", "Ad/Partnership settings" і "Payment information". У розділі "Notifications" присутні опції "Allow Notifications" і "Doesn't Allow Notifications" із позначкою за замовчуванням, а також кнопка "Save" для збереження змін.

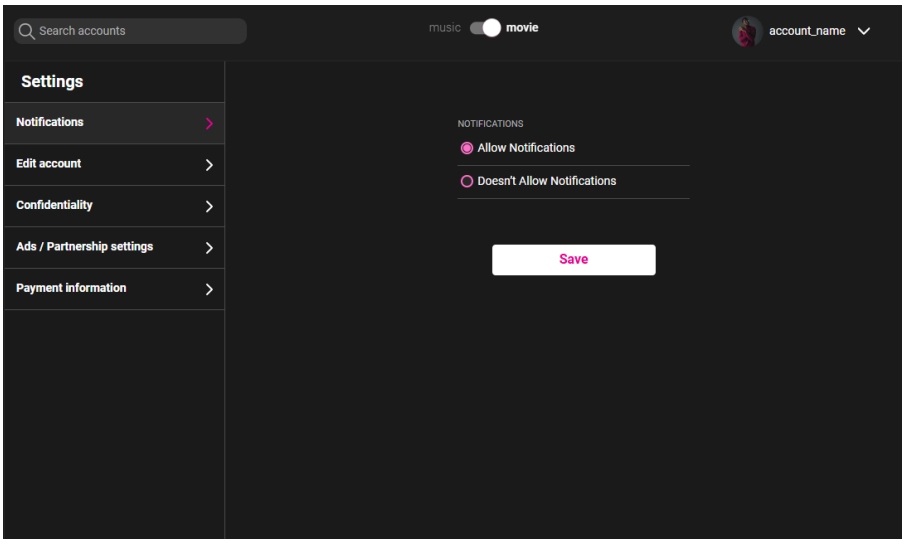

Рисунок 3.8 – Перегляд налаштувань профілю

Протестуємо зміну даних акаунту, а саме головної сторінки. Для цього відкрито розділ "Edit account" у налаштуваннях, де введено нові дані: ім’я змінено на "TestUser", опис на на "123456" та дату народження на 05.10.2006. Натиснуто кнопку "Save" для збереження змін (рис. 3.9).

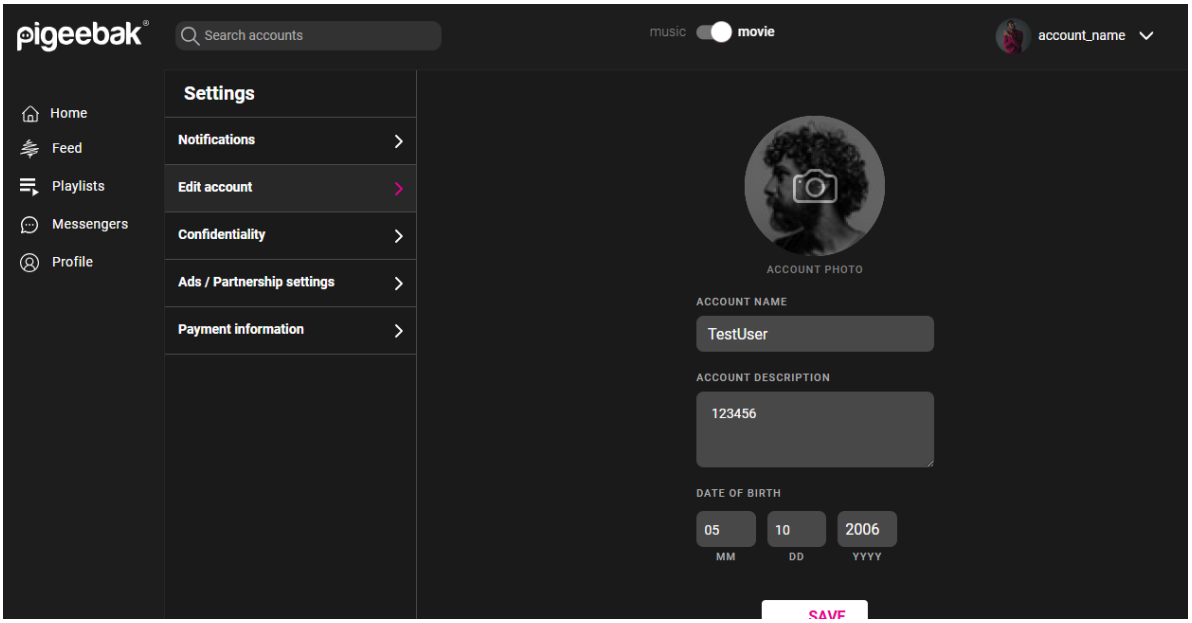

Рисунок 3.9 – Результат тестування редагування акаунту

Результати тестування показали, що зміни даних акаунту не зберігаються. Після натискання "Save" та оновлення сторінки поля повертаються до попередніх значень ("Account Name" і "Account Description" залишаються незмінними, дата народження скидається). Також при спробі завантаження нового фото аватару, відкривається файлове сховище і після вибору фото, не спостерігаються ніяких змін. Ця функція потребує подальшого вдосконалення і налаштувань.

Передостаннє, що потребує тестування, це адаптація до різних пристроїв і браузерів. Попередньо встановлено, що інтерфейс коректно відображається в браузерах, зазначених у підрозділі 3.1, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Для перевірки адаптації на мобільних пристроях і планшетах буде використано інструмент розробника (F12), де налаштуємо емуляцію обраних пристроїв, таких як iPhone, iPad та інші типові розміри екранів, щоб оцінити поведінку елементів інтерфейсу, включаючи кнопки, каруселі та плеєри. Спершу, спробуємо, як виглядатиме головна сторінка з фільмами на телефон iPhone 12 pro (рис. 3.10).

