Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: Розробка вебзастосунку для пошуку та класифікації анімованих фільмів та новел

Виконав: студент IV курсу, групи ЗПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Гандзюк Віктор Сергійович
(прізвище та ініціали)
Керівник:д-р. філос., асист. каф. ПЗ Карась О.В.
(прізвище та ініціали)
Рецензент:, к.т.н., доц. каф. ОТ Черняк О. І.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ О.Н. Романюк
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Гандзюку Віктору Сергійовичу

1. Тема роботи – розробка вебзастосунку для пошуку та класифікації анімованих фільмів і новел.

Керівник роботи: Карась Олександр Володимирович, д-р. філос., асист. каф. ПЗ затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи 2 червня 2025

3. Вихідні дані до роботи: структура анімаційного контенту у вигляді масиву об'єктів JavaScript; параметри фільтрації: назва, жанр, рік, тип, рейтинг; технології реалізації – HTML5, CSS3, JavaScript; до інтерфейсу – адаптивність, інтерактивність, підтримка анімації; цільові пристрої – ПК та мобільні браузери; модулі – реєстрація, авторизація, пошук, класифікація, список бажаного, профіль користувача.

4. Зміст текстової частини: вступ; аналіз стану технологій пошуку та класифікації анімованих фільмів та новел; аналіз методів розв'язання задачі; порівняльний аналіз аналогів; постановка задач дослідження; аналіз вхідних даних системи; розробка моделі системи; розробка алгоритмів роботи системи; варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення; розробка модуля збереження в список обраного; розробка модуля для пошуку анімаційних фільмів; розробка інтерфейсу програмного застосунку; тестування системи; розробка інструкції користувача.

5. Перелік ілюстративної частини: титульний слайд, актуальність розробки, мета, об'єкт та предмет дослідження, задачі дослідження, новизна і практична цінність отриманих результатів, порівняльний аналіз аналогів, загальний

алгоритм роботи системи, алгоритм реєстрації користувача, діаграма алгоритму пошуку, головна сторінка вебзастосунку, профіль користувача, список бажаного, сторінка з інформацією про анімований фільм, апробація та публікація.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Карась О.В., д-р. філос., асист. каф. ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз підходу до пошуку та класифікації анімованих фільмів та новел	25.03.25 – 07.04.25	викон.
2	Розробка моделі та алгоритмів програмного продукту	07.04.25 – 20.04.25	викон.
3	Розробка програмних модулів вебсистеми пошуку та класифікації анімованих фільмів	20.04.25 – 03.05.25	викон.
4	Тестування програмного застосунку	03.05.25 – 16.05.25	викон.
5	Оформлення матеріалів до захисту БКР	16.05.25– 30.05.25	викон.

Студент _____ **В. С. Гандзюк**
(підпис) (ініціали та прізвище)

Керівник бакалаврської кваліфікаційної роботи _____ **О. В. Карась**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

УДК 004.4

Гандзюка В.С. Розробка вебзастосунку для пошуку та класифікації анімованих фільмів і новел : бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 51 с.

Бакалаврська кваліфікаційна робота містить 51 сторінку формату А4, на яких наведено 25 рисунків та 2 таблиці, а список використаних джерел налічує 24 найменування.

У бакалаврській кваліфікаційній роботі було розроблено програмні модулі вебзастосунку пошуку та класифікації анімаційних фільмів та новел. Проведено аналіз сучасних технологій пошуку та класифікації анімаційних фільмів та новел, а також порівняльний аналіз існуючих подібних систем. Обґрунтовано необхідність розробки власної розробки для підвищення точності та ефективності пошуку, а також усунення недоліків існуючих систем, зокрема, відсутності розширених можливостей фільтрації та категоризації.

Розроблено програмні модулі, зокрема модуль додавання анімаційних фільмів до списку обраного, модуль динамічного пошуку контенту за назвою, а також адаптивний та зручний графічний інтерфейс користувача.

На основі проведених у БКР досліджень було розроблено модель програмного застосунку та алгоритми класифікації.

Тестування розробленої вебзастосунку засвідчило її повну працездатність та відповідність заданим функціональним специфікаціям. Розроблені програмні модулі та вебзастосунок в цілому можуть бути застосовані для покращення взаємодії користувачів з великими обсягами анімаційного контенту.

Ключові слова: анімовані фільми, аніме, легкі новели, пошук, фільтрація, класифікація, JavaScript.

ABSTRACT

UDC 004.4

Handziuka V.S. Development of a web application for searching and classifying animated films and novels: bachelor's thesis in specialty 121 - Software Engineering, educational program - Software Engineering. Vinnytsia: VNTU, 2025. 51 c.

The bachelor's thesis contains 51 pages in A4 format, with 25 figures and 2 tables, and a list of 24 references.

In the bachelor's thesis, the author developed software modules for a web application for searching and classifying animated films and short stories. An analysis of modern technologies for searching and classifying animated films and short stories, as well as a comparative analysis of existing similar systems, was carried out. The necessity of developing an in-house development to improve the accuracy and efficiency of search, as well as to eliminate the shortcomings of existing systems, in particular, the lack of advanced filtering and categorization capabilities, is substantiated.

Software modules have been developed, including a module for adding animated films to the favorites list, a module for dynamic content search by title, and an adaptive and user-friendly graphical user interface.

Based on the research conducted in the GDR, a model of the software application and classification algorithms were developed.

Testing of the developed web application has shown its full operability and compliance with the specified functional specifications. The developed software modules and the web application as a whole can be used to improve user interaction with large volumes of animated content in streaming services, information portals, and thematic.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ ПІДХОДУ ДО ПОШУКУ ТА КЛАСИФІКАЦІЇ АНІМОВАНИХ ФІЛЬМІВ ТА НОВЕЛ 7
1.1 Аналіз стану технологій пошуку та класифікації анімованих фільмів та новел 7
1.2 Аналіз методів розв’язання задачі 10
1.3 Порівняльний аналіз аналогів 12
1.4 Постановка задач дослідження 16
1.5 Висновки 17
2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ 18
2.1 Аналіз вхідних даних системи 18
2.2 Розробка моделі системи 20
2.3 Розробка алгоритмів роботи системи 23
2.4 Висновки 27
3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ ПОШУКУ ТА КЛАСИФІКАЦІЇ АНІМОВАНИХ ФІЛЬМІВ 28
3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення 28
3.2 Розробка модуля збереження в список обраного 30
3.3 Розробка модуля для пошуку анімаційних фільмів 34
3.4. Розробка інтерфейсу програмного застосунку 36
3.5 Висновки 41
4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ 42
4.1 Тестування системи 42
4.2 Розробка інструкції користувача 48
4.3 Висновки 51
ВИСНОВКИ 52

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 54
ДОДАТКИ 57
Додаток А – Технічне завдання 58
Додаток Б – Протокол перевірки на плагіат 62
Додаток В – Лістинг програмного забезпечення 63
Додаток Г – Графічна частина 112

ВСТУП

Обґрунтування вибору теми дослідження.

Сучасний етап розвитку інформаційних технологій вимагає постійного вдосконалення методів пошуку та класифікації інформації, особливо у сфері анімаційного контенту. Серед важливих питань у цій галузі є точність та корисність отриманих результатів, яких вимагають користувачі.

Існуючі методи пошуку анімованих фільмів [1] обмежуються базовою фільтрацією тексту і не мають достатньої деталізації категоризації. У зв'язку з цим застосування сучасних вебтехнологій та методів аналізу контенту відкриває нові можливості для оптимізації процесу пошуку та категоризації.

Об'єктивність і достовірність рекомендацій, яких вимагають глядачі, зацікавлені в доступі до якісних анімаційних фільмів, визначається достовірністю зібраної інформації. Наприклад, алгоритми, що враховують близькість сюжетів та глядацькі рейтинги, можуть значно покращити роботу рекомендаційних систем.

Сучасні подібні вебсистеми мають певні недоліки, такі як нерозвиненість механізмів глибокого аналізу контенту та замала кількість критеріїв для класифікації. Крім того, вони здебільшого використовують платні моделі доступу для розширеного функціоналу. Тому актуальною є розробка власної вебсистеми пошуку та класифікації анімаційних фільмів на основі сучасних алгоритмів обробки даних.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою дослідження є підвищення точності та ефективності пошуку анімованих фільмів шляхом розробки вебсистеми з модулями реєстрації та авторизації, фільтрації контенту, користувацького інтерфейсу та збереження у список бажаного, що забезпечує

зручний доступ до впорядкованої та персоналізованої інформації. Відповідно до поставленої мети в бакалаврській кваліфікаційній роботі потрібно вирішити такі задачі:

– аналіз предметної галузі;

– розробка модуля для збереження в список бажаного;

– розробка графічного інтерфейсу [2], який буде зручним у використанні та забезпечить швидкий доступ до інформації;

– розробка модуля пошуку анімованих фільмів;

– тестування розроблених програмних продуктів.

Об'єкт дослідження – процес розробки застосунку для пошуку та класифікації анімаційних фільмів.

Предмет дослідження – методи та засоби розробки застосунку для пошуку та класифікації анімаційних фільмів.

Методи дослідження. У процесі дослідження застосовувалися: методи веброзробки для створення інтерфейсу користувача та функціональності; комп'ютерне моделювання для аналізу та перевірки отриманих результатів.

Наукова новизна отриманих результатів:

1. Подальшого розвитку набув метод класифікації анімованих фільмів та новел на основі їх жанрових особливостей та вікових обмежень, що дозволяє підвищити точність пошуку та персоналізації контенту для користувачів.

2. Подальшого розвитку набув метод динамічної генерації та відображення контенту вебзастосунку з використанням DOM-маніпуляцій, що забезпечує гнучкість інтерфейсу та ефективне оновлення інформації про анімовані фільми та новели.

Практичне значення отриманих результатів полягає використанні вебсистеми, яка забезпечить точний пошук і класифікацію анімаційних фільмів з використанням сучасних алгоритмів. Розроблені програмні модулі можуть бути використані у сфері мультимедійного контенту.

Особистий внесок здобувача. Усі наукові результати отримано здобувачем самостійно. У працях, опублікованих у співавторстві, студенту належать: [3] – розглянуто підхід до розробки вебзастосунку для ефективного пошуку та рекомендації анімованих фільмів і новел; [4] – розглянуто підхід до розробки та інтеграції системи сповіщень у реальному часі в вебзастосунок для пошуку та класифікації анімованих фільмів і новел.

Апробація матеріалів бакалаврської кваліфікаційної роботи. Основні положення БКР доповідалися та обговорювалися на Міжнародній і Всеукраїнській конференціях: Міжнародна науково-практична інтернетконференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (м. Вінниця, 2025), LIV Всеукраїнська науково-технічна конференція підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) (м. Вінниця, 2025).

Публікації. За темою бакалаврської роботи надруковано 2 праці у матеріалах міжнародній і всеукраїнській конференціях.

Структура та обсяг БКР. Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків, у які винесено технічне завдання, протокол перевірки на плагіат, лістинг програмного забезпечення й ілюстративний матеріал до захисту роботи

1 АНАЛІЗ ПІДХОДУ ДО ПОШУКУ ТА КЛАСИФІКАЦІЇ АНІМОВАНИХ ФІЛЬМІВ ТА НОВЕЛ

1.1 Аналіз стану технологій пошуку та класифікації анімованих фільмів та новел

Сучасне цифрове медіасередовище відзначається неймовірно великим рівнем контенту, і анімовані фільми, не є винятком з цього правила. Щороку індустрія випускає сотні нових телевізійних шоу, художніх фільмів, з приголомшливим розмаїттям жанрів, стилів, тематики та аудиторій, від дитячих пригод і шкільних романів до розкішних науково-фантастичних епопей, метафізичних драм та авангардного артхаусу. Величезна кількість і розмаїття створюють величезний виклик для користувачів, а отже, породжують проблему так званого «інформаційного перевантаження». Пересічному глядачеві стає дедалі важче орієнтуватися в цьому океані контенту, знаходити твори, що відповідають його індивідуальним смакам, і відокремлювати хороші продукти від поганих. Хороші системи пошуку, категоризації та рекомендацій є не просто бажаними, а необхідним мінімумом для успіху будь-якої платформи, що працює з аніме-контентом.

Раніше перші спроби індексації аніме були здійснені за допомогою досить примітивних методів. Нормою було ручне введення метаданих і категоризація за невеликою кількістю загальних жанрів. Пошук [5] зазвичай полягав у простому зіставленні слів, введених користувачем, з назвами. З цим виникало багато проблем. Точність пошуку була низькою через те, що система не могла розпізнавати синоніми, працювати з різними формами назв та ігнорувала контекст і семантику запиту. Пошук за іменем персонажа міг дати десятки нерелевантних результатів. Жорстка жанрова класифікація не дозволяла виразити широкий спектр піджанрів [6] і конкретних сюжетів, які є частиною аніме, наприклад, «ісекай», «меха», «махо-дзюцу» або «повсякденне життя». Нанесення тегів вручну було дорогим, трудомістким і суб'єктивним процесом,

який не встигав за розвитком індустрії. Відсутність будь-якої персоналізації означала, що користувацький досвід був ідентичним для всіх користувачів, незалежно від індивідуальних уподобань.

З розвитком інформаційних технологій, таких як вебтехнології, бази даних, алгоритми обробки даних, з'явилися набагато ефективніші рішення цих проблем. Сучасні системи пошуку та категоризації аніме використовують синтез різних підходів. Системи пошуку та фільтрації стали набагато досконалішими. Багатовимірний пошук, в якому користувачі можуть комбінувати фільтри на основі різноманітних метаданих. Ці метадані охоплюють не лише загальні жанри, а й специфічні піджанри, теги для конкретних тем, тропів чи налаштувань, рік випуску, студію, режисера, сценариста, композитора, акторів озвучення, віковий рейтинг, користувацький рейтинг, тип аніме, статус релізу і навіть деталі, що стосуються зв'язку між різними сезонами чи частинами франшизи. Однією з особливостей стала навігація, яка дозволяє користувачам поступово, крок за кроком, звужувати свій пошук. Обробка синонімів та альтернативних назв також має важливе значення для покращення пошуку, що зазвичай здійснюється за допомогою спеціальних словників або баз знань.

Системи рекомендацій [7] зросли в геометричній прогресії і стали невід'ємним компонентом сучасних платформ. Вони пройшли шлях від простих списків «найпопулярніших» до складних персоналізованих алгоритмів. Фільтрація на основі контенту співставляє атрибути аніме, яким користувачі надають перевагу, і рекомендує схожі. Спільна фільтрація, що охоплює набір методів, таких як фільтрація на основі користувачів, фільтрація на основі предметів і матрична факторизація, виявляє шаблони активності серед активності великої кількості користувачів і надає рекомендації на основі колективного інтелекту. Гібридні методи намагаються поєднати сильні сторони двох процесів, уникаючи слабких сторін, тобто проблеми «холодного старту», що описує труднощі у створенні рекомендацій для нових користувачів або об'єктів, а також небезпеки надмірної спеціалізації, широко відомої як

«бульбашка фільтрів». Сучасні системи також намагаються пропонувати «зрозумілі рекомендації», надаючи користувачеві пояснення причин рекомендації того чи іншого твору, тим самим підвищуючи довіру і прозорість.

Сторонні сховища знань та інтерфейси прикладного програмування, такі як MyAnimeList [8], AniList [9], відіграють центральну роль у забезпеченні цілісності даних. Багато платформ взаємодіють з цими джерелами для отримання структурованих метаданих, які, хоч і не завжди однакові, включають рейтинги, статус релізу та інші відповідні деталі.

Це дає змогу підтримувати персональні каталоги в актуальному стані з мінімальними зусиллями вручну. Водночас, повнота і точність таких даних залежить від активності спільноти користувачів, які їх створюють, що має як переваги, так і недоліки. Розвиток користувацьких інтерфейсів та користувацького досвіду заслуговує на значну увагу. Вебсайти перейшли від базових текстових списків до візуально насичених сіток з рекламними зображеннями, інтерактивними системами фільтрації, вбудованими трейлерами, а також системами оцінювання та коментування. Крім того, включення функцій, орієнтованих на спільноти, таких як форуми, клуби та списки друзів, підвищує залученість користувачів, одночасно генеруючи допоміжні дані для налаштування рекомендаційних систем.

Незважаючи на значний прогрес, який був досягнутий, деякі перешкоди все ще залишаються. Досягнення високого ступеня точності в процесах автоматичної класифікації та тегування, особливо для творів, які не відповідають традиційним жанрам або є експериментальними за своєю природою, залишається значним викликом. Вирішення проблеми «холодного старту» вимагає впровадження складних гібридних моделей. Підтримання цілісності та однорідності метаданих, особливо коли вони отримані з різних джерел або за допомогою методів краудсорсингу, вимагає постійної пильності. Зростає також потреба враховувати більш тонкі особливості, такі як попередження щодо контенту на делікатні теми.