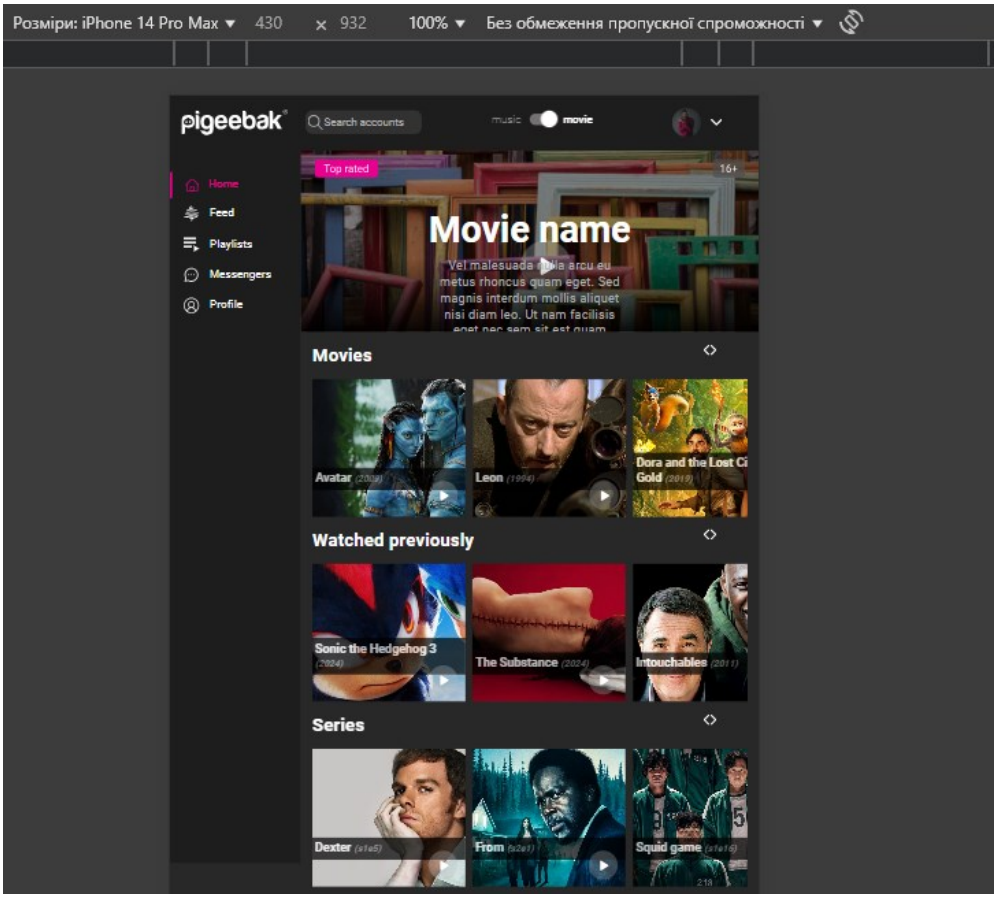

Рисунок 3.10 – Тестування адаптації на телефон iPhone 12 pro

Результати тестування підтвердили успішну адаптацію. Усі елементи інтерфейсу, включаючи постери, кнопки та меню, коректно відображаються на вертикальному екрані iPhone 14 Pro Max. Крім того, при зміні орієнтації на горизонтальну адаптація зберігається (рис. 3.11).

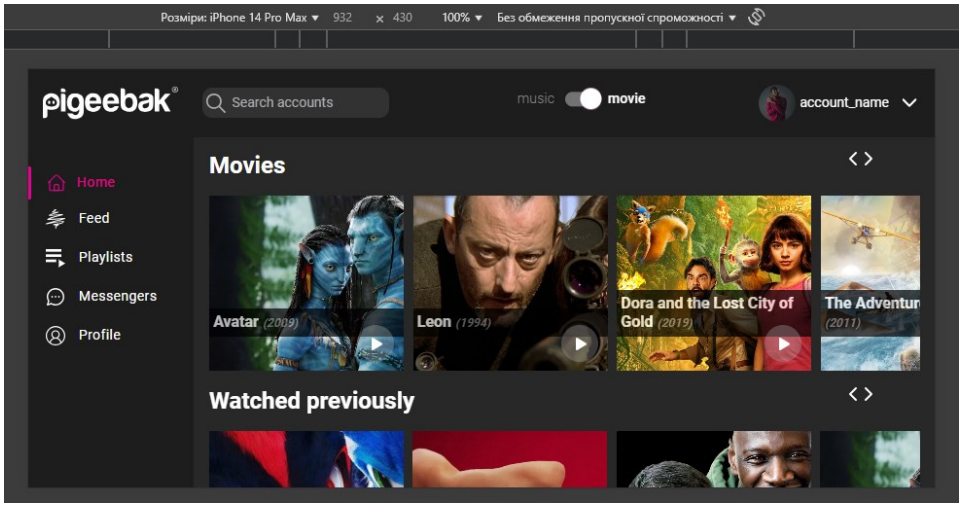

Рисунок 3.10 – iPhone 12 pro при горизонтальному положенні

На iPad Pro адаптація також працює, однак спостерігаються невеликі погрішності у вигляді зсуву опису сторінки "Movie name" (рис. 3.11), що може бути пов’язано з недостатньою обробкою медіа-запитів. Функціональність (перегляд контенту, натискання кнопок) зберігається.

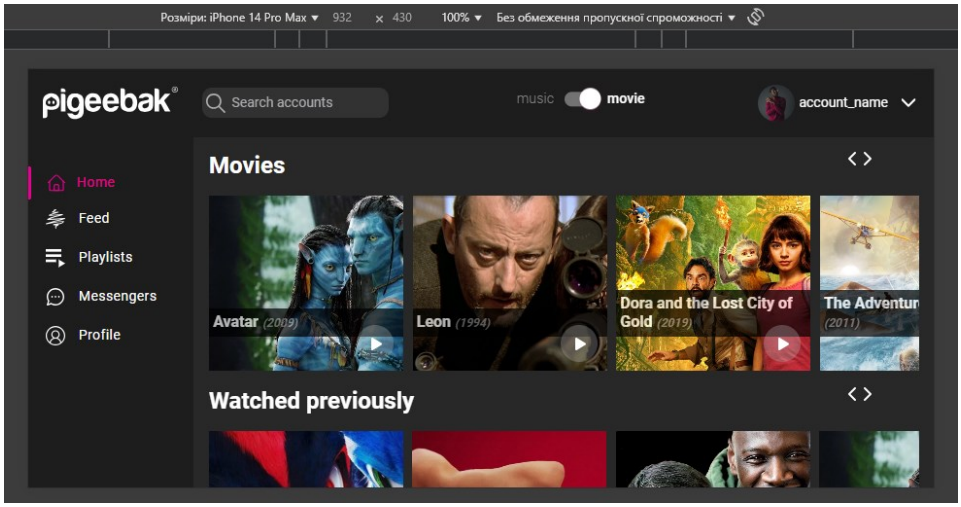

Рисунок 3.11 – Результат тестування адаптації головної сторінки з фільмами на iPad Pro.

У процесі тестування всіх попередніх функцій (авторизація, реєстрація, перемикання між модулями, каруселі, відтворення відео та аудіо, налаштування профілю, адаптація) усі сторінки веб-сайту "Piggyback" відкривалися коректно. Завантаження відбувалося без затримок, а навігація між розділами ("Home", "Feed", "Playlists", "Messengers", "Profile") була доступною в усіх тестуваних сценаріях. Це відповідає критеріям із підрозділу 3.1 щодо стабільності інтерфейсу.

3.4 Висновок до третього розділу

У третьому розділі проведено аналіз, реалізацію та тестування веб-сайту "Piggyback" із використанням архітектури MPA та технологій HTML, CSS і JavaScript. Визначено ключові критерії якості, такі як час відгуку, інтуїтивність інтерфейсу та адаптивність до пристроїв із роздільною здатністю від 320 до 1920

пікселів. Реалізовано інтерактивні елементи, включаючи авторизацію, каруселі, плеєри та налаштування профілю, із забезпеченням базової валідації та адаптивності через медіа-запити.

Тестування підтвердило коректну роботу авторизації, реєстрації, перемикання модулів, каруселей, відтворення відео та адаптації на iPhone 14 Pro Max і iPad Pro, а також доступність усіх сторінок. Однак виявлено проблеми: відтворення аудіо відображає дані адміністратора й не працює, зміни в налаштуваннях профілю не зберігаються, а на iPad Pro спостерігається зсув опису. Ці недоліки пов’язані з помилками в логіці доступу та обробці медіа-запитів, що потребує оптимізації.

Отже, реалізація веб-сайту досягла значного прогресу, але подальша робота зосередиться на виправленні виявлених проблем для забезпечення повної функціональності та зручності використання.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОХОРОНИ ПРАЦІ

4.1 Ергономіка інтерфейсу веб-сайту та її вплив на безпеку користувачів

Ергономіка, як синтез науки та мистецтва, спрямована на створення зручного й безпечного середовища для взаємодії людини з технічними системами. Вона відіграє ключову роль у забезпеченні безпеки користувачів, зокрема у запобіганні різного виду дискомфорту.

Ергономіка інтерфейсу веб-сайту відіграє ключову роль у безпеки користувачів, зокрема у запобіганні перенавантаження зору, стресів та фізичного дискомфорту. Ця особливість є важливою для веб-сайту, де користувачі активно взаємодіють із каруселями, відео та аудіо плеєрами, що може впливати на їхній комфорт і продуктивність. Завдяки ергономічному дизайну зменшується ризик втоми очей, покращується концентрація уваги та знижується ймовірність фізичних навантажень, таких як напруга шиї чи зап’ясть, що робить користування сайтом більш безпечним і приємним.

На сьогоднішній день ергономіка інтерфейсу стикається з низкою проблем, особливо в контексті інтерактивних елементів. Швидке прокручування каруселей може викликати дезорієнтацію, особливо на невеликих екранах, а тривале переглядання відео без пауз підвищує ризик перенавантаження зору. Контрастність тексту та зображень також потребує уваги, адже недостатній контраст може ускладнювати читання, а миготіння елементів під час перемикання модулів потенційно впливає на зір. Ці аспекти не завжди відповідають ергономічним стандартам, таким як ДСТУ EN ISO 9241-151:2022 [20], де визначено мінімальний розмір шрифту, рекомендації щодо контрастності та уникнення візуальних перешкод, що можуть викликати дискомфорт.

Аналіз інтерфейсів сучасних веб-сайтів показує, що хоча темна тема та інтуїтивне розміщення елементів є позитивними аспектами, залишаються можливості для покращення. Каруселі з контентом працюють плавно, але їхня

висока швидкість прокручування може викликати дезорієнтацію при частих взаємодіях. Відеоплеєри з таймлайном і кнопками керування зручні для короткочасного перегляду, однак відсутність автопаузи після тривалого використання підвищує ризик втоми користувачів. Тестування адаптації на різних пристроях, таких як смартфони та планшети, підтверджує коректність відображення в різних орієнтаціях, але зсуви текстових елементів на великих екранах потребують уваги, так як їхня наявність може викликати дискомфорт для читання користувачами, що приведе до перенапруження зору. Ці аспекти не завжди відповідають ергономічним стандартам, і для подолання перелічених проблем існує стандарт ДСТУ EN ISO 9241-151:2022, який визначає рекомендації щодо видимості тексту та уникнення візуальних перешкод, щоб покращити користування веб-сайтом для зручності користувачів. Для підвищення ергономіки доцільно збільшити контрастність тексту до 5:1 за допомогою CSS, додати автопаузу відеоплеєра після 15 хвилин бездіяльності, зменшити швидкість каруселей на 20% із попередженням про швидкий рух, забезпечити масштабування шрифтів до 14 pt на малих екранах через медіа-запити та провести юзабіліті-тест із 10 користувачами для оцінки втоми. Такі заходи сприятимуть усуненню недоліків, підвищенню безпеки праці та роблячи користування веб-ресурсами більш комфортним і безпечним.