Отже, сучасний стан технологій пошуку та класифікації анімаційних фільмів ілюструє помітний перехід від рудиментарних методів до впровадження складних, багатогранних систем, які включають ефективні алгоритми обробки даних, можливості машинного навчання, інтеграцію із зовнішніми ресурсами та акцент на індивідуальному користувацькому досвіді. Успішна реалізація нової системи в цій галузі вимагає хорошого розуміння цих технологій, їх функціональності та обмежень, а також здатності об'єднати їх в оперативний і придатний для використання продукт.

1.2 Аналіз методів розв’язання задачі

Розробка функціональних модулів програмного забезпечення для пошуку та системи категоризації, розробленої для анімаційних фільмів, вимагає комплексного підходу, що включає ефективне управління інформацією, планування систем фільтрації та сортування та створення персоналізованих рекомендаційних методів. Це включає надання зручних інструментів для навігації, фільтрації та відкриття нового вмісту. Комплексний підхід вимагає ретельного планування процесів управління інформацією, розробки адаптивних систем фільтрації та сортування та інтеграції методів персоналізації взаємодії з користувачем, зокрема за допомогою систем рекомендацій. При побудові такої системи можна застосувати кілька фундаментальних напрямків і методологій.

Одним із найважливіших компонентів є пошукова система. Його основна функція полягає в тому, щоб надати користувачам можливість швидко та легко знаходити певні назви анімованих фільмів на основі різних параметрів. Найпростіший варіант передбачає базовий текстовий пошук, аналіз збігів між запитом користувача та назвами та потенційно короткими описами аніме в базі даних. Тим не менш, цей підхід має значні обмеження, які перешкоджають його ефективності в цій області. По-перше, він не враховує варіації в назвах. Аніме часто мають офіційну назву японською мовою, локалізовану назву англійською мовою, локалізовану назву іншою мовою, а також загальні скорочення чи

акроніми. Базовий пошук без урахування синонімів може не знайти твір, навіть якщо користувач введе одну з його добре відомих альтернативних назв. По-друге, простий текстовий пошук не сприяє ефективній організації вмісту на основі конкретних критеріїв, таких як жанр, рік випуску, студія виробництва, режисер, рейтинг або тип аніме. Користувачі не можуть створювати складні запити, наприклад «знайти всі фантастичні аніме MAPPA, випущені після 2020 року з рейтингом вище 8,0». Це робить навігацію величезним каталогом надзвичайно неефективною та проблематичною. Більш просунуте рішення передбачає реалізацію багатогранного пошуку за допомогою фасетної фільтрації. Цей підхід передбачає індексування різних атрибутів кожного анімованих фільмів та надання користувачеві можливості комбінувати фільтри за цими атрибутами. Система динамічно оновлює результати після кожного застосування фільтра, дозволяючи користувачеві поступово уточнювати пошук до досягнення бажаного набору результатів. Реалізація такого пошуку вимагає ретельного проєктування структури даних і ефективних алгоритмів фільтрації.

Іншою важливою функціональністю, яка безпосередньо впливає на залучення користувачів, є система рекомендацій. Його роль полягає в тому, щоб запропонувати вміст, який потенційно може зацікавити користувача, на основі його попередньої діяльності або вподобань. Найпростіший підхід надає рекомендації на основі загальної популярності чи показників рейтингу. Користувачам просто показують найпопулярніші аніме або аніме з високим рейтингом. Недоліком цього методу є те, що він повністю ігнорує індивідуальні смаки користувача те, що популярне серед більшості, може зовсім не відповідати їхнім інтересам. Більш персоналізований підхід включає фільтрацію на основі вмісту. Це аналізує характеристики аніме, які користувач раніше оцінив позитивно і рекомендує інші аніме з подібними характеристиками. Хоча цей метод визнає особливі інтереси користувача, він може призвести до «міхура фільтра», коли система рекомендує лише дуже схожий вміст, обмежуючи відкриття нових, нетрадиційних жанрів або стилів. Хоча впровадження складної

системи рекомендацій виходить за рамки цієї роботи, розуміння цих методів є життєво важливим для визначення майбутніх напрямків розвитку.

Одним із елементів, який підвищує привабливість системи для шанувальників аніме, є включення в неї додаткових даних, зокрема щодо легких романів. Ці романи разом із мангою часто служать основою для улюблених аніме-серіалів. Надання користувачам таких подробиць, як існування вихідного матеріалу, залучених осіб , статус його завершення та стисле резюме, може помітно збільшити їхнє задоволення. Це дає змогу користувачам краще зрозуміти передісторію серіалу, бути в курсі продовження оповідання за межами неповної адаптації аніме або просто задовольнити свій інтерес до оригінального літературного твору.

Обраний підхід зосереджений на розгортанні функції пошуку, функції списку побажань, призначеної для полегшення майбутнього налаштування, і розширення вмісту шляхом включення пов’язаних творів, таких як легкі романи.

1.3 Порівняльний аналіз аналогів

Сучасні алгоритми дозволяють користувачеві швидко знаходити потрібний контент за безліччю критеріїв, таких як жанр, рейтинг, популярність і навіть конкретний режисер. При всьому цьому різні онлайн-платформи приносять свої унікальні рішення, які кардинально впливають на зручність використання і функціональність.

Створення такого програмного забезпечення для пошуку та класифікації анімаційних фільмів може запропонувати значне поліпшення навігації, сильний користувальницький досвід і підвищену ефективність доступу до контенту. Особливостями такої розробки є інтеграція ефективних методів сортування, адаптація інтерфейсних функцій до різних типів користувачів.

Зараз серед кращих сервісів для перегляду аніме виділяють:

– Anitube;

– Eneyida;

– UASerial;

– AniPlay.

Anitube [10] є одним з найпопулярніших українських сайтів, які дозволяють користувачам дивитися аніме онлайн. На сайті є повний каталог аніме-серіалів і фільмів, відсортованих за жанрами, що полегшує користувачам пошук будь-якого контенту за допомогою пошукового інструменту, однак, він забезпечує лише обмежену кількість фільтрів і може зробити пошук ще більш проблематичним. Також немає вбудованої персоналізованої системи рекомендацій, яка допомогла б користувачам знайти нові назви відповідно до їхніх уподобань. На рисунку 1.1 показано інтерфейс вебзастосунку Anitube.

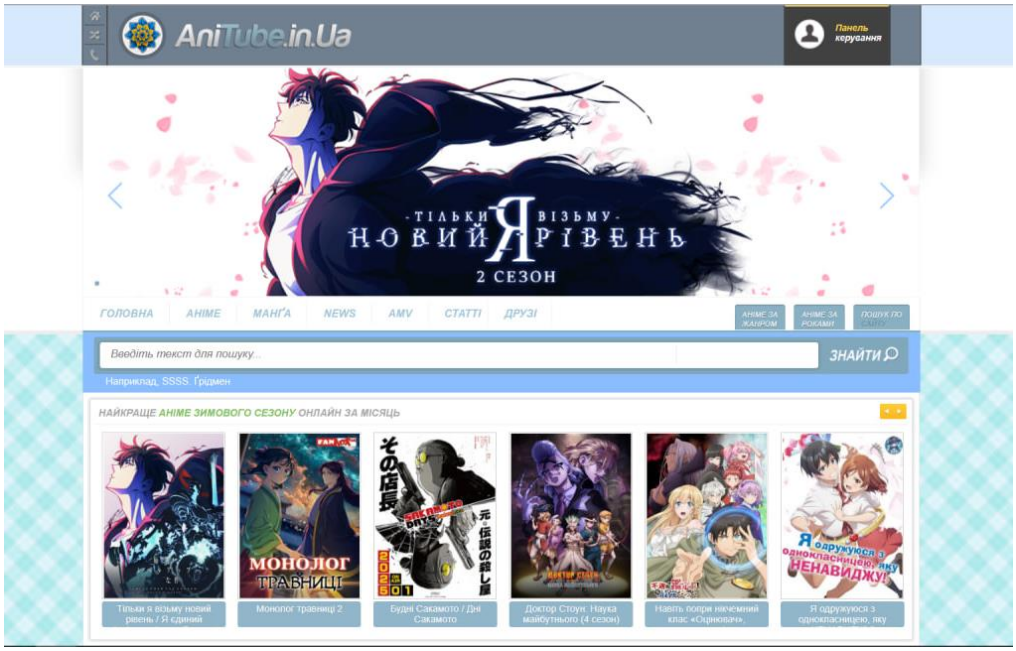

Рисунок 1.1 – Інтерфейс вебзастосунку Anitube

Інший аналог Eneyida [11] має широкий спектр анімованих фільмів, які можна сортувати на основі популярності, але немає можливості налаштовувати рекомендації на основі особистих переваг. вебзастосунок підтримує миттєвий перегляд серій без необхідності реєстрації, що є полегшенням для більшості користувачів. На рисунку 1.2 показано інтерфейс вебзастосунку Eneyida.

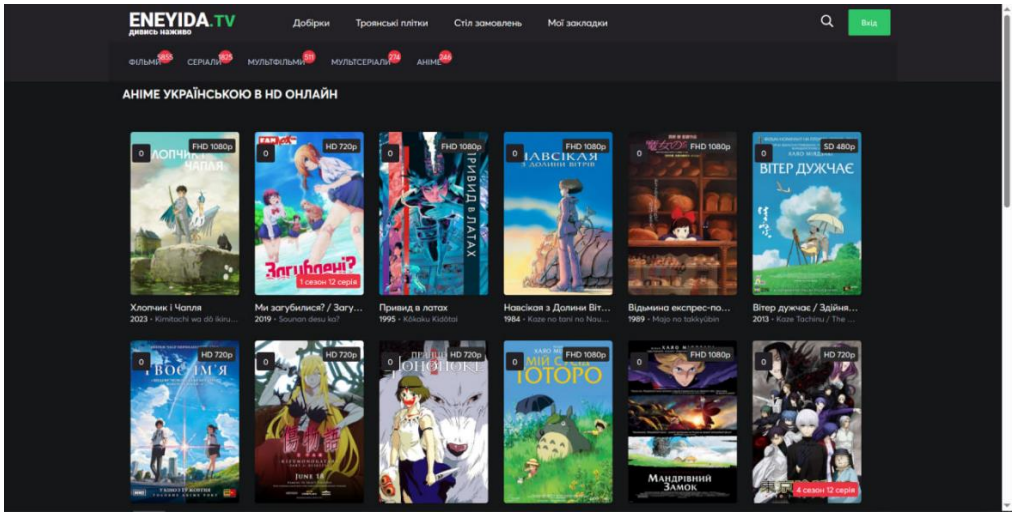

Рисунок 1.2 – Інтерфейс вебзастосунку Eneyida

Наступний аналог UASerial [12], даний вебзастосунок пропонує аніме контент, а також інші відео. Простий інтерфейс дозволяє легко шукати високо оцінені аніме-шоу. Але оскільки сайт має різні категорії контенту, бібліотека аніме може бути менш каталогізована тут порівняно з іншими спеціалізованими сайтами. На рисунку 1.3 показано інтерфейс вебзастосунку UASerial.

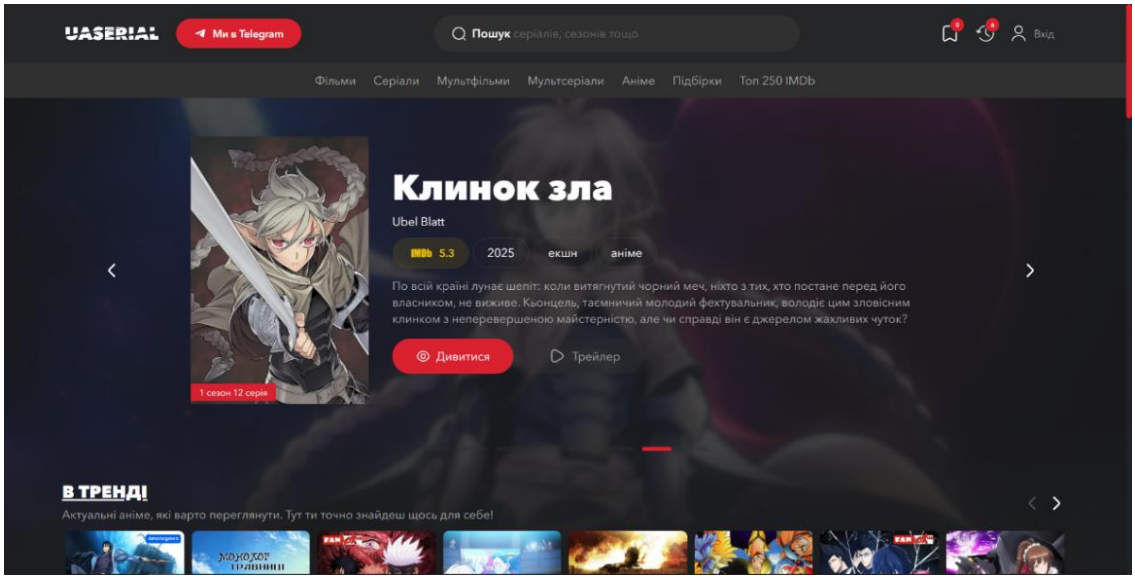

Рисунок 1.3 – Інтерфейс вебзастосунку UASerial

Наступним аналогом є AniPlay [13], який орієнтований на перегляд аніме-контенту онлайн. Вебзастосунок має сучасний інтерфейс і базову систему фільтрації за жанрами та типом аніме, що полегшує пошук популярних тайтлів.

Однак сайт використовує зовнішні джерела, структура його каталогу може бути менш систематизованою, ніж на спеціалізованих аніме-платформах. На рисунку 1.4 зображено головну сторінку AniPlay.

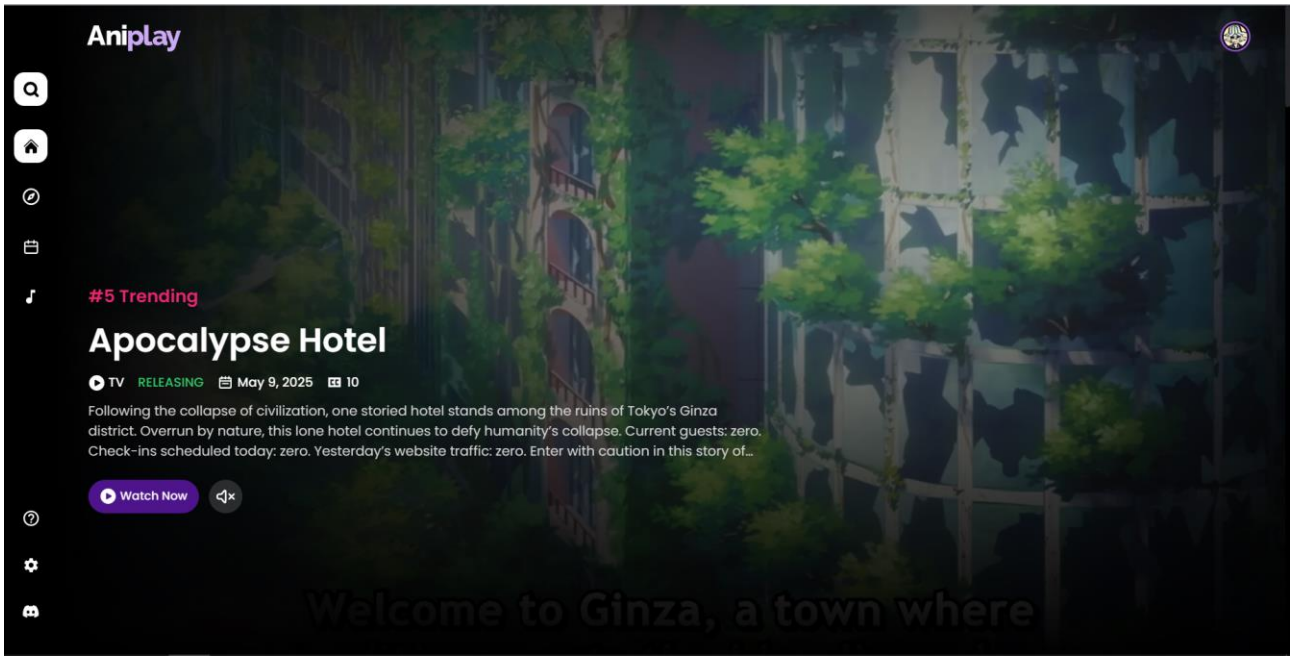

Рисунок 1.4 – Інтерфейс вебзастосунку Aniplay

Розробка власного вебзастосунку для пошуку та класифікації аніме вимагає створення ефективного механізму фільтрації та сортування контенту. Одним з найважливіших аспектів є включення функціональності, що дозволяє користувачам отримувати персоналізовані рекомендації, шукати аніме за різнимим критеріями і надавати простий доступ до детальної інформації по кожному фільму або серіалу. Ефективна система пошуку та класифікації має не лише спрощувати навігацію, а й покращувати користувацький досвід, надаючи релевантні рекомендації та зручний доступ до всієї необхідної інформації.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерії	Anitube	Eneyida	UASerial	Aniplay	Власна розробка
Список бажаного	+	-	+	+	+
Додатковий контент у вигляді легких новел	-	-	-	-	+
Велика кількість фільтрів пошуку	+	+	-	+	+
Система персоналізованих рекомендацій	-	+	-	-	-
Загальна оцінка	50%	50%	25%	50%	75%

Відповідно до таблиці порівняльних характеристик, розробка власного вебзастосунку для пошуку та класифікації анімованих фільмів є доцільною.