Щоб забезпечити ще більшу безпеку та комфорт, важливо враховувати аспекти доступності для осіб із особливими потребами, зокрема тих, хто має порушення зору чи рухові обмеження. Інтеграція функцій, таких як підтримка екранних читачів через ARIA-атрибути та голосове керування за допомогою JavaScript, дозволить зменшити бар’єри у взаємодії з інтерфейсом. Крім того, використання тактильних сигналів для сенсорних екранів [21] і адаптація кольорової палітри для людей із дальтонізмом (наприклад, уникнення червоно-зелених комбінацій) сприятиме інклюзивності, а регулярне оновлення стандартів на основі зворотного зв’язку від користувачів допоможе підтримувати високу якість і безпеку платформи.

4.2 Організація робочого місця розробника під час створення веб-сайту

Розробка веб-сайтів, як складний і багатогранний процес, супроводжується низкою загальних труднощів, які впливають на здоров’я та продуктивність розробників. Створення інтерактивних елементів, адаптація до різних пристроїв і забезпечення стабільної роботи потребують тривалого перебування за комп’ютером, що часто призводить до фізичних навантажень, таких як напруга в шиї, зап’ястях чи очах через тривале фокусування на екрані. Неправильне освітлення, незручна позиція сидіння або відсутність регулярних перерв можуть викликати хронічні болі, тунельний синдром чи втомленість зору, що значно ускладнює роботу. Психологічний тиск додається через стислі терміни, необхідність вирішувати технічні проблеми, а також відповідальність за безпеку даних, що може призводити до стресу й вигорання. Ці фактори роблять організацію безпечного робочого місця не лише бажаним, але й необхідним аспектом для збереження здоров’я розробників.

У контексті розробки веб-сайту, правильне розташування обладнання, освітлення та ергономічна позиція допомагають уникнути професійних захворювань, таких як сколіоз чи хронічна втома очей, що є поширеними серед ІТ-фахівців. Сучасні стандарти, такі як ДСТУ 7299:2013 [22] та рекомендації Міжнародної організації праці, підкреслюють важливість адаптації робочого місця до фізіологічних потреб людини. На сьогоднішній день багато розробників стикаються з проблемами через недостатню увагу до ергономіки, що посилює фізичні та психологічні навантаження.

Аналіз робочого місця показує, що треба для забезпечення безпеки необхідно звернути увагу на кілька аспектів. Оптимальна висота столу повинна відповідати 70–80 см, а крісло — мати регульовану спинку та підлокітники, щоб кут ліктів становив 90°. Освітлення має бути розсіяним, щоб уникнути відблисків на моніторі, а екран розташований на відстані 50–70 см від очей. Рекомендується робити перерви кожні 40–50 хвилин по 5–10 хвилин, виконуючи вправи для

розтяжки шиї та зап’ясть. Для захисту зору варто використовувати програмне забезпечення з функцією "нічний режим" і фільтрами синього світла, а також періодично перевіряти зір у спеціаліста. Такі заходи, узгоджені з ДСТУ 7299:2013, допоможуть мінімізувати ризики для здоров’я та підвищити ефективність роботи над проєктом.

Також, якщо є можливість суттєво покращити безпеку та зручність розробників, доцільно впроваджувати сучасні технології та інструменти, які сприяють ергономічному робочому процесу. Використання спеціалізованих клавіатур із роздільними блоками та ергономічних мишей зменшить напругу в зап’ястях, а програмне забезпечення для моніторингу часу роботи, наприклад, Time Out або Stretchly, нагадуватиме про необхідність перерв і простих вправ.

4.3 Висновок до четвертого розділу

У цьому розділі проаналізовано ключові аспекти безпеки життєдіяльності та охорони праці, пов’язані з веб-сайтом та веб-розробкою. Розглянуто ергономіку інтерфейсу, яка забезпечує комфорт і безпеку праці користувачів, зменшуючи ризик втоми очей і стресів завдяки темній темі та інтуїтивному дизайну. Проаналізовано організацію безпечного робочого місця розробника, де наголошено на важливості ергономіки для уникнення фізичних навантажень, таких як тунельний синдром, та психологічного стресу через стислі терміни. Неправильне розташування обладнання та відсутність перерв створюють ризики для здоров’я.

Запропоновані заходи, включаючи оптимізацію контрастності інтерфейсу, регулярні перерви та правильне розташування робочого місця, сприяють зменшенню професійних ризиків і підвищенню продуктивності. Таким чином, подальша інтеграція цих рекомендацій у практику веб-розробки та використання сайтів стане важливим кроком до створення здорового цифрового середовища, що відповідає сучасним вимогам безпеки та комфорту.

ВИСНОВКИ

Під час виконання та закінчення кваліфікаційна роботи по розробці веб-сайту "Piggyback" з використання HTML, CSS та JavaScript, було доведено, що інтеграція зручності та безпеки є ключем до успішного веб-рішення. Основними поставленими задачами були аналіз потреб користувачів, проєктування архітектури платформи, практична реалізація її функціоналу та забезпечення ергономіки користувачів та розробників, а виконаними задачами стали створення інтегрованої платформи, що об’єднує соціальні, музичні та відеофункції, її тестування та пропозиція заходів для підвищення безпеки. Використання сучасних технологій дозволило створити платформу, яка відповідає потребам молодої аудиторії, підтверджуючи її актуальність і відкриваючи перспективи для комерційного розвитку.

Перший розділ мав на меті дослідити проблеми розрізненості цифрових платформ і визначити ключові потреби користувачів у доступі до соціальних, музичних та відеофункцій, сформулювати вимоги до системи, ідентифікувати акторів і сценарії взаємодії, а також провести порівняння з аналогами. Ці завдання виконано шляхом ґрунтовного аналізу, що створило основу для подальшої роботи, визначивши цільову аудиторію та обґрунтувавши унікальність "Piggyback".