Власна система зможе усунути ці недоліки і ввести нові функції, які підвищать користувацький досвід. Застосунок буде оснащений такими функціями, як додавання анімованих фільмів до списку бажань, в якому будуть зберігатися фільми які можна подивитись в майбутньому.

Отже, розробка вебзастосунку для пошуку та класифікації аніме дозволить усунути недоліки існуючих рішень і забезпечити сучасний, зручний та функціональний інструмент для перегляду, сортування та рекомендації анімованих фільмів.

1.4 Постановка задач дослідження

Проаналізувавши переваги та недоліки існуючих вебплатформ для пошуку та класифікації аніме, було визначено основні завдання, які необхідно виконати для успішної розробки вебзастосунку:

– розробити механізм пошуку, що дозволить користувачам знаходити анімовані фільми;

– розробити зручний графічний інтерфейс користувача, який буде інтуїтивно зрозумілим та забезпечить швидкий доступ до всіх функцій вебзастосунку;

– реалізувати систему збереження списку бажаного, яка дозволить користувачам відстежувати аніме, яке вони планують переглянути;

– провести тестування вебзастосунку відповідно до поставлених завдань, щоб забезпечити стабільність роботи, коректність відображення даних та зручність використання.

1.5 Висновки

У першому розділі було проведено аналіз існуючих вебплатформ для пошуку та класифікації аніме. Було розглянуто такі вебзастосунки, як Anitube, Eneyida та UASerial. Кожен із цих сайтів має свої переваги та недоліки, які були детально проаналізовані та порівняні за основними критеріями, такими як наявність списку бажаного, функціональність пошуку, персоналізовані рекомендації та додатковий контент.

Проаналізувавши можливості та обмеження існуючих рішень, було сформульовано завдання для подальшої розробки вебзастосунку. Ці завдання включають створення ефективної системи фільтрації контенту, впровадження персоналізованих рекомендацій, розширення функціональності за рахунок інтеграції легких новел, а також розробку зручного та інтуїтивно зрозумілого інтерфейсу.

Отже, аналіз показав доцільність розробки власного вебзастосунку, який усуне недоліки існуючих платформ і запропонує користувачам нові можливості для пошуку, класифікації та збереження аніме-контенту. На основі отриманої інформації було сформовано перелік завдань, які необхідно виконати для реалізації ефективного та сучасного вебзастосунку.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних даних системи

При розробці програмного застосунку, для пошуку та класифікації анімованих фільмів, було використано вебтехнології, зокрема JavaScript [14] для взаємодії з користувачем.

На функціональність і ефективність будь-якої програмної системи, як-от вебпрограми, що створюється для пошуку та класифікації анімаційних фільмів, безпосередньо впливають дані, які вона отримує та з якими оперує. Перевірка вхідних даних є важливою фазою проєктування, оскільки вона допомагає прийняти рішення щодо вимог до структури зберігання, алгоритмів обробки та технологій реалізації. У рамках цієї роботи, коли клієнтська розробка виконується з використанням HTML [15], CSS [16] і JavaScript, поняття вхідних даних охоплює всі дані, необхідні для функціонування реалізованих модулів. Ці дані можна умовно розділити на ряд значущих категорій.

Найважливішою категорією є дані про контент – анімаційні фільми та телешоу. Це основне джерело інформації користувача, з яким він/вона взаємодіє за допомогою операцій пошуку, перегляду та додавання до списку бажань. Через характер проєкту та відсутність серверної сторони та бази даних джерелом даних для цього є статична структура даних, явно визначена у вихідному коді JavaScript. Зазвичай це досягається у вигляді масиву об’єктів або об’єкта словника, де один об’єкт або запис зарезервовано для однієї назви аніме. Структура кожного такого запису повинна мати набір властивостей, необхідних для правильного функціонування системи. Для кожного з них необхідний ідентифікатор аніме, який використовується для посилання та ідентифікації, коли хтось хоче додати його до списку бажань. Необхідно використовувати назву твору; для простоти в цій реалізації дозволено використовувати лише один заголовок. Не менш важливими є ознаки класифікації та фільтрації: тип контенту, наприклад, телевізійний серіал чи фільм, належність до жанру, як

правило, у вигляді списку жанрів, рік випуску чи період трансляції. Щоб показати користувачеві детальну інформацію, текстовий опис або синопсис, посилання на зображення плаката, яке знаходиться на картках і сторінці деталей. Також можна додати режисера, анімаційну студію, кількість епізодів серіалу, поточний статус випуску та назву виробничої студії. Від якості та повноти таких даних залежить фільтрація, пошук та інформативність системи для кінцевого користувача. Навіть зі статичними даними слід звернути увагу на їх послідовність і правильне форматування.

Друга важлива категорія — дані облікових записів користувачів. Оскільки система пропонує реєстрацію та авторизацію для полегшення персоналізованих функцій, таких як список бажань, дані про кожного зареєстрованого користувача повинні бути збережені. Ці дані пропонуються шляхом введення реєстраційних даних користувача. У цій роботі використовується функція зберігання даних на стороні клієнта, вбудована у веббраузер, щоб зберегти ці дані. Для кожного користувача зберігається його власна адреса електронної пошти, яка використовується як логін, відображається ім’я користувача та пароль. Слід підкреслити, що зберігання паролів на стороні клієнта є надзвичайно небезпечною практикою в реальних системах і використовувалося лише для спрощення в освітніх цілях. Також у локальному сховищі браузера разом із електронною поштою користувача зберігається його власний список побажань, як правило, у вигляді масиву ідентифікаторів або аніме-об’єктів, які він сам додав. Також може зберігатися посилання на аватар користувача. Ці дані зчитуються користувачем під час входу та використовуються для персоналізації інтерфейсу та поведінки, і чи зберігаються вони під час сеансів чи ні, залежить від налаштувань браузера користувача та відсутності дій для видалення інформації сайту.

Третя категорія — це дані, що генеруються під час сеансу користувача з системою. Це пошукові запити, які користувач вводить у відповідне поле. Цей фрагмент тексту використовується логікою JavaScript для сортування основного

масиву даних аніме. Сюди також входить взаємодія користувача зі списком побажань – натискання на символ сердечка, які викликають оновлення відповідного масиву даних у сховищі браузера. Зміни користувача на сторінці профілю, такі як нове ім’я, новий пароль або завантажений аватар, також є вхідними даними, обробленими та збереженими системою в локальному сховищі на стороні клієнта.

Аналіз категорії даних наголошує на виборі технологічного стека. Оскільки основний вміст не змінюється, а дані користувача разом із їхньою взаємодією зберігаються локально на стороні клієнта, реалізація структурування за допомогою HTML, стилізації за допомогою CSS і JavaScript для всієї логіки обробки таких даних є логічним і правильним підходом до реалізації запланованої функціональності. JavaScript запускає масиви даних для пошуку, перевіряє вхідні дані, спілкується з системою зберігання даних браузера, переводячи дані в належну структуру та зчитуючи їх назад, а також динамічно оновлює вебсторінку для відображення відповідних даних. Дизайн інтерфейсу користувача з використанням HTML і CSS дозволяє відображати цю інформацію в зручному і приємному на вигляд вигляді, а логіка JavaScript підтримує її динамічне оновлення і обробку на основі активності користувача. Знання структури та джерел вхідних даних дозволило правильно спроєктувати програмні модулі та їх взаємодію на стороні клієнта.

2.2 Розробка моделі системи

Для візуалізації різних аспектів програмних систем існує уніфікована мова моделювання UML [17], яка пропонує набір стандартизованих діаграм. Вибір конкретного типу діаграми залежить від того, який саме аспект системи необхідно змоделювати. Наприклад, діаграми варіантів використання описують взаємодію користувачів із системою, діаграми класів показують статичну структуру системи, а діаграми послідовності ілюструють взаємодію об'єктів у часі.

Моделювання в розробці програмного забезпечення відіграє головну роль, оскільки воно дозволяє візуалізувати структуру, поведінку та взаємодію компонентів системи ще до написання коду. Це сприяє кращому розумінню вимог як розробником, так і іншими зацікавленими сторонами, полегшує комунікацію в команді, допомагає виявити потенційні проблеми, неузгодженості чи логічні помилки на ранніх стадіях проєктування, що значно дешевше виправити, ніж на етапі кодування чи тестування. Крім того, створені моделі слугують важливою частиною технічної документації застосунку.

Для кращого розуміння діяльності модулів вебзастосунку пошуку та класифікації анімованих фільмів розроблено модель системи на прикладі діаграми активності UML (рис. 2.1). Як видно з представленої схеми, взаємодія користувача з системою починається з відкриття застосунку. На першому кроці користувач вводить пошуковий запит, який може містити назву твору, жанр, рік видання або додаткові фільтри.

Система перевіряє правильність вхідних даних, а саме, чи запит не порожній або недійсний за форматом. У разі помилки користувачеві видається відповідне повідомлення. У разі дійсного запиту система надсилає його внутрішньої бази даних для отримання відповідних результатів.

Другий крок – обробка отриманих даних. Система сортує та класифікує за встановленими параметрами, а потім перевіряє, чи відповідають знайдені результати. У разі відсутності відповідних записів користувачеві повертається повідомлення про відсутність даних. У разі успішного пошуку результати виводяться у формі списку.

Користувач має можливість детальніше ознайомитися з обраною роботою, а також мати можливість застосувати додаткові фільтри для пошуку. Також можна взаємодіяти з елементами інтерфейсу користувача, такими як сортування на основі рейтингу, популярності чи дати випуску.

Після взаємодії користувачеві пропонується виконати повторний пошук або завершити сеанс, закривши вебзастосунок.

Отже, наступна діаграма активності інкапсулює процеси вебсистеми та пропонує логічну структуру, яка полегшує подальший розвиток і вдосконалення її функціональності.

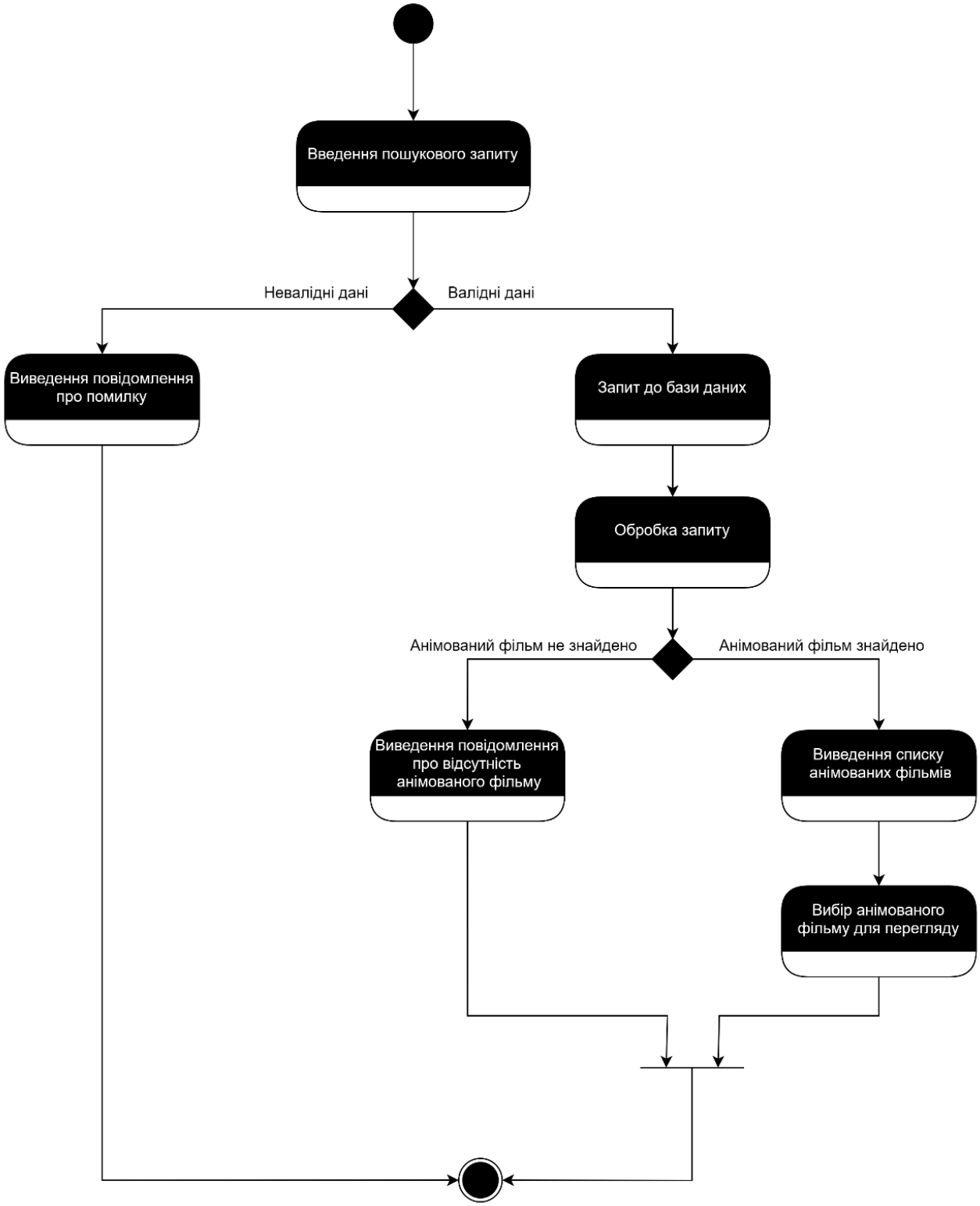

Рисунок 2.1 – Модель роботи системи у вигляді UML

Розробка діаграми діяльності була виконана у програмному забезпеченні draw.io [18].

2.3 Розробка алгоритмів роботи системи

Для розробки вебсистеми, що призначена для ефективного пошуку й класифікації аніме-фільмів, був розроблений загальний алгоритм її функціонування, представлений у формі UML-діаграми діяльності (рис. 2.2). Ця система створена з метою оптимізації процесу пошуку аніме-контенту, забезпечуючи користувачам швидкий доступ до бази анімованих фільмів з функціями фільтрації та сортування результатів.

Згідно з алгоритмом, початковим етапом є ініціалізація інтерфейсу, що надає можливість користувачеві взаємодіяти із системою. На цьому етапі відбувається завантаження основних компонентів інтерфейсу, включаючи поле введення запиту, кнопки навігації та фільтрації контенту, а також списки категорій та жанрів.

Наступним кроком є введення пошукового запиту, який може містити назву анімованого фільму, що конкретизують бажаний контент. Введені дані передаються до системи для подальшої обробки.

Після отримання запиту, система звертається до внутрішньої бази даних, де зберігається інформація про анімовані фільми. Далі проводиться перевірка на наявність відповідних записів у базі. Якщо база містить мультфільми, що відповідають запиту, система формує та відображає користувачеві список знайдених результатів. У випадку відсутності відповідних записів, система інформує користувача про це.

На наступному етапі користувач отримує можливість переглянути список анімованих фільмів, що відповідають його запиту. Система надає можливість відкрити розширену інформацію про кожен знайдений анімований фільм. У цьому розділі відображається детальна інформація, зокрема рік випуску, студія-

виробник, режисер, короткий опис сюжету, рейтинг користувачів, доступні мови озвучення та субтитри.