Другий розділ ставив за мету детально опрацювати архітектуру та структуру платформи, обґрунтувати вибір технологій і інструментів для адаптивного дизайну та реалізувати логічну організацію сторінок із інтерактивними елементами. Завдання виконано через розробку модульної структури з використанням Gulp, Bootstrap та JavaScript, забезпечивши технічну основу для всіх модулів платформи.

Третій розділ охопив практичну реалізацію та оцінку роботи платформи, перевіривши функціональність авторизації, плеєрів і адаптації. В результаті тестування веб-сайту, було виявлено певні недоліки, такі як проблеми з аудіо та налаштуваннями, що вказало на напрями для подальшого вдосконалення.

Четвертий розділ про безпеку був спрямований на аналіз ергономіки інтерфейсу та робочого місця розробників, оцінку впливу на комфорт і здоров’я, а також розробку заходів для зменшення втоми та профілактики захворювань. Завдання виконано через детальний розгляд стандартів і пропозицій, таких як автопауза відеоплеєра та тренінги з ергономіки, сприяючи створенню безпечного середовища.

У підсумку, проведена робота поєднує теоретичний аналіз, практичну реалізацію та пропонує аспекти для безпеки, закладаючи міцну основу для розвитку створеного веб-сайту та загальної роботи в напрямку веб-розробки.

ПЕРЕЛІК ДЖЕРЕЛ

1 Популярність соцмереж і можливості для брендів. Як людство взаємодіє з цифровими технологіями — звіт Digital 2024. [Електронний ресурс]. URL: https://mediamaker.me/yak-lyudstvo-vzayemodiye-z-czyfrovymy-tehnologiyamy-zvit-digital-2024-8566/ (дата звернення: 26.05.2025).

2 Євген Паталяк. Найновіші технології веб-розробки 2025: як створюються сучасні сайти. [Електронний ресурс]. URL: https://wezom.com.ua/ua/blog/naynovishi-tehnologiyi-veb-rozrobki-2025-yak-stvoryuyutsya-suchasni-sayti (дата звернення: 26.05.2025).

3 Людмила Зінченко. Огляд та ключові тези вебінару Non-Functional Requirements: посібник для архітекторів-початківців. [Електронний ресурс]. URL: https://careers.epam.ua/blog/non-functional-requirements-a-guide-for-beginner-architects-part-one-webinar-key-take-aways (дата звернення: 26.05.2025).

4 Object Management Group. UML 2.5.1 Specification. C. 639-643. [Електронний ресурс]. URL: https://www.omg.org/spec/UML/2.5.1/PDF (дата звернення: 27.05.2025).

5 Development of an algorithm for identification of damage types on the surface of sheet metal Palianytsia, Y., Lytvynenko, I., Menou, A., Shymchuk, G., Dubchak, A. CEUR Workshop Proceedings, 2024, 3742, pp. 84–96.

6 Олександр Прус. Що таке фронтенд розробка: складові, етапи та технології. [Електронний ресурс]. URL: https://wezom.com.ua/ua/blog/chto-takoe-front-end-razrabotka (дата звернення: 27.05.2025).

7 Володимир Шайтан. Що таке бібліотека в програмуванні. [Електронний ресурс]. URL: https://wezom.com.ua/ua/blog/chto-takoe-front-end-razrabotka (дата звернення: 27.05.2025).

8 Sanity.io. What is Multi-Page Application? Multi-Page Application definition. [Електронний ресурс]. URL: https://www.sanity.io/glossary/multipage-application (дата звернення: 28.05.2025).

9 Automated algorithm for determining surface's oil capacity based on the analysis of the Abbot-Firestone diagram's parameters. Iaroslav Lytvynenko, Volodymyr Dzyura, Pavlo Maruschak, CEUR Workshop Proceedings, 2024, 3896, pp. 74–79.

10 Official Gulp Documentation. Quick Start. [Електронний ресурс]. URL: https://gulpjs.com/docs/en/getting-started/quick-start (дата звернення: 28.05.2025).

11 Boryslava Omelchenko. Single-Page Applications (SPA) vs Multi-Page Web Applications (MPA) — Which is Better? [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (дата звернення: 29.05.2025).

12 Microsoft. Visual Studio Code Documentation. [Електронний ресурс]. URL: https://code.visualstudio.com/docs (дата звернення: 29.05.2025).

13 Gulp-clean-css Official Documentation. [Електронний ресурс]. URL: https://www.npmjs.com/package/gulp-clean-css (дата звернення: 29.05.2025).

14 Bootstrap Get started with Bootstrap. [Електронний ресурс]. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 29.05.2025).

15 Methodology of the Formation of Sports Matches Statistical Information Using Neural Networks Sorokivska, O., Lytvynenko, I., Sorokivskyi, O., Kozbur, H., Strutynska, I. CEUR Workshop Proceedings, 2023, 3628, pp. 389–403.

16 Official jQuery Documentation. jQuery API. [Електронний ресурс]. URL: https://api.jquery.com (дата звернення: 29.05.2025).

17 Video.js Guides. [Електронний ресурс]. URL: https://videojs.com/guides/embeds/ (дата звернення: 01.05.2025).

18 Chart.js. [Електронний ресурс]. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 01.05.2025).

19 The Method of Computer Modeling of Heart Rhythm based on the Vector of Stationary and Stationary-related Random Sequences Onyskiv, P., Lytvynenko, I., Oleksandr, V., Shymchuk, G., Hotovych, V. CEUR Workshop, 2023, 3468, pp. 223–232.

20 ДСТУ EN ISO 9241-151:2022 Ергономіка взаємодії людина-система. Частина 151. Настанови щодо інтерфейсів користувача World Wide Web (EN ISO 9241-151:2008, IDT; ISO 9241-151:2008, IDT). [Електронний ресурс]. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=112548 (дата звернення: 02.06.2025).

21 Computer modeling of cardiac rhythm based on vector of stationary random sequences. Serhii Lupenko, Iaroslav Lytvynenko, Petro Onyskiv, Anatolii Lupenko, Oleksandr Volianyk, Olena Tsitsiura // Scientific Journal of TNTU. Tern.: TNTU, 2023. Vol 108. No 4. P. 131–143.