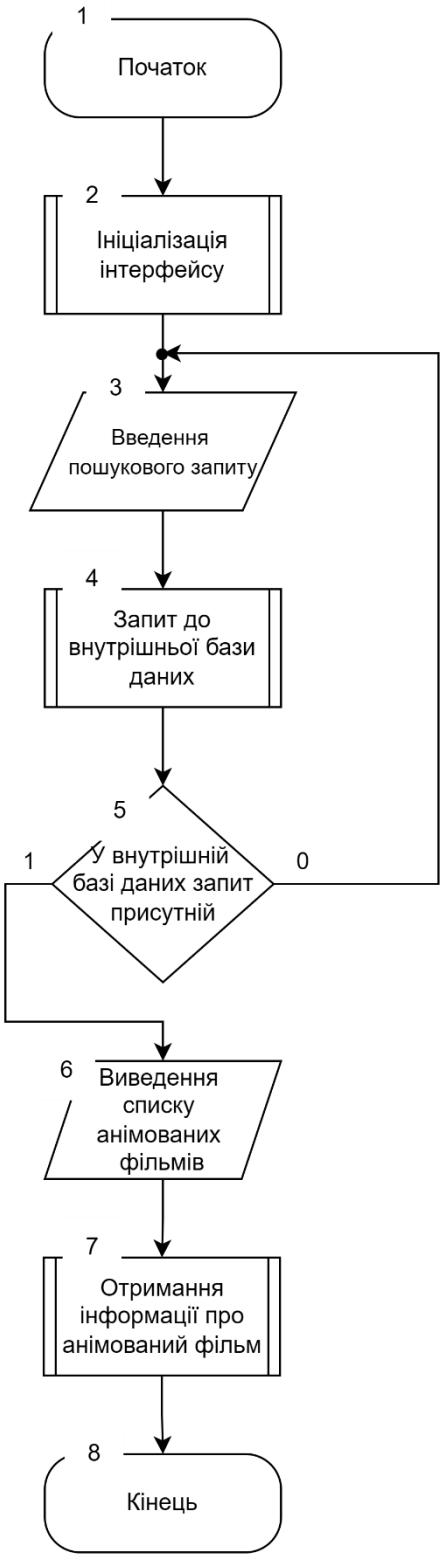

Рисунок 2.2 – Алгоритм роботи програмного застосунку

Отже, розроблений алгоритм забезпечує швидкий та зручний пошук аніме-контенту, спрощуючи навігацію серед великої кількості мультфільмів. Завдяки автоматизованій обробці запитів система дозволяє користувачам легко знаходити потрібний контент, що значно підвищує ефективність взаємодії з вебзастосунком.

Алгоритм реєстрації користувача (рис. 2.3) описує процес реєстрації нового користувача в системі. Процес розпочинається і переходить до ініціалізації форми реєстрації, де користувачеві надається модальне вікно для введення своїх даних.

Наступним кроком є введення користувачем своєї електронної пошти. Після введення email система здійснює перевірку чи існує вже в базі даних користувач з таким email. Якщо так користувачеві відображається повідомлення про необхідність змінити email і він повертається до кроку 3 для повторного введення. Якщо ж користувача з таким email не знайдено, процес переходить до наступного етапу.

Далі користувач вводить своє ім’я користувача. Система проводить перевірку чи існує вже користувач з таким username. Якщо так користувачеві показується повідомлення про необхідність змінити username. Якщо username є унікальним, процес продовжується.

Далі користувач вводить свій пароль. Після цього система перевіряє чи є введений пароль меншим за 6 символів. Якщо так, користувачеві відображається повідомлення про необхідність змінити пароль. Якщо пароль відповідає вимогам щодо довжини, система переходить до наступного кроку.

Далі система отримує введені користувачем атрибути email, username та password. Після успішного отримання цих даних відбувається створення нового облікового запису користувача в системі. На цьому процес реєстрації завершується.

Отже, діаграма алгоритму реєстрації нового користувача наочно демонструє покроковий алгоритм реєстрації, включаючи валідацію введених даних на унікальність та відповідність мінімальним вимогам.

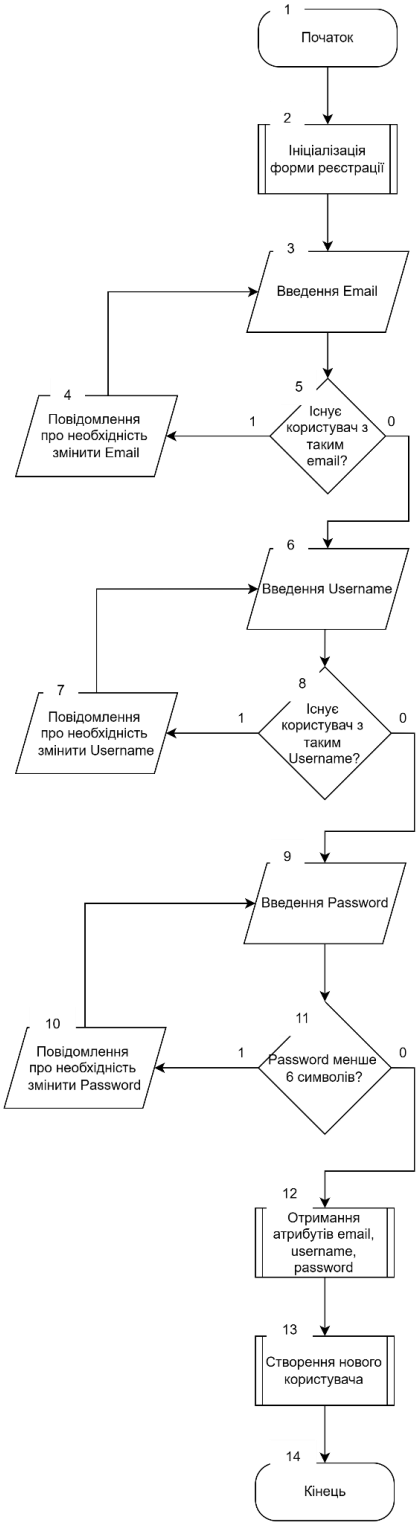

Рисунок 2.3 – Алгоритм реєстрації користувача

2.4 Висновки

У другому розділі здійснено аналіз інформації, що надходить до вебсистеми, призначеної для пошуку та класифікації анімованих фільмів. Було розроблено алгоритм функціонування вебзастосунку, котрий охоплює обробку звернень, пошук у базі даних, фільтрування й упорядкування отриманих даних. Для наочного представлення процесів були створені UML-діаграми, що візуалізують активність системи та взаємодію користувачів з нею.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ВЕБСИСТЕМИ ПОШУКУ ТА КЛАСИФІКАЦІЇ АНІМОВАНИХ ФІЛЬМІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Щоб побудувати програмні модулі для вебпошуку та класифікації в системі анімаційних фільмів, для платформи веброзробки було обрано HTML, CSS і JavaScript. Розробка програми як вебсторінки забезпечує плавний доступ на будь-якому пристрої, на якому запущено сучасний браузер [19], повністю обходячи необхідність попередньої інсталяції.

Мова розмітки гіпертексту служила основною структурою для структурування всього вмісту програми. HTML допоміг у визначенні логічного розташування кожної сторінки. Використання семантичних тегів HTML5, таких як <header>, <nav>, <main>, <section> і <article>, забезпечило не тільки організацію вмісту, але й покращену доступність і можливість індексації. Для форм реєстрації, входу, скидання пароля та зміни імені користувача було створено HTML-структури з використанням відповідних типів полів введення тексту, електронної пошти, пароля та файлу, для полегшення попередньої перевірки даних на основі браузера та покращення простоти введення.

Каскадні таблиці стилів було використано, щоб покращити візуальне представлення програми, визначити її естетичні якості та гарантувати зручну роботу користувача. CSS відповідав за всі аспекти візуального стилю, включаючи колірні схеми, вибір гарнітури, розмір шрифту та компонування елементів інтерфейсу користувача, а також визначення відступів і полів. Для створення макетів сторінок, особливо для розміщення заголовка, основної області вмісту, будь-яких бічних панелей і сітки аніме-карток, широко використовувалися сучасні інструменти CSS Flexbox [20] і Grid Layout [21]. Ці інструменти дозволили розробити гнучку та адаптовану структуру, здатну належним чином масштабувати та реорганізовувати екрани різних розмірів,

атрибут, перевірений за допомогою реалізації медіа-запитів. Крім того, CSS також використовувався для включення візуальних ефектів, таких як тіні, заокруглені кути та плавні переходи в станах елементів при наведенні або зміні стану, що значно покращило інтерактивність із діями користувача. Суворе дотримання принципів поділу структу і представлення сприяло більш організованому та легко модифікованому стилю коду.

JavaScript був двигуном динамізму та інтерактивності всієї програми. Оскільки застосунок покладався виключно на клієнтські технології, JavaScript був виконавцем усієї програмної логіки. По-перше, JavaScript широко використовувався для маніпулювання DOM [22]. Це включало динамічне генерування елементів HTML, модифікацію їхніх характеристик і повне їх видалення. Маніпуляції з DOM дозволили оновлювати список анімованих фільмів під час пошуку та відображати список побажань без необхідності перезавантажувати сторінку. По-друге, JavaScript обробляв усі взаємодії користувача з інтерфейсом через систему обробки подій. Обробники були закодовані для натискання кнопок, активації посилань і вибору піктограм, для подій введення тексту в полі пошуку і для події надсилання форми. Це дозволяло перехоплювати стандартну поведінку браузера та виконувати налаштовану логіку програми. По-третє, уся логіка на стороні користувача була реалізована в JavaScript, включаючи функціональность пошуку, перевірки даних, введених у форми реєстрації та входу, керування статусом списку бажань, а також реалізовувати зміни пароля та імені користувача. Масиви даних, які зберігаються в LocalStorage [23], що стосуються зареєстрованих користувачів і їхніх конкретних списків бажань. JavaScript відповідав за запис, читання та оновлення цих збережених даних. Нарешті, JavaScript забезпечив відображення динамічних елементів інтерфейсу, наприклад, модальних вікон, призначених для редагування профілю та сповіщення користувача про виконані операції.

Отже, поєднання HTML для структурування інформації, CSS для візуального оформлення та JavaScript для створення інтерактивної логіки та

динамічного оновлення даних утворює повну технологічну основу для створення ефективних програмних модулів для системи пошуку та категоризації анімованих фільмів.

3.2 Розробка модуля збереження в список обраного

Розробка списку бажаного починається з функції «displayContent» для заповнення вебсторінки вмістом, представленим у вигляді карток.

Після ініціалізації здійснюється виклик для отримання посилання на деякий структурний компонент системи. Після цього виконується умовна перевірка на існування або дійсність компонента.

Якщо елемент невідомий або розглядається як відсутній, потік спрямовується на шлях, який включає реєстрацію повідомлення про відмову або помилку, після чого відбувається ймовірне завершення процесу або його переведення в стан очікування.

Після успішної ідентифікації об'єкта починається основний шлях виконання. Початковою дією в цьому напрямку є зміна внутрішнього складу отриманого об'єкта, що передбачає його очищення або заміну.

Другою умовною дією є перевірка стану будь-якого набору даних або структури даних. Перевірка гарантує, що дана колекція не містить жодного елемента.

Якщо набір даних порожній, процес переходить до введення заздалегідь встановленого контенту, який би відображав відсутність даних, після чого виконується останній процедурний захід.

Якщо збір даних містить елементи, процес переходить у гілку, яка передбачає ітеративну обробку даних.

Для кожного елемента колекції виконується низка кроків: створюється новий елемент структури з властивостями, взятими з поточного елемента даних; на основі даних встановлюється внутрішня структура цього елемента, а потім новий елемент додається до вже створеного основного елемента структури. Цей

процес виконується для всіх елементів колекції. Після завершення ітеративної обробки даних або після вставки вмісту заповнювача у випадках, коли колекція не містить елементів, виконується завершальна процедурна фаза, яка передбачає виклик визначеної функції або процедури. Процес завершується в точці виходу, яка позначає кінець алгоритму. Таким чином, діаграма (рис. 3.1) ілюструє типовий потік обробки даних з отриманням ресурсів, верифікацією, умовним розгалуженням на основі стану даних і ресурсів, а також змістовними і структурними операціями.

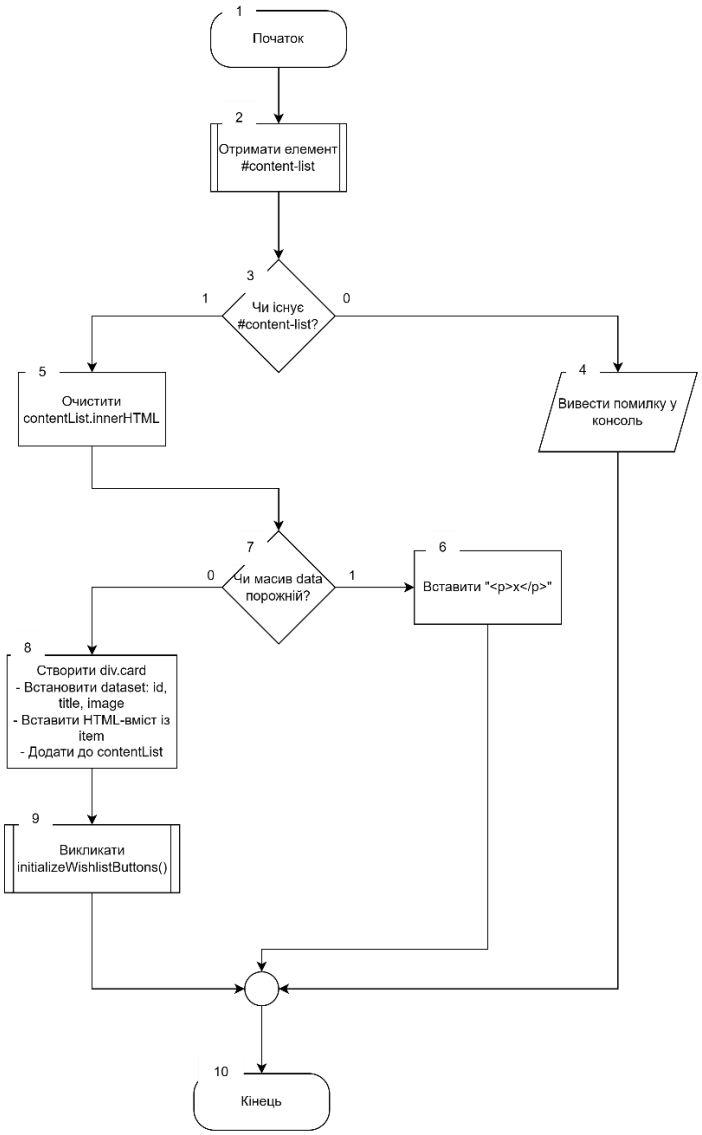

Рисунок 3.1 – Алгоритм заповнення вебсторінки картками

Функція «initializeWishlistButtons» відповідає за додавання кнопки у вигляді анімованого серця до кожної картки контенту.

На початковому етапі робляться спроби ідентифікувати або отримати інформацію про поточний об'єкт у системі. Потім виконується умовна оцінка, щоб перевірити, чи існує ця сутність, чи ні, або чи є вона дійсною. Якщо сутність не існує або є недійсною, то процес перенаправляється на шлях, який полегшує документування помилки або відхилення, після чого процес завершується. Якщо об'єкт ідентифіковано успішно, то потік продовжує рухатися первинним шляхом. На цьому шляху відбувається пошук даних, що стосуються знайденої сутності, в межах певної функціональності. На наступному етапі виконується умовна перевірка, щоб виявити, чи присутній певний структурний елемент або контейнер в інтерфейсі користувача або в дизайні системи.

Якщо цей структурний компонент не з'являється, решта операції, що продовжується після його виявлення, зупиняється або переходить до завершеного стану, незалежно від відсутності окремого механізму обробки помилок. Після визначення структурного елемента виконується операція вилучення певних атрибутів даних, таких як ідентифікатор, описовий текст та URL-адреса зображення із залученого джерела даних щодо поточного контексту. Отримані атрибути даних проходять умовну перевірку на цілісність та доступність.

Якщо всі необхідні атрибути даних доступні та перевірені, виконується процес створення нового елемента або компонента користувацького інтерфейсу в наданому контейнері. Створений новий елемент призначений для візуального або функціонального представлення асоціації з автентифікованими даними. Після завершення всіх застосовних процедур, на основі результатів умовних тестів, процес досягає точки виходу, яка позначає його завершення.

Діаграма алгоритму додавання кнопки у вигляді анімованого серця до кожної картки контенту показано на рисунку 3.2.

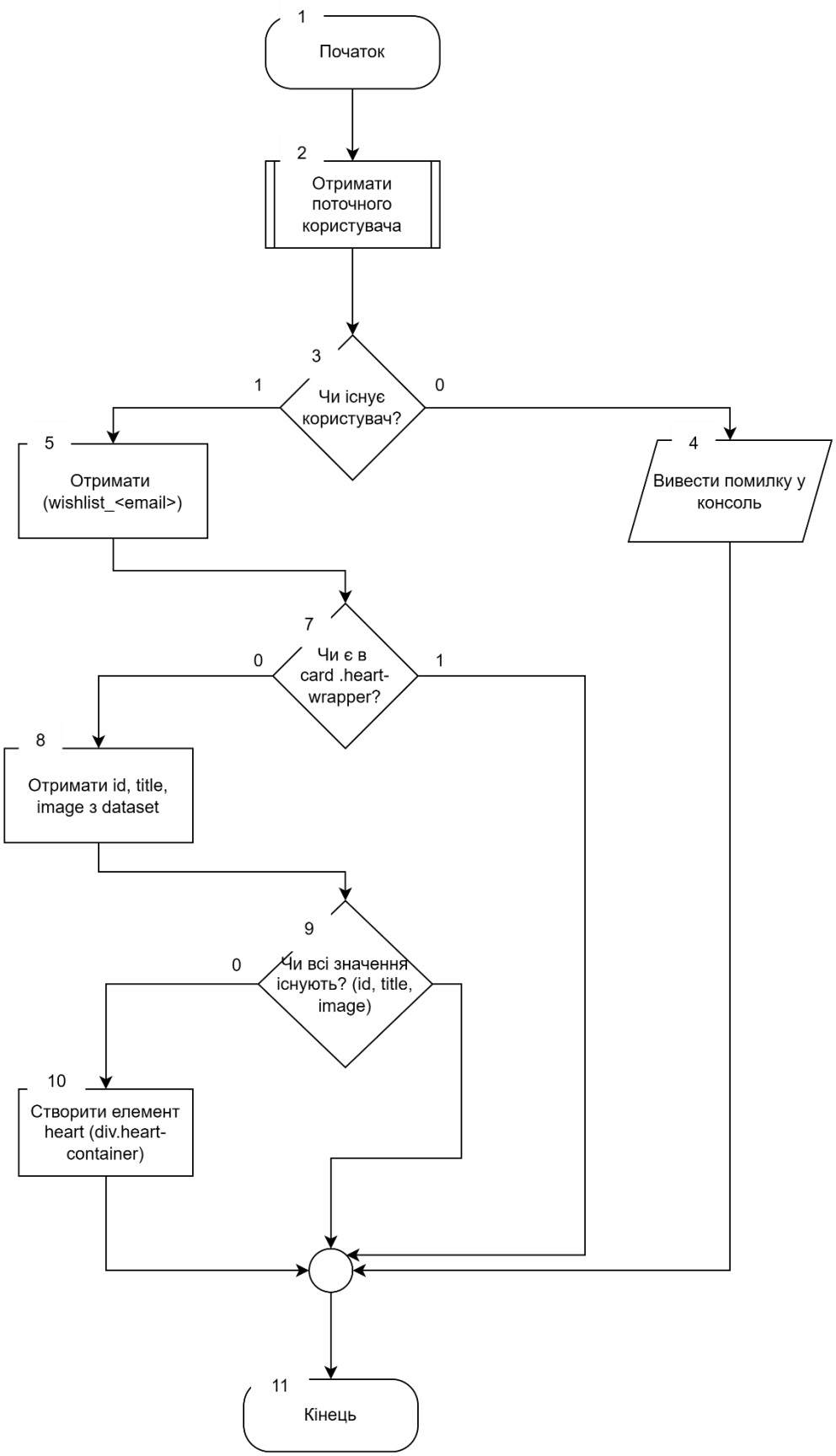

Рисунок 3.2 – Діаграма алгоритму додавання кнопки

3.3 Розробка модуля для пошуку анімаційних фільмів

Було реалізовано модуль динамічного пошуку. Основною метою розробки стало забезпечення користувача функціоналом, який дозволяє миттєво знаходити потрібний фільм за його назвою без необхідності перезавантаження сторінки чи натискання кнопок.

На першому кроці докладаються зусилля для отримання посилань на різні важливі елементи інтерфейсу або функціональні компоненти системи. Потім виконується умовна перевірка на наявність або успішну ініціалізацію цих елементів. Якщо будь-який з необхідних компонентів не може бути отриманий, процес переходить до гілки, яка передбачає або реєстрацію, або виведення повідомлення про помилку, таким чином ефективно зупиняючи подальше виконання основного потоку. З іншого боку, якщо всі компоненти успішно знайдено та отримано, процес продовжується відповідно. У цьому випадку виконується крок підключення системи обробки подій до одного з наданих компонентів, який має відреагувати на певну взаємодію або зміну стану.

Потім модифіковане значення проходить умовний аналіз вмісту, включаючи перевірку на порожній вміст. Залежно від результату цієї перевірки процес обирає один з двох можливих напрямків виконання, якщо значення дорівнює нулю, для подальшої обробки береться весь вихідний набір даних в іншому випадку, якщо значення не дорівнює нулю, вихідний набір даних фільтрується відповідно до збігу атрибуту елементів набору даних з перетвореним значенням.

Отриманий в результаті прямого відбору або фільтрації набір даних піддається ще одному умовному тесту, який перевіряє його розмір, тобто чи є в ньому елементи. На основі результатів цієї перевірки керується видимість певного елемента інтерфейсу, який створюється для показу сповіщень, він відображається, коли набір даних порожній, і залишається прихованим, коли в наборі даних є елементи. Діаграму алгоритму пошуку показано на рисунку 3.3.

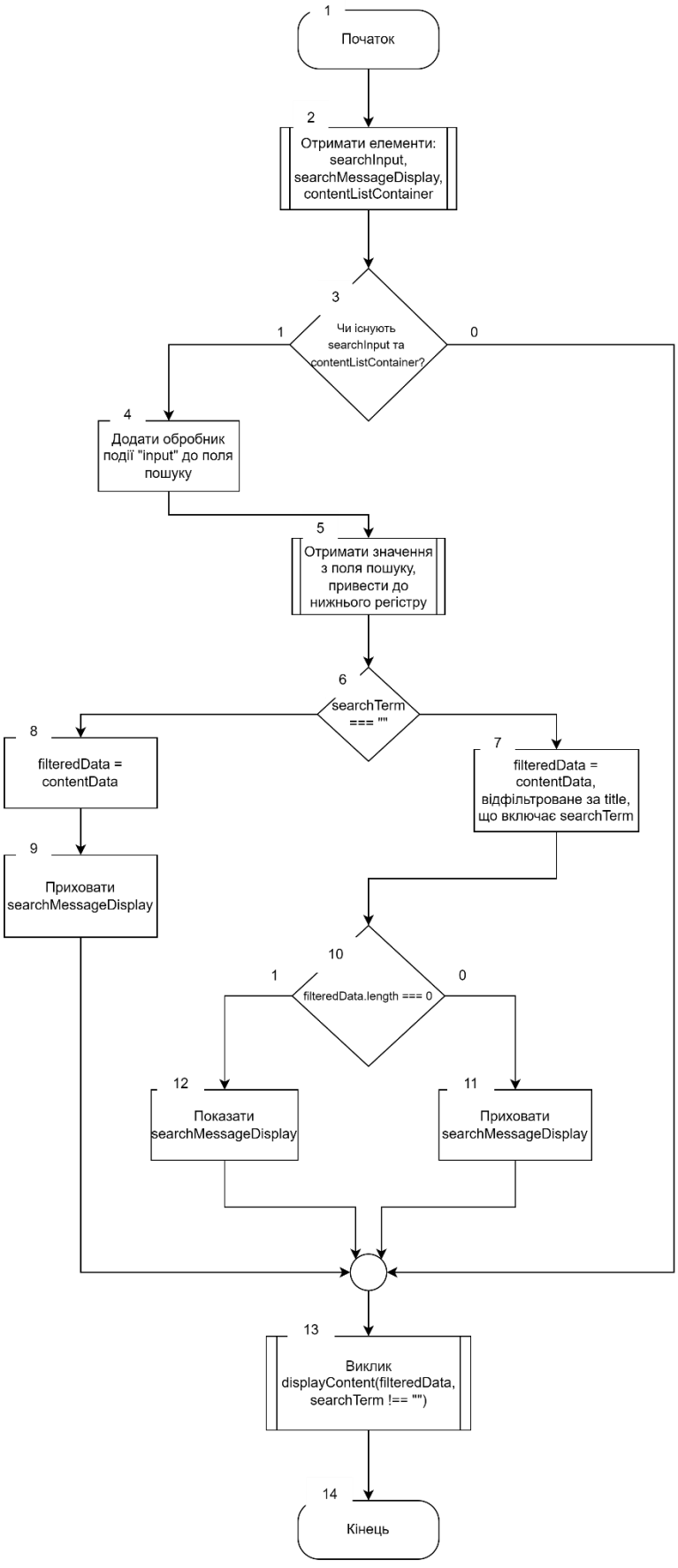

Рисунок 3.3 – Діаграма алгоритму пошуку

3.4. Розробка інтерфейсу програмного застосунку

Дизайн інтерфейсу користувача і забезпечення хорошого досвіду користувача є важливими аспектами під час розробки будь-якого застосунку, особливо того, який призначений для перегляду та взаємодії з візуальним вмістом, таким як анімовані фільми. Для вебзастосунку, зосередженого на пошуку та класифікації анімаційних фільмів, найважливішими аспектами при розробці інтерфейсу були простота використання, інтуїтивність та ефективність у наданні доступу до основних функціональних можливостей. Було розроблено середовище, в якому користувач міг би зручно орієнтуватися на сайті, знаходити потрібну інформацію та виконувати дії з меншими зусиллями.

Дизайн базувався на принципах чистоти, послідовності та забезпечення чіткого зворотного зв’язку з користувачем. Як тільки користувач заходить на сайт, він потрапляє на головну сторінку, яка є центральним інформаційним центром і воротами до всіх функцій системи. Головна сторінка призначена для надання користувачеві миттєвого перегляду доступного вмісту та інструментів для негайного подальшого використання.

Хедер сайту займає верхню частину сторінки, забезпечуючи користувачеві постійну навігацію програмою (рис. 3.5).

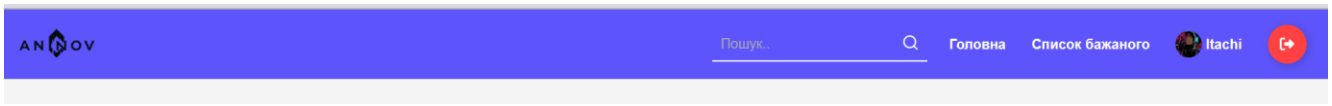

Рисунок 3.5 – Хедер застосунку

Він містить логотип для повернення на домашню сторінку, основні навігаційні посилання, такі як «Домашня сторінка», «Список бажань» і «Профіль». Іншим важливим компонентом заголовка є включене вікно пошуку, яке дозволяє користувачам починати пошук аніме за його назвою на будь-якій сторінці програми. У правій частині заголовка знаходяться елементи керування обліковим записом, для неавторизованих користувачів це виділені кнопки

«Вхід» і «Реєстрація», які після авторизації замінюються іменем поточного користувача, аватаром і кнопкою «Вийти». Ця динамічна зміна заголовка забезпечує візуальний зворотний зв’язок щодо стану авторизації користувача.

Зверху головної сторінки, трохи нижче хедера, є розділ «Популярні», реалізований у стилі інтерактивної каруселі (рис. 3.6).

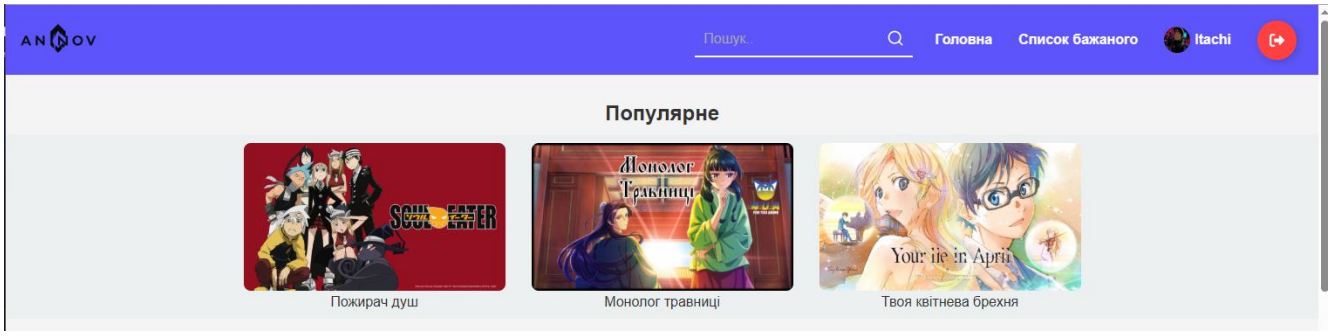

Рисунок 3.6 – Розділ «Популярне»

Він слугує для акцентування уваги на поточному чи трендовому вмісті, надаючи користувачеві естетично приємний спосіб познайомитися з новими аніме без необхідності активного пошуку.

Основний вміст головної сторінки та сторінки результатів пошуку представлений у вигляді сітки аніме-карток (рис. 3.7).

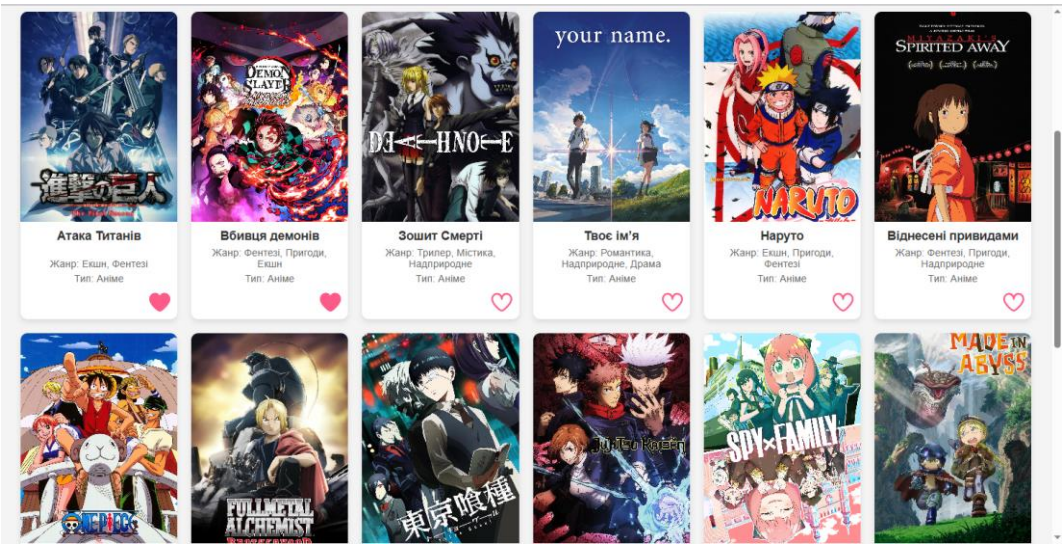

Рисунок 3.7 – Дизайн списку анімованих фільмів

Цей вид картки був обраний як доречний прийом для візуалізації списку компонентів з функціоналом представлення необхідної інформації у згорнутому форматі, великий постер, назва роботи, первинні властивості та інтерактив – іконка «улюблене», щоб мати можливість додати анімований фільм до списку бажань. Розташування елементів на картці визначено ієрархічно, щоб заголовок і плакат привернули першочергову увагу. Було використано CSS Grid Layout, щоб отримати адаптивну сітку, яка автоматично змінює кількість стовпців залежно від ширини екрана, тому презентація буде найкращою як на комп'ютерах, так і на мобільних пристроях. Ефект наведення, тобто тонкий підйом карти та рух тіні, додає інтерактивності та візуального зворотного зв'язку.

Сторінка профілю користувача – це особистий простір, у якому збирається інформація про обліковий запис та керування особистими даними. Це показано на рисунку 3.8.