22 ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки. [Електронний ресурс]. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=57364 (дата звернення: 02.06.2025).

23 Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.

24 Lupenko, S., Lytvynenko, I., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, December). Approach to gas consumption process forecasting on the basis of a mathematical model in the form of a random cyclic process. In Proceedings of the International Conference „Advanced applied energy and information technologies 2021", 2021 (pp. 213-219). TNTU, Zhytomyr «Publishing house „Book-Druk "» LLC.

25 Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни Комп'ютерна графіка для студентів освітнього рівня «бакалавр» спеціальності 125 «Кібербезпека».

26 Shymchuk, G., Lytvynenko, I., Hromyak, R., Lytvynenko, S., & Hotovych, V. (2023). Gas Consumption Forecasting Using Machine Learning Methods and Taking Into Account Climatic Indicators. In CITI (pp. 156-163).

27 Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability

Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

28 ШИМЧУК, Г., ШЕВЧЕНКО, Н., ШВИРЛО, К., & ГАРМАТЮК, Н. (2025). СИСТЕМА ВІДНОВЛЕННЯ ДАНИХ У БЕЗДРОТОВИХ СЕНСОРНИХ МЕРЕЖАХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ. Herald of Khmelnytskyi National University. Technical sciences, 353(3.2), 246-250.

29 Nazarevych, O., Leshchyshyn, Y., Lupenko, S., Hotovych, V., Shymchuk, G., & Shabliy, N. (2020, September). Method of Gas Consumption Change-point Detection Based on Seasonally Multicomponent Model. In 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 152-155). IEEE.

30 Palianytsia, Y., Lytvynenko, I., Menoub, A., Shymchuk, H., & Dubchak, A. (2024). Development of an algorithm for identification of damage types on the surface of sheet metal.

ДОДАТКИ

Додаток А

Лістинг 1 – Налаштування збірника Gulp (gulpfile.js)

```
let preprocessor = 'sass', // Preprocessor 'sass' also work with the
Scss syntax in blocks/ folder.
              fileswatch   = 'html,htm,txt,json,md,woff2' // List of
files extensions for watching & hard reload

const { src, dest, parallel, series, watch } = require('gulp')
const browserSync  = require('browser-sync').create()
const bssi         = require('browsersync-ssi')
const ssi          = require('ssi')
const webpack      = require('webpack-stream')
const sass         = require('gulp-sass')
const sassglob     = require('gulp-sass-glob')
const less         = require('gulp-less')
const lessglob     = require('gulp-less-glob')
const styl         = require('gulp-stylus')
const stylglob     = require("gulp-empty")
const cleancss     = require('gulp-clean-css')
const autoprefixer = require('gulp-autoprefixer')
const rename       = require('gulp-rename')
const imagemin     = require('gulp-imagemin')
const newer        = require('gulp-newer')
const rsync        = require('gulp-rsync')
const del          = require('del')
const sourcemaps = require('gulp-sourcemaps');


function browsersync() {
       browserSync.init({
              server: {
                     baseDir: 'app/',
                     middleware:  bssi({  baseDir:  'app/',  ext:
'.html' })
              },
              tunnel:  'piggebak',  //  Attempt  to  use  the  URL
https://yousutename.loca.lt
              notify: false,
              online: true
       })
}

function scripts() {
       return src(['app/js/*.js', '!app/js/*.min.js'])
              .pipe(webpack({
                     mode: 'production',
                     module: {
                            rules: [
                                   {
```

```
                                        test: /\.(js)$/,
                                        exclude:
/(node_modules)/,
                                        loader: 'babel-loader',
                                        query: {
                                               presets:
['@babel/env'],
                                               plugins:   ['babel-
plugin-root-import']
                                        }
                                 }
                          ]
                   }
             })).on('error', function handleError() {
                    this.emit('end')
             })
             .pipe(rename('app.min.js'))
             .pipe(dest('app/js'))
             .pipe(browserSync.stream())
}

function styles() {
       return                src([`app/styles/${preprocessor}/*.*`,
`!app/styles/${preprocessor}/_*.*`])
             .pipe(sourcemaps.init())
             .pipe(eval(`${preprocessor}glob`)())
             .pipe(eval(preprocessor)())
             .pipe(autoprefixer({ overrideBrowserslist: ['last 10
versions'], grid: true }))
             .pipe(cleancss({ level: { 1: { specialComments: 0 }
},/* format: 'beautify' */ }))
             .pipe(rename({ suffix: ".min" }))
             .pipe(sourcemaps.write('../maps'))
             .pipe(dest('app/css'))
             .pipe(browserSync.stream())
}

function stylesBild() {
       return                src([`app/styles/${preprocessor}/*.*`,
`!app/styles/${preprocessor}/_*.*`])
             .pipe(eval(`${preprocessor}glob`)())
             .pipe(eval(preprocessor)())
             .pipe(autoprefixer({ overrideBrowserslist: ['last 10
versions'], grid: true }))
             .pipe(cleancss({ level: { 1: { specialComments: 0 }
},/* format: 'beautify' */ }))
             .pipe(rename({ suffix: ".min" }))
             .pipe(dest('app/css'))
             .pipe(browserSync.stream())
}

function images() {
       return src(['app/images/src/**/*'])
```

```
                .pipe(newer('app/images/dist'))
                .pipe(imagemin())
                .pipe(dest('app/images/dist'))
                .pipe(browserSync.stream())
}

function buildcopy() {
        return src([
                '{app/js,app/css}/*.min.*',
                'app/images/**/*.*',
                '!app/images/src/**/*',
                'app/fonts/**/*'
        ], { base: 'app/' })
        .pipe(dest('dist'))
}

async function buildhtml() {
        let includes = new ssi('app/', 'dist/', '/**/*.html')
        includes.compile()
        del('dist/parts', { force: true })
}

function cleandist() {
        return del('dist/**/*', { force: true })
}

function deploy() {
        return src('dist/')
                .pipe(rsync({
                        root: 'dist/',
                        hostname: 'piggebak.com',
                        destination: 'dist/',
                        include:  [/*  '*.htaccess'  */],  //  Included
files to deploy,
                        exclude: [ '**/Thumbs.db', '**/*.DS_Store' ],
                        recursive: true,
                        archive: true,
                        silent: false,
                        compress: true
                }))
}

function startwatch() {
        watch(`app/styles/${preprocessor}/**/*`, { usePolling: true
}, styles)
        watch(['app/js/**/*.js',      '!app/js/**/*.min.js'],      {
usePolling: true }, scripts)
        watch('app/images/src/**/*.{jpg,jpeg,png,webp,svg,gif}',   {
usePolling: true }, images)
        watch(`app/**/*.{${fileswatch}}`,    {    usePolling:    true
}).on('change', browserSync.reload)
}
```

```
exports.scripts = scripts
exports.styles  = styles
exports.images  = images
exports.deploy  = deploy
exports.assets  = series(scripts, styles, images)
exports.build    = series(cleandist, scripts, stylesBild, images,
buildcopy, buildhtml)
exports.default     =     series(scripts,     styles,     images,
parallel(browsersync, startwatch))
```

Лістинг 2 – Код сторінки авторизації (index.html)

```
<!DOCTYPE html>
<html>

<head>

       <meta charset="utf-8">
       <title></title>
       <meta  name="description"  content="Startup  HTML  template
OptimizedHTML 5">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1, shrink-to-fit=no">
       <link rel="icon" href="images/favicon.png">
       <!--                <meta                property="og:image"
content="images/dist/preview.jpg"> -->
       <link rel="stylesheet" href="css/main.min.css">
       <link  href="https://vjs.zencdn.net/7.10.2/video-js.min.css"
rel="stylesheet" />
</head>

<body id="body" theme="dark">

<div class="container-fluid login-container">
    <div class="row">
        <div class="col-12">
    <header id="login-head">
        <img                                        class="login-svg"
src="images/dist/IconCollectionPigeebak/pigeebak logo.svg" alt="">
    </header>
</div>
    </div>
    <div class="row">
        <div class="col-4 mx-auto">
    <main id="login-body">
        <img class="login-main-img" src="images/dist/login-img.png"
alt="">
        <h3 class="login-hellow">Happy to see you!</h3>
        <form action="">
            <div class="form-group">
              <label class="login-label" for="input-email">EMAIL OR
USERNAME</label>
```

```
              <input type="email" class="form-control login-input"
name="" id="input-email" value="admin"  placeholder="">
              <div    id="emailValidationFeedback"    class="d-none
email-invalid-feedback p-2 text-danger">
                The email address you entered is invalid. Please
check and try again.
              </div>
            </div>
            <div class="form-group">
              <label        class="login-label"        for="input-
password">PASSWORD</label>
              <input  type="password"  class="form-control  login-
input"  name=""  id="input-password"  value="adminadminadmin"  aria-
describedby="passwordhelpId" placeholder="">
              <div  id="passwordValidationFeedback"  class="d-none
password-invalid-feedback p-2 text-danger">
                The password you entered is incorrect. Please try
again.
              </div>
              <small  id="passwordhelpId"  class="form-text  text-
muted   login-help"><a   href="forgot-password-page-1.html">forgot
password?</a></small>
            </div>
          </form>
          <a   class="btn"   onclick="validate()"   href="#"><button
type="submit"  class="btn login-submit">Login</button></a>
          <button  class="btn  login-register"><a  href="register-
step-1-page.html">register</a></button>
    </main>
</div>
    </div>
</div>

<script>
    function validate() {
        const   passwordInput   =   document.querySelector('#input-
password');
        const emailInput = document.querySelector('#input-email');

        const                 emailInvalidFeedback                =
document.querySelector('#emailValidationFeedback');
        const               passwordInvalidFeedback               =
document.querySelector('#passwordValidationFeedback');

        const password = passwordInput.value;
        const email = emailInput.value;

        if (password === '') {
            passwordInput.classList.remove('is-valid');
            passwordInput.classList.add('is-invalid');
            passwordInvalidFeedback.classList.add('d-block');
            passwordInvalidFeedback.classList.remove('d-none');
        } else {
```

```
            passwordInput.classList.add('is-valid');
            passwordInput.classList.remove('is-invalid');
            passwordInvalidFeedback.classList.add('d-none');
            passwordInvalidFeedback.classList.remove('d-block');
        }

        if (email === '') {
            emailInput.classList.remove('is-valid');
            emailInput.classList.add('is-invalid');
            emailInvalidFeedback.classList.add('d-block');
            emailInvalidFeedback.classList.remove('d-none');
        } else {
            emailInput.classList.add('is-valid');
            emailInput.classList.remove('is-invalid');
            emailInvalidFeedback.classList.add('d-none');
            emailInvalidFeedback.classList.remove('d-block');
        }

        if     (emailInput.classList.contains('is-invalid')     ||
passwordInput.classList.contains('is-invalid')) {
            return
        }
        window.location.href = 'home-page.html'
    }
</script>
       <script src="js/app.min.js"></script>
       <script
src="https://vjs.zencdn.net/7.10.2/video.min.js"></script>
</body>
</html>
```

Лістинг 3 – Код сторінки реєстрації акаунту (register-step-b-2-page.html)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
    <meta   name="description"   content="Startup   HTML   template
OptimizedHTML 5" />
    <meta
      name="viewport"
      content="width=device-width,   initial-scale=1,   shrink-to-
fit=no"
    />
    <link rel="icon" href="images/favicon.png" />
    <!--                  <meta                  property="og:image"
content="images/dist/preview.jpg"> -->
    <link rel="stylesheet" href="css/main.min.css" />
    <link
      href="https://vjs.zencdn.net/7.10.2/video-js.min.css"
      rel="stylesheet"
```