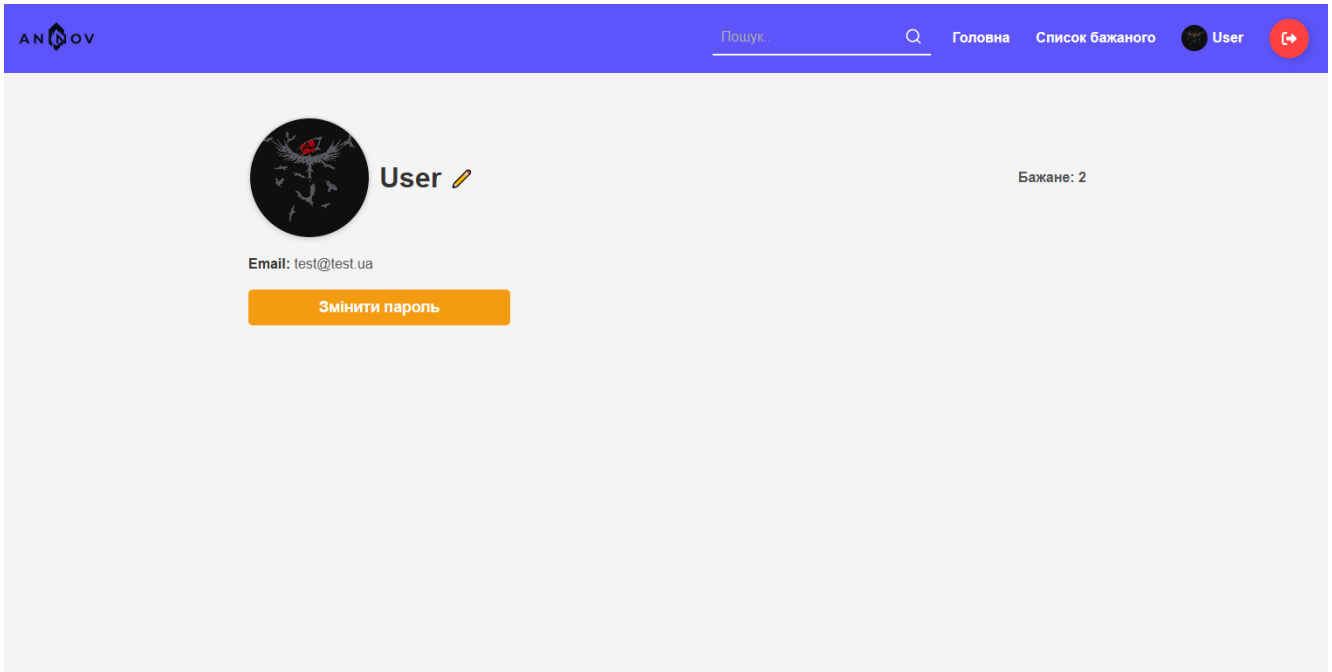

Рисунок 3.8 – Профіль користувача

Дизайн цієї сторінки передбачає чіткий візуальний фокус на аватарі та імені користувача, відображення його адреси електронної пошти та статистики. Редагування функціональних можливостей здійснюється за допомогою простих елементів керування, натискання самого аватара використовується для запуску процесу завантаження нового зображення, спеціальний значок поруч із іменем користувача викликає процедуру модального вікна для його перейменування, а інша кнопка призначена для виклику процесу зміни пароля.

Важливою особливістю персоналізованого досвіду користувача є сторінка «Список бажань» (рис. 3.9).

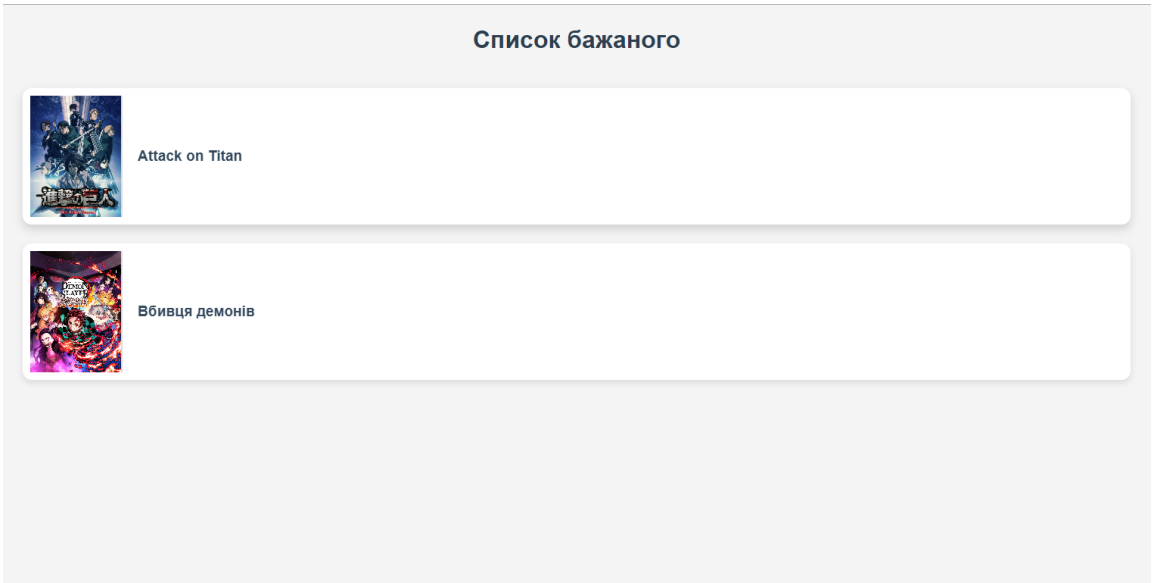

Рисунок 3.9 – Список бажаного

Це власний простір користувача, у якому об’єднуються всі аніме, які користувач позначив для подальшого перегляду, поставивши їм значок серця на картках у загальному каталозі або в результатах пошуку. Навігація до цієї сторінки зазвичай здійснюється за допомогою відповідного пункту навігаційного меню в заголовку сайту та зарезервована лише для зареєстрованих та авторизованих користувачів. Якщо неавторизований користувач спробує відкрити це посилання або якщо автентифікований користувач поки що не має нічого у своєму списку бажань, на сторінці відобразиться відповідне

інформаційне повідомлення, наприклад, «Увійдіть, щоб переглянути список бажань» або «Список бажань порожній».

Після вибору певного анімованого фільму зі списку або в результаті пошуку користувач потрапляє на сторінку, де він може побачити інформацію про анімований фільм (рис. 3.10).

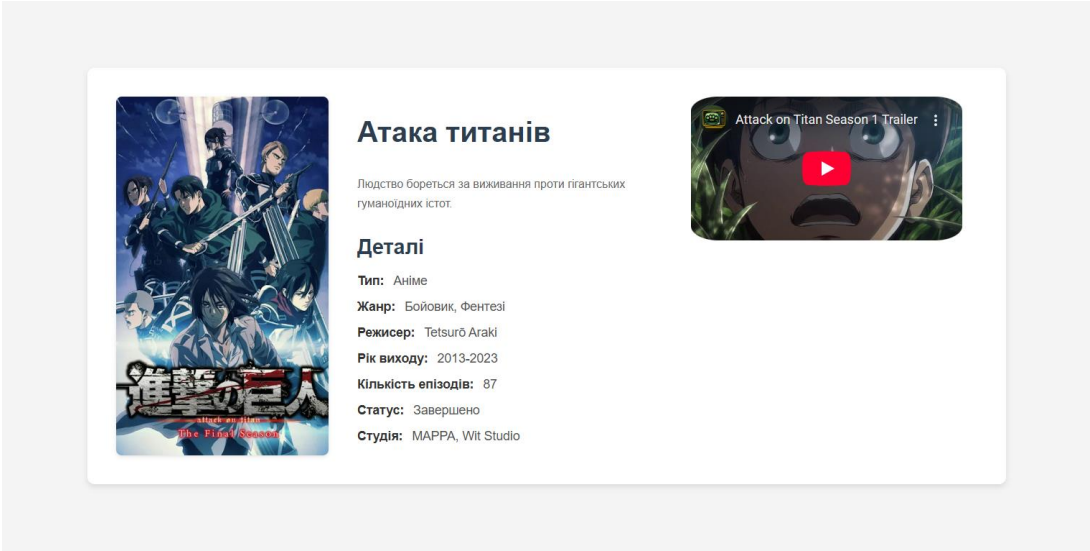

Рисунок 3.10 – Сторінка з інформацією про анімований фільм

Ця сторінка існує, щоб надати детальну інформацію про вибрану назву аніме.

Переходячи на цю сторінку, користувач миттєво стикається з візуальним матеріалом.

Потім користувач знаходить фрагмент тексту, який є описом або коротким викладом сюжету, і це дає деяку вказівку на історію та найважливіші події твору. Під описом наведено блок додаткової інформації у форматованому вигляді. Ця сторінка містить відомі атрибути аніме, такі як його жанр, тип роботи, ім’я режисера, рік випуску, номер епізоду, поточний статус виробництва та назва анімаційної студії. Сторінка відображає лише ті характеристики, про які зараз доступна інформація для цього конкретного аніме.

Сторінки також містять розділ для перегляду офіційного трейлера, якщо такий був наданий для аніме. Це дозволяє користувачеві миттєво відчути візуальний стиль і настрій перед переглядом.

Компонування елементів на сторінці зазвичай призначене для читабельності. Зображення та текст розміщуються поруч на великих екранах, тоді як на менших екранах вони накладаються один на інший для кращої читабельності.

Важливою частиною є система сповіщень у спливаючих вікнах. Ці короткі повідомлення автоматично з’являються у верхній частині екрана після виконання певних операцій і інформують користувача про результат. Вони мають чіткий візуальний стиль, щоб вказувати на успіх або помилку, і автоматично зникають через кілька секунд, не переповнюючи інтерфейс протягом тривалого часу.

Структура сторінки та логіка переходу між сторінками виконуються за допомогою звичайного HTML і JavaScript. Динамічне оновлення вмісту та залучення користувачів забезпечуються лише за допомогою JavaScript.

3.5 Висновки

У третьому розділі було обґрунтовано вибір технологій та інструментів, що використовувались для розробки вебзастосунку. Було прийнято рішення використовувати HTML для структурування вмісту сторінки, CSS для стилізації інтерфейсу, а також JavaScript для реалізації логіки взаємодії з користувачем та динамічного оновлення контенту. Детально описано розробку модуля додавання анімаційних фільмів до списку бажаного. Також, описано модуль пошуку, який надає можливість фільтрації контенту за назвою анімаційного фільму в режимі реального часу.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Тестування системи

Тестування програмного забезпечення [24] є життєво важливим етапом життєвого циклу розробки будь-якого програмного продукту, включаючи вебзастосунки. Це процес аналізу та оцінки програмного забезпечення на відповідність визначеним вимогам, виявлення дефектів і надання зацікавленим сторонам інформації про його якість.

Якісне тестування підвищує надійність, оскільки дозволяє програмі працювати передбачувано та послідовно. Він оцінює функціональну коректність, перевіряючи, що всі вказані функції реалізовано та функціонують відповідно до визначених вимог. Крім того, тестування відіграє важливу роль у підвищенні зручності використання, оцінюючи легкість і простоту взаємодії користувача з інтерфейсом. Хоча це не включено в цю роботу, варто зазначити, що тестування може включати тестування продуктивності системи та безпеки.

Існує ряд класифікацій видів тестування. За рівнем доступності коду розрізняють тестування «чорного ящика», при якому взаємодія тестувальника з програмою обмежена її інтерфейсом і ніколи не з вихідним кодом; тестування «білої скриньки», в якому тестер взаємодіє з вихідним кодом і може перевірити внутрішні шляхи виконання; і «сіра скринька», поєднання підходів із певним знанням внутрішньої структури.

Тестування можна розділити на різні рівні, наприклад модульне тестування, яке перевіряє окремі компоненти; інтеграційне тестування, яке перевіряє взаємодію між різними модулями; системне тестування, яке перевіряє систему в цілому; і приймальні випробування, що виконуються замовником або кінцевими користувачами.

Залежно від мети тестування розрізняють функціональне, нефункціональне, регресійне та ін. У зв’язку з обмеженими ресурсами та специфікою вимог бакалаврської роботи значну увагу було приділено ручному

проведенню функціонального тестування системи за методом «чорної скриньки». Причиною вибору ручного тестування є відсутність необхідності впроваджувати автоматизовані тести для порівняно обмеженої кількості функціональних можливостей, на застосунок до вимоги перевірки зручності використання інтерфейсу користувача.

Системне тестування використовувалося для тестування всієї функціональності вебзастосунку в цілому, тим самим імітуючи реальні сценарії взаємодії користувача. Функціональне тестування стосувалося перевірки належного виконання основних функцій, таких як реєстрація, вхід, пошук, список бажань і керування профілем. Використання підходу «чорної скриньки» просто означало, що тестування проводилося виключно на рівні інтерфейсу користувача, а не через вихідний код, а шляхом покладання на очікувану поведінку системи. Цілі етапу тестування включали забезпечення належної реалізації процесів реєстрації та автентифікації, перевірку ефективності та точності функції пошуку, перевірку функціональної ефективності функції списку бажань, перевірку можливості перегляду та редагування деталей профілю, перевірку точності обробки помилок і процесів перевірки даних, а також оцінку загальної міцності та простоти використання інтерфейсу.

Було проведено тестування системи ручним методом тестування. Першим було перевірено відображення сторінки реєстрації та доступність усіх її елементів.

В центрі екрану знаходиться форма, яка містить три текстові поля для введення: ім’я користувача, email та пароль. Нижче розміщено кнопку «Зареєструватися», яка активує процес збереження даних після перевірки введеної інформації. У верхній частині форми розташоване посилання «На головну», яке дозволяє швидко повернутись на головну сторінку сайту. У нижній частині форми є посилання «Увійдіть тут» для переходу на сторінку авторизації, якщо користувач вже має обліковий запис.

На рисунку 4.1 зображено інтерфейс сторінки реєстрації.

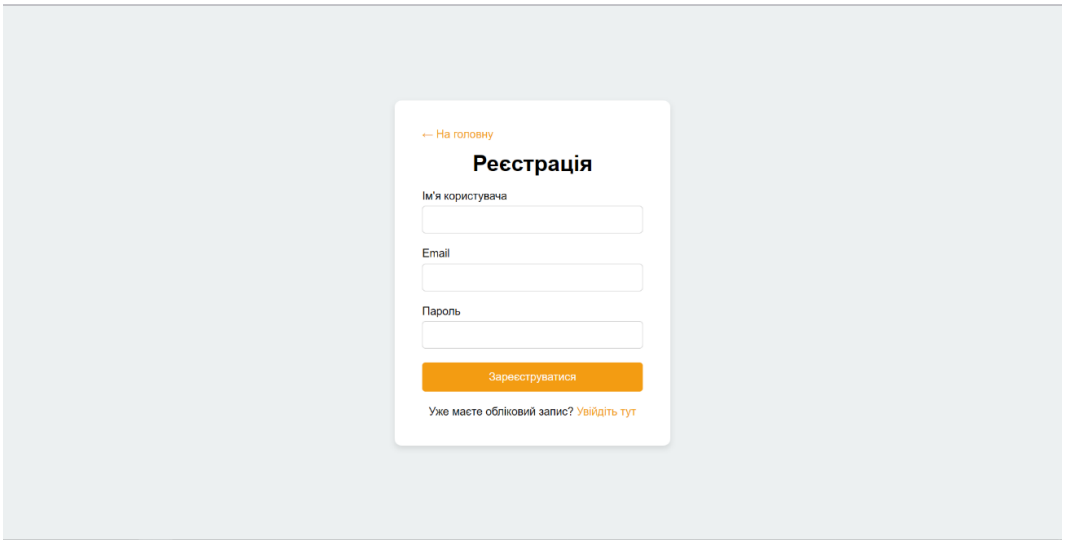

Рисунок 4.1 – Реєстраційна форма

Далі було проведено тестування валідності введеного пароля на сторінці реєстрації. Усі поля форми були заповнені та пароль, який складається лише з одного символу. Після натискання на кнопку «Зареєструватися» система відобразила повідомлення про помилку у верхній частині сторінки червоним кольором «Пароль повинен містити не менше 6 символів». На рисунку 4.2 показано сповіщення з системи.

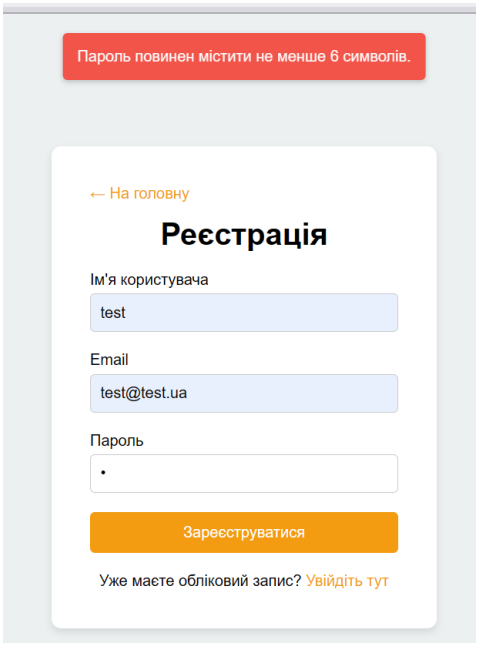

Рисунок 4.2 – Сповіщення про недостатню кількість символів в паролі

Далі було введено коректні значення у поля «Ім’я користувача», «Email» та «Пароль», після чого натиснуто на кнопку «Зареєструватися». У випадку, якщо користувач з введеним email вже існує в системі, буде відображено інформативного повідомлення про помилку, що ілюструється червоним блоком з текстом «Користувач з таким email вже існує» Це свідчить про коректну обробку спроби повторної реєстрації з існуючим email. На рисунку 4.3 наведено сповіщення про помилку.