```
    />
  </head>

  <body id="body" theme="dark">
    <div class="container-fluid login-container">
      <div class="row">
        <div class="col-12">
          <header id="login-head">
            <img
              class="login-svg"
              src="images/dist/IconCollectionPigeebak/pigeebak
logo.svg"
              alt=""
            />
          </header>
        </div>
      </div>
      <div class="row">
        <div class="col-4 mx-auto">
          <main id="login-body">
            <h3 class="login-hellow mt-5">Registration</h3>
            <form action="">
              <div class="form-group">
                <label   class="login-label"   for="input-email">e-
mail</label>
                <input
                  type="text"
                  class="form-control login-input"
                  name=""
                  id="input-email"
                  oninput="validate('email')"
                  placeholder=""
                  required
                />
                <div
                  id="emailValidationFeedback"
                  class="d-none  email-invalid-feedback  p-2  text-
danger"
                >
                  The email address you entered is invalid.
                </div>
              </div>
              <div class="form-group">
                <label class="login-label" for="account-name"
                  >name your account</label
                >
                <input
                  type="text"
                  class="form-control login-input"
                  name=""
                  id="account-name"
                  oninput="validate('name')"
                  aria-describedby="helpId"
```

```
                placeholder=""
              />
              <div
                id="nameValidationFeedback"
                class="d-none  name-invalid-feedback  p-2  text-
warning"
              >
                Your name must be at least 4 characters long.
              </div>
            </div>
            <div class="form-group">
              <label class="login-label" for="input-password-new"
                >create a password</label
              >
              <input
                type="password"
                class="form-control login-input"
                name=""
                minlength="8"
                id="input-password-new"
                oninput="validate('password')"
                aria-describedby="passwordhelpId"
                placeholder=""
              />
              <div
                id="passwordValidationFeedback"
                class="d-none password-invalid-feedback p-2 text-
warning"
              >
                Password must be at least 8 characters long.
              </div>
              <div class="form-check mt-2">
                <input
                  type="checkbox"
                  class="form-check-input"
                  name=""
                  id="input-checkbox"
                  value="checkedValue"
                  checked
                />
                <label
                  for="input-checkbox"
                  class="form-check-label login-checkbox"
                >
                  Save password
                </label>
              </div>
            </div>
          </form>
          <div class="d-flex justify-content-center">
            <a href="register-step-1-page.html"
              ><button  type="submit"  class="btn  login-submit
disabled">
```

```
                  <img
                    class="sound-arrow-pink rotate180"
src="images/dist/IconCollectionPigeebak/Icon/arrow-pink.svg"
                    alt=""
                  />
                  back
                </button></a
              >
              <a href="register-step-4-page.html"
                ><button
                  type="submit"
                  disabled
                  id="nextButton"
                  class="btn login-submit"
                >
                  next
                  <img
                    class="sound-arrow-pink"
src="images/dist/IconCollectionPigeebak/Icon/arrow-pink.svg"
                    alt=""
                  /></button
              ></a>
            </div>
          </main>
        </div>
      </div>
    </div>
    <script>
      document
        .getElementById("input-email")
        .addEventListener("input", function (event) {
          const pattern = /^[a-zA-Z0-9@._-]*$/;
          if (!pattern.test(event.target.value)) {
            event.target.value = event.target.value.replace(
              /[^a-zA-Z0-9@._-]/g,
              ""
            );
          }
        });

      function validate(inputName) {
        const   passwordInput   =   document.querySelector("#input-
password-new");
        const emailInput = document.querySelector("#input-email");
        const nextButton = document.querySelector("#nextButton");
        const nameInput = document.querySelector("#account-name");

        const emailInvalidFeedback = document.querySelector(
          "#emailValidationFeedback"
        );
        const passwordInvalidFeedback = document.querySelector(
```

```
          "#passwordValidationFeedback"
        );
        const nameInvalidFeedback = document.querySelector(
          "#nameValidationFeedback"
        );
        const   emailPattern   =   /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-
]+\.[a-zA-Z]{2,}$/;

        const password = passwordInput.value;
        const email = emailInput.value;
        const name = nameInput.value;

        if (inputName === "password") {
          if (password === "" || password.length < 8) {
            passwordInput.classList.remove("is-valid");
            passwordInput.classList.add("is-invalid");
            passwordInvalidFeedback.classList.add("d-block");
            passwordInvalidFeedback.classList.remove("d-none");
          } else {
            passwordInput.classList.add("is-valid");
            passwordInput.classList.remove("is-invalid");
            passwordInvalidFeedback.classList.add("d-none");
            passwordInvalidFeedback.classList.remove("d-block");
          }
        }

        if (inputName === "name") {
          if (name === "" || name.length < 4) {
            nameInput.classList.remove("is-valid");
            nameInput.classList.add("is-invalid");
            nameInvalidFeedback.classList.add("d-block");
            nameInvalidFeedback.classList.remove("d-none");
          } else {
            nameInput.classList.add("is-valid");
            nameInput.classList.remove("is-invalid");
            nameInvalidFeedback.classList.add("d-none");
            nameInvalidFeedback.classList.remove("d-block");
          }
        }

        if (inputName === "email") {
          if (email === "" || !emailPattern.test(email)) {
            emailInput.classList.remove("is-valid");
            emailInput.classList.add("is-invalid");
            emailInvalidFeedback.classList.add("d-block");
            emailInvalidFeedback.classList.remove("d-none");
          } else {
            emailInput.classList.add("is-valid");
            emailInput.classList.remove("is-invalid");
            emailInvalidFeedback.classList.add("d-none");
            emailInvalidFeedback.classList.remove("d-block");
          }
        }
```

```
        if (
          emailInput.classList.contains("is-invalid") ||
          passwordInput.classList.contains("is-invalid") ||
          nameInput.classList.contains("is-invalid")
        ) {
          nextButton.disabled = true;
        } else {
          nextButton.disabled = false;
        }
      }
    </script>
    <script src="js/app.min.js"></script>
    <script
src="https://vjs.zencdn.net/7.10.2/video.min.js"></script>
  </body>
</html>
```