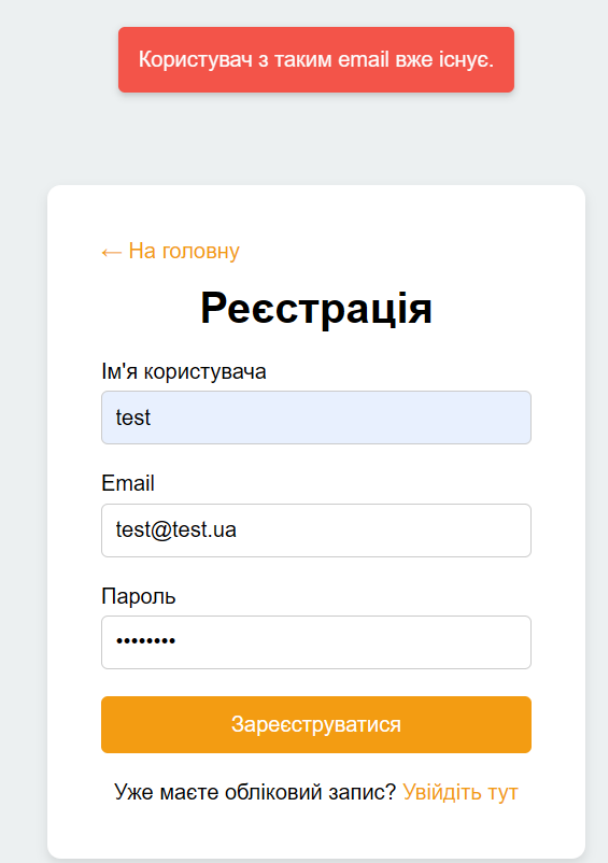

Рисунок 4.3 – Сповіщення про зареєстрований email

Наступним кроком було введення email та паролю у відповідні поля форми входу невалідними даними. В результаті цих дій, у верхній частині екрану з’явилося повідомлення «Невірний email або пароль», що вказує на те, що введені облікові дані не є коректними для входу в систему. На рисунку 4.4 наведено сповіщення про помилку входу.

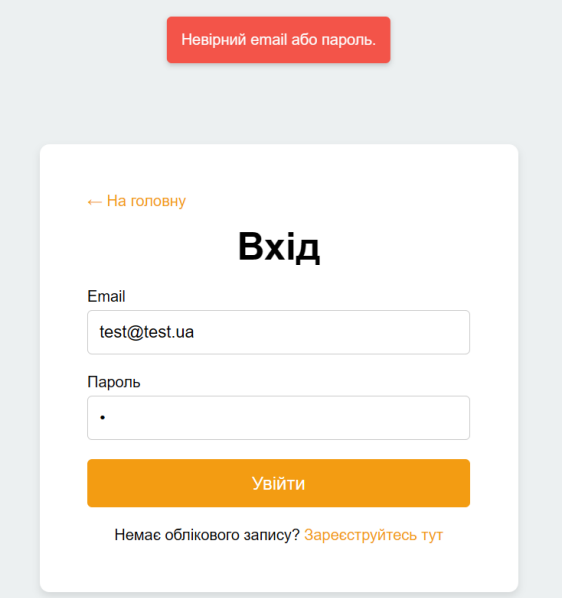

Рисунок 4.4 – Перевірка входу з невалідними даними

Після успішного входу з валідними даними було перевірено зміну пароля до облікового запису [25]. На даному етапі було відкрито модальне вікно [26] «Зміна паролю». У верхньому полі було введено невалідний поточний пароль. Після було натиснуто кнопку «Зберегти» оскільки поточний пароль було введено неправильно, було виведено повідомлення про помилку. На рисунку 4.5 показано сповіщення про помилку зміни паролю в системі.

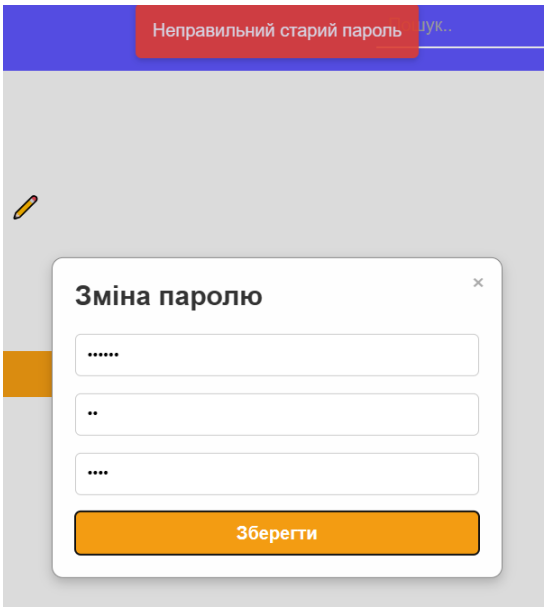

Рисунок 4.5 – Сповіщення про помилку заміни паролю

Після правильного введення даних в поля, було виведено повідомлення про успішне змінення паролю (рис. 4.6).

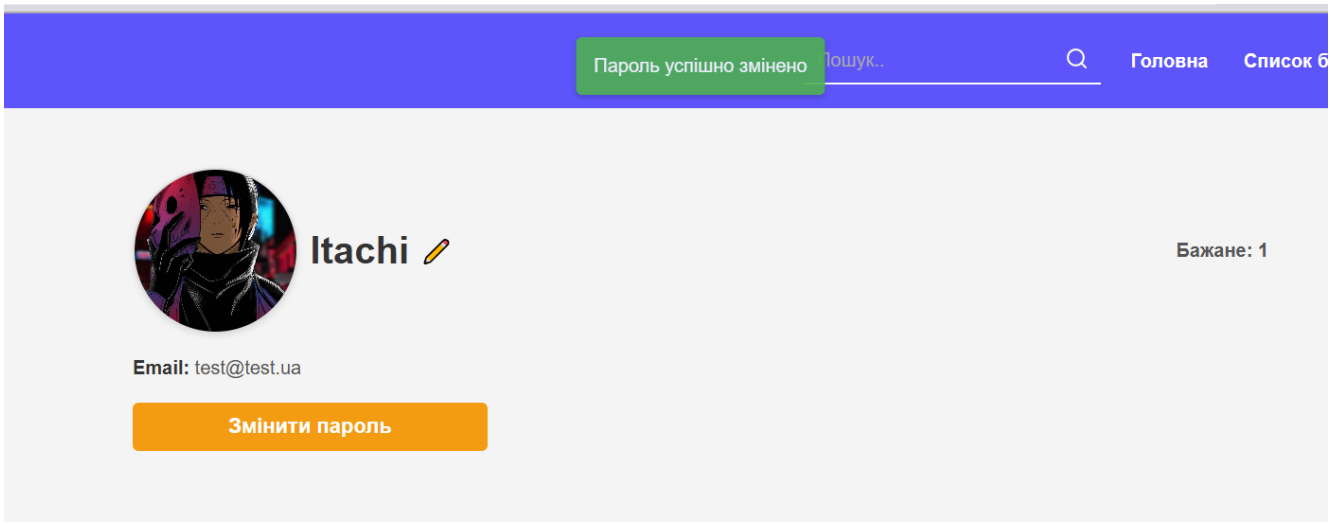

Рисунок 4.6 – Успішна зміна паролю

Наступним кроком є перевірка коректності пошуку. Було введено валідне значення. Було виведено один результат пошуку, що може свідчити про коректну роботу пошуку. На рисунку 4.7 наведено результати пошуку.

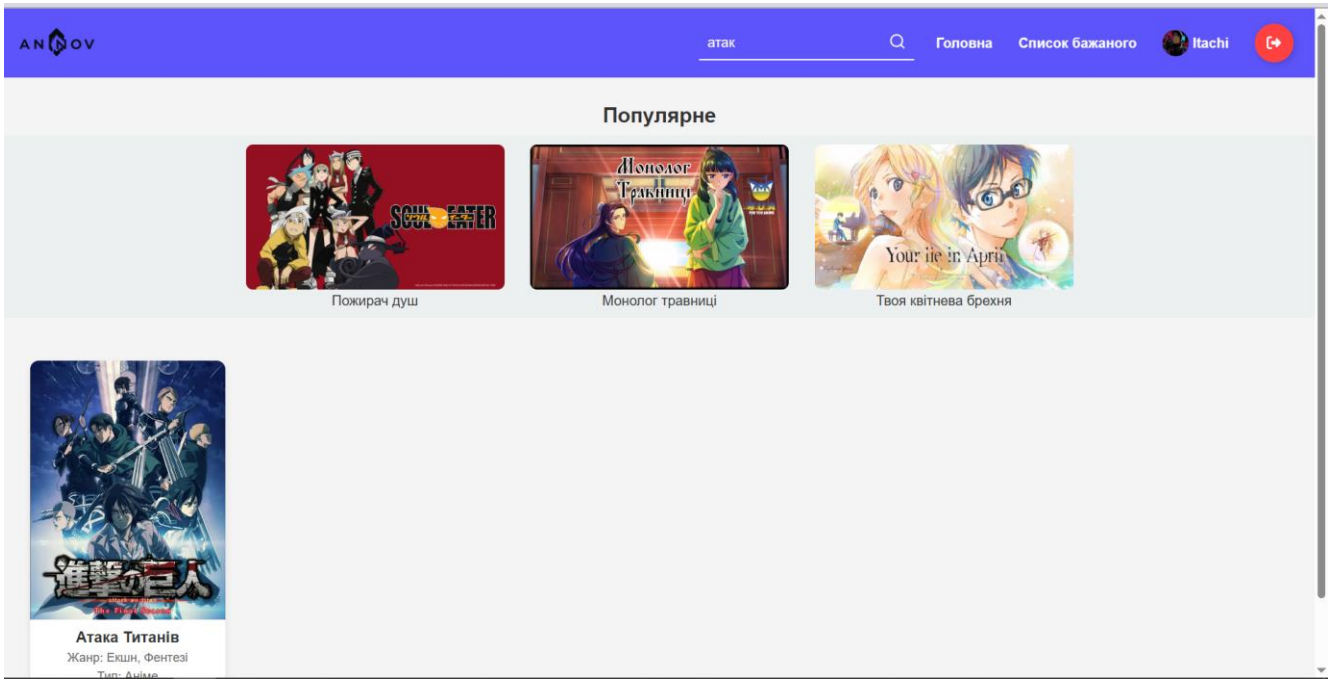

Рисунок 4.7 – Результат пошуку

Далі необхідно вписати в поле пошуку невалідні дані для перевірки сповіщення про помилку пошуку. Було введено некоректний запит та було виведено сповіщення про помилку пошуку «За вашим запитом нічого не знайдено».

На рисунку 4.8 наведено сповіщення про помилку пошуку.

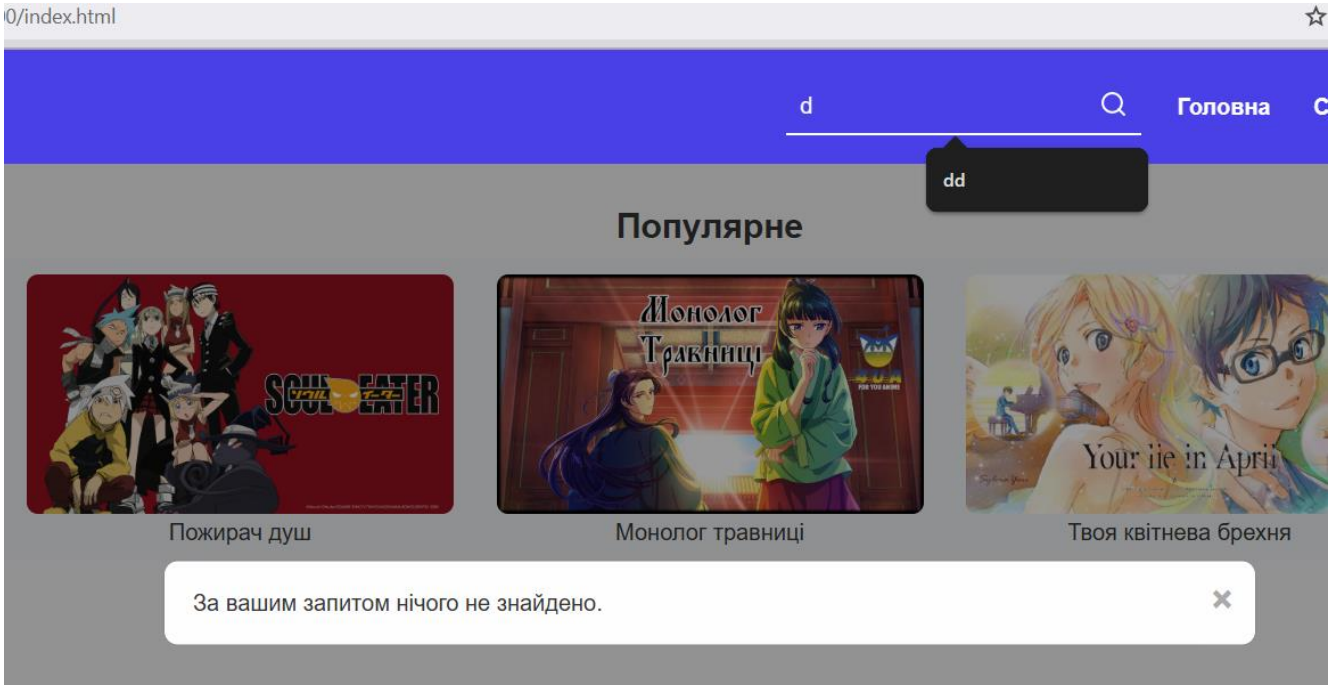

Рисунок 4.8 – Сповіщення про помилку пошуку

Отже, було проведено тестування системи за її основним функціоналом. За результатом тестування можна зробити висновок, що основні функції застосунку працюють коректно.

4.2 Розробка інструкції користувача

Згідно з інструкцією користувача, для запуску застосунку необхідно враховувати певні технічні вимоги. Детальна інформація щодо мінімальної та рекомендованої конфігурації системи представлена у таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

Вимоги до конфігурування	Рекомендовано	Мінімально
Процесор	Intel Core i7	Intel Core i3
Оперативна пам’ять	4 ГБ RAM	2 ГБ RAM
Операційна система	Windows 11	Windows 10
Швидкість підключення Інтернет	10 МБ/с	1 МБ/с

Щоб вебзастосунок працював правильно, комп’ютер користувача має відповідати мінімальним і рекомендованим технічним вимогам.

Пропонована конфігурація містить процесор Intel Core i7, 4 ГБ оперативної пам’яті, Windows 11 і швидкість Інтернету 10 МБ/с. Мінімально необхідний процесор Intel Core i3, 2 ГБ оперативної пам’яті, Windows 10 та швидкість Інтернету 1 МБ/с. Бажано використовувати найновіші версії Google Chrome, Firefox або Edge.

Щоб почати користуватися вебзастосунком, не потрібно встановлювати програмне забезпечення, користувачеві достатньо відкрити запропонований веббраузер і перейти до файлів програми. Після завантаження сторінки користувач побачить головну сторінку застосунку. Вгорі розміщено заголовок сайту, який містить логотип, основні навігаційні посилання і вікно пошуку для пошуку аніме. Також частиною заголовка є кнопки «Увійти» та «Реєстрація» для гостей або ім’я користувача та аватар і кнопка «Вийти» для тих, хто вже ввійшов у систему.

Під заголовком зазвичай є розділ «Популярні», який відображається у вигляді інтерактивної каруселі. Карусель відображає обкладинки та назви кількох популярних або нещодавно доданих аніме, дозволяючи користувачеві

легко наздоганяти, що цікаво, і клацати, щоб одночасно переглянути цікавий вміст.

Середній блок головної сторінки заповнений каталогом аніме, створеним у вигляді сітки карток. За замовчуванням тут відображаються всі доступні в системі аніме. Головну сторінку сайту наведено на рисунку 4.9.

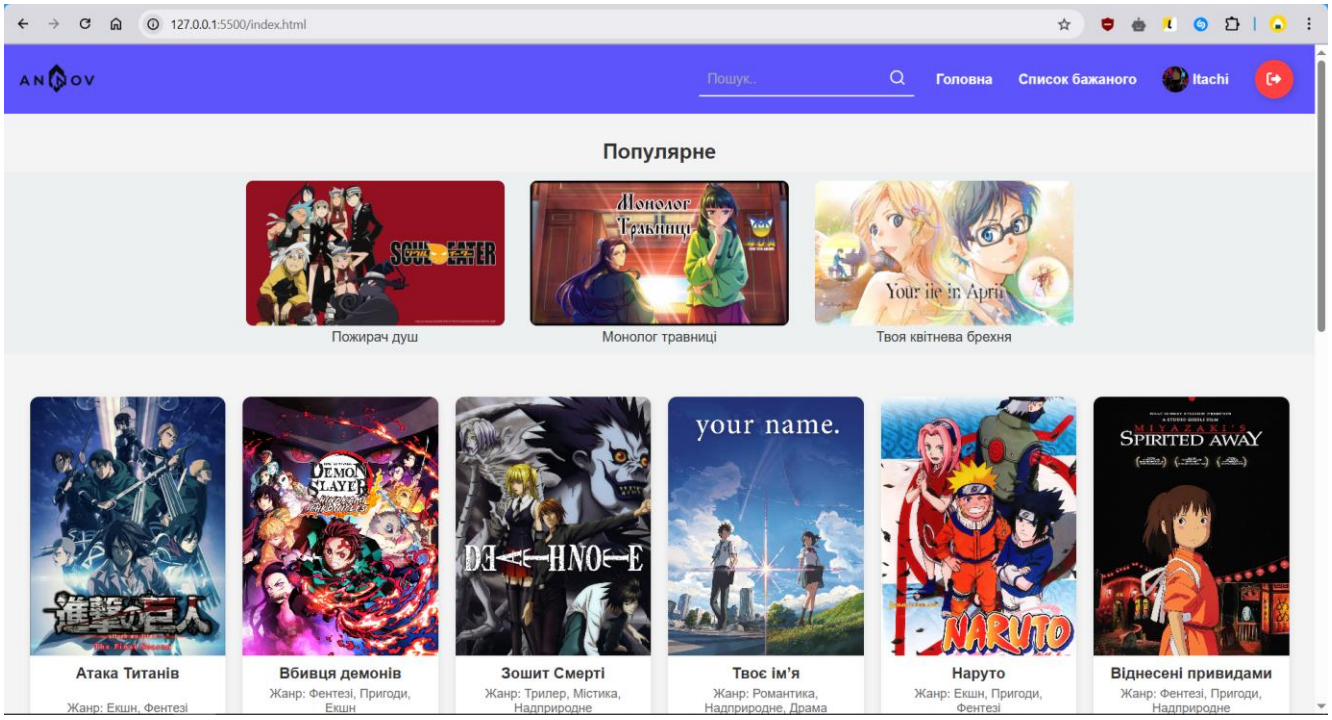

Рисунок 4.9 – Головна сторінка застосунку

Інтерфейс застосунку інтуїтивно зрозумілий. Заголовок сайту присутній на всіх сторінках і містить логотип, головне меню і посилання для керування обліковим записом. Тут також знаходиться вікно пошуку. Для неавторизованих користувачів заголовок містить кнопки «Увійти» та «Зареєструватися». Для авторизованих користувачів є аватар, логін і кнопка «Вийти».

Анімовані фільми представлено у вигляді карток. Кожна картка містить постер, назву, яка є посиланням на сторінку з інформацією, коротку інформацію та піктограму серця для списку бажань. Спливаючі вікна використовуються для певних операцій. Система також надає зворотний зв'язок у вигляді коротких повідомлень у верхній частині сторінки. Ці повідомлення повідомляють вам про

успішність операції. Щоб мати можливість використовувати особисті функції, такі як список бажань потрібно мати обліковий запис.

На сторінці профілю відображаються дані, такі як ім’я, адреса електронної пошти, аватар і кількість елементів у списку бажань. Тут користувач може змінити своє ім’я, натиснувши значок редагування поруч із ним і заповнивши модальне спливаюче вікно новим ім’ям. Змінити аватар можна, клацнувши на поточному аватарі та вибравши новий на локальному комп’ютері. Щоб змінити пароль, користувач натискає опцію зміни пароля, яка відкриває модальне вікно, де потрібно ввести старий пароль, новий пароль і підтвердження.

Застосунок дає можливість здійснювати пошук, ввівши заголовок або частину заголовка в поле пошуку у верхній частині сайту. Під час входу список анімованих фільмів, який відображається на сторінці, динамічно фільтрується та дає лише релевантні результати. Пошук не враховує регістр. Якщо збігів немає, система повідомить вас про це. Очищення поля пошуку повертає весь каталог. Користувач може ознайомитися з каталогом, прокрутивши сторінку. Щоб отримати більш конкретну інформацію про анімований фільм.

4.3 Висновки

У четвертому розділі було проведено тестування модулів вебзастосунку для пошуку та класифікації анімованих фільмів. Перевірено коректність обробки запитів, а також функціонал авторизації, зміни профілю, додавання анімованих фільмів до списку бажаного. Також було розроблено коротку інструкцію користувача щодо початку роботи із застосунком.

ВИСНОВКИ

У бакалаврській кваліфікаційній роботі було розроблено методи та програмні засоби для пошуку та класифікації анімованих фільмів та новел з використанням вебтехнологій.

Проведено аналіз сучасних підходів до розробки вебзастосунків для роботи з медіаконтентом. Обґрунтовано необхідність розробки вебзастосунку для ефективного пошуку та класифікації анімованих фільмів та новел з метою підвищення зручності користувача.

Вперше запропоновано метод автоматизованої класифікації анімованих фільмів та новел на основі їх метаданих для підвищення точності та релевантності результатів пошуку.

Вперше запропоновано метод динамічної генерації та відображення контенту вебзастосунку з використанням DOM-маніпуляцій, що забезпечує гнучкість інтерфейсу та ефективне оновлення інформації про анімовані фільми та новели.

Вперше запропоновано метод персоналізації досвіду користувача шляхом інтеграції функціоналу списку бажаного, реалізованого за допомогою локального сховища браузера для збереження вподобань користувачів.

Практичне значення отриманих результатів полягає в тому, що на основі теоретичних досліджень розроблено алгоритми та програмні засоби для вебзастосунку, що дозволяє користувачам ефективно знаходити та класифікувати анімовані фільми та новели.

Розроблено методи відображення інформації про медіаконтент та його інтерактивної взаємодії з користувачем, що дозволяють з максимальною зручністю знаходити та управляти обраним контентом.

На основі проведених у БКР досліджень розроблено алгоритми та програмні засоби для вебзастосунку пошуку та класифікації анімованих фільмів та новел.

У результаті виконання поставленої мети – підвищення точності й ефективності пошуку анімаційних фільмів – було розроблено та впроваджено вебзастосунок із модулями реєстрації, фільтрації та додавання до списку бажаного. Реалізація системи дозволила усунути основні недоліки існуючих платформ, а саме: обмеженість категоризації, відсутність персоналізованих рекомендацій і незручність пошуку.

Тестування розробленого вебзастосунку підтвердило його працездатність, відповідність заявленим функціональним вимогам та зручність для кінцевих користувачів. Це свідчить про повне виконання поставлених у вступі завдань.

Ключові слова: анімовані фільми, аніме, легкі новели, пошук, фільтрація, класифікація, JavaScript.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Anime? Everything You Need To Know [Електронний ресурс] – Режим доступу до ресурсу: https://www.nfi.edu/what-is-anime/ (дата звернення: 08.05.2025)

2. Що таке дизайн графічного інтерфейсу користувача GUI? [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/what-is-graphical-user-interface-design/ (дата звернення: 15.03.2025)

3. Гандзюк В.С. Вебплатформа для пошуку та рекомендації анімаційного контенту: архітектура та функціональність / В.С. Гандзюк, О.В. Карась / Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2025) : збірник доповідей [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24275/20115

4. Гандзюк В.С. Інтеграція функціоналу сповіщень у реальному часі в вебзастосунку / В.С. Гандзюк, О.В. Карась / Матеріали Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» : збірник доповідей [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/24924/20588

5. Search Mechanism - an overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/search-mechanism (дата звернення: 08.05.2025)

6. Що таке літературний піджанр? [Електронний ресурс] – Режим доступу до ресурсу: https://pole.liberty.cx.ua/gromadyanam/shho-take-literaturniy-pidzhanr.html (дата звернення: 10.05.2025)

7. Як працює система рекомендацій контенту? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mgid.com/uk/blog/yak-praczyuye-sistema-rekomendaczij-kontentu (дата звернення: 10.05.2025)

8. MyAnimeList.net – Anime and Manga Database and Community [Електронний ресурс] – Режим доступу до ресурсу: https://myanimelist.net/ (дата звернення: 11.05.2025)

9. AniList: Track, Discover, Share Anime & Manga [Електронний ресурс] – Режим доступу до ресурсу: https://anilist.co/ (дата звернення: 11.05.2025)

10. AniTube – аніме онлайн українською [Електронний ресурс] – Режим доступу до ресурсу: https://anitube.in.ua/ (дата звернення: 12.05.2025)

11. Дивитися аніме онлайн українською в хорошій HD якості [Електронний ресурс] – Режим доступу до ресурсу: https://eneyida.tv/anime/ (дата звернення: 12.05.2025)

12. UASerials – серіали та фільми українською мовою онлайн [Електронний ресурс] – Режим доступу до ресурсу: https://uaserials.pro/ (дата звернення: 12.05.2025)

13. Aniplay - Watch Anime Online [Електронний ресурс] – Режим доступу до ресурсу: https://aniplaynow.live/ (дата звернення: 12.05.2025)

14. Eloquent JavaScript, 4th Edition: A Modern Introduction to Programming / Marijn Haverbeke – San Francisco : No Starch Press, 2024 – 456 p.

15. What Is HTML? Hypertext Markup Language Basics [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com/tutorials/what-is-html (дата звернення: 13.05.2025)

16. CSS Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/css/css_intro.asp (дата звернення: 13.05.2025)

17. Introducing Types of UML Diagrams | Lucidchart Blog [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/blog/types-of-UML-diagrams (дата звернення: 16.05.2025)

18. draw.io – Diagrams for Confluence and Jira – draw.io [Електронний ресурс] – Режим доступу до ресурсу: https://app.diagrams.net/ (дата звернення: 17.05.2025)

19. What is browser? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/browser (дата звернення: 17.05.2025)

20. CSS Flexbox Layout Guide [Електронний ресурс] – Режим доступу до ресурсу: https://css-tricks.com/snippets/css/a-guide-to-flexbox/ (дата звернення: 18.05.2025)

21. Basic concepts of grid layout - MDN Web Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout (дата звернення: 18.05.2025)

22. Що таке DOM? [Електронний ресурс] – Режим доступу до ресурсу: https://tseivo.com/b/memecode/t/lrpjlozzyn (дата звернення: 20.05.2025)

23. Window localStorage Property [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/jsref/prop_win_localstorage.asp (дата звернення: 21.05.2025)

24. Team Guide to Software Testability: Better software through greater testability / Ash Winter, Rob Meaney – California : Conflux Books, 2021. – 188 p.

25. Що таке обліковий запис. Навіщо він потрібний [Електронний ресурс] – Режим доступу до ресурсу: https://lvivservice.com.ua/sysadmin/chto-takoe-akkaunt-dlya-chego-on-nuzhen/index.htm?srsltid=AfmBOorAiJat2ULoJe8DWyq6ax2ZWgx_bDPoocyDhLajKdmams7Azsl_ (дата звернення: 22.05.2025)

26. Модальне вікно на сайті та в застосунку: як зробити [Електронний ресурс] – Режим доступу до ресурсу: https://sendpulse.ua/blog/modal-window (дата звернення: 24.05.2025)

ДОДАТКИ

58

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
«25» 03 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка вебзастосунку для пошуку та класифікації анімованих фільмів і новел» за спеціальністю
121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
д-р. філос., асист. каф. ПЗ О. В. Карась
«24» 03 2025 р.

Виконав:
студент гр.ЗПІ-21б В.С. Гандзюк
«24» 03 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка вебзастосунку для пошуку та класифікації анімованих фільмів і новел». Галузь застосування – мультимедійні сервіси.

2. Підстава для розробки

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки

Метою дослідження є підвищення точності та ефективності пошуку анімованих фільмів шляхом розробки вебсистеми з модулями реєстрації та авторизації, фільтрації контенту, користувацького інтерфейсу та збереження у список бажаного, що забезпечує зручний доступ до впорядкованої та персоналізованої інформації.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Eloquent JavaScript, 4th Edition: A Modern Introduction to Programming / Marijn Haverbeke – San Francisco : No Starch Press, 2024 – 456 p.

5. Технічні вимоги

Вихідні дані до роботи: типова роздільна здатність екрана 1920×1080 px, сучасні веббраузери. Дані про анімовані фільми та новели подаються у форматі масиву об’єктів JavaScript з параметрами: назва, рік, жанр, опис, тип, студія, обкладинка.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

1. Пояснювальна записка до БКР.
2. Технічне завдання.
3. Лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з\п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз підходу до пошуку та класифікації анімованих фільмів та новел	25.03.25 – 07.04.25
2	Розробка моделі та алгоритмів програмного продукту	07.04.25 – 20.04.25
3	Розробка програмних модулів вебсистеми пошуку та класифікації анімованих фільмів	20.04.25 – 03.05.25
4	Тестування програмного застосунку	03.05.25 – 16.05.25
5	Оформлення матеріалів до захисту БКР	16.05.25– 30.05.25

10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

62

Додаток Б – Протокол перевірки на плагіат

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка вебзастосунку для пошуку та класифікації анімованих фільмів і новел

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ, ЗПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,8 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

(прізвище, ініціали, посада) (підпис)

(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Карась О.В. д-р. філос., асист. каф. ПЗ
(прізвище, ініціали, посада)

Здобувач (підпис) Гандзюк В.С
(прізвище, ініціали)

Додаток В – Лістинг програмного забезпечення

content.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    color: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
}

.card-info-page {
    width: 90%;
    max-width: 1200px;
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    display: flex; /* Додаємо flex для розміщення елементів поруч */
    align-items: flex-start; /* Вирівнюємо елементи по верхньому краю */
    justify-content: space-between; /* Розміщуємо простір між основною інформацією та трейлером */
    padding: 40px; /* Збільшуємо загальний відступ */
}

.card-details {
    flex: 1; /* Основна інформація займає доступний простір */
    display: flex;
    gap: 40px; /* Додаємо простір між зображенням та текстом */
}

.card-image {
    flex: 0 0 40%; /* Трохи збільшуємо для балансу */
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.card-image img {
    display: block;
    width: 100%;
    height: 100%;
    object-fit: cover;
}
```

```
.card-text {
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}

.card-title {
    font-size: 2.5rem; /* Трохи збільшуємо заголовок */
    color: #2c3e50;
    margin-bottom: 20px; /* Збільшуємо відступ під заголовком */
}

.card-description {
    color: #666;
    line-height: 1.7; /* Збільшуємо міжрядковий інтервал */
    margin-bottom: 30px; /* Збільшуємо відступ під описом */
}

.details-section h2 {
    font-size: 1.8rem; /* Трохи збільшуємо підзаголовок */
    color: #2c3e50;
    margin-top: 0;
    margin-bottom: 20px; /* Збільшуємо відступ під підзаголовком */
}

.details-list {
    list-style: none;
    padding: 0;
}

.details-list li {
    margin-bottom: 15px; /* Збільшуємо відступ між елементами списку */
    color: #555;
    font-size: 1.1rem; /* Трохи збільшуємо розмір шрифту */
}

.details-list li strong {
    font-weight: bold;
    color: #333;
    margin-right: 8px; /* Збільшуємо відступ після жирного тексту */
}

.trailer-section {
    flex: 0 0 35%; /* Трохи зменшуємо ширину трейлера */
    margin-left: 40px; /* Збільшуємо відступ зліва */
    box-sizing: border-box; /* Враховуємо padding та border у ширину */
    display: flex; /* Вирівнюємо вміст по центру */
    flex-direction: column;
    align-items: center;
}
```

```
.trailer-section iframe {
    width: 90%; /* Забезпечуємо, щоб iframe займав всю ширину контейнера */
    height: 200px; /* Збільшуємо висоту iframe */
    display: block; /* Усуваємо зайвий простір під iframe */
    border-radius: 10%;
    margin-bottom: 20px; /* Додаємо відступ під iframe */
}

.trailer-section a {
    display: inline-block;
    padding: 12px 25px; /* Збільшуємо padding кнопки */
    background-color: #f39c12;
    color: white;
    text-decoration: none;
    border-radius: 5px;
    font-weight: bold;
    transition: background-color 0.3s ease;
    font-size: 1.1rem; /* Трохи збільшуємо розмір шрифту кнопки */
}

.trailer-section a:hover {
    background-color: #e67e22;
}

/* Адаптивність */
@media (max-width: 768px) {
    .card-info-page {
        flex-direction: column; /* Перемикаємо на вертикальне розташування на малих екранах */
        align-items: center; /* Центруємо елементи */
        padding: 30px;
    }

    .card-details {
        flex-direction: column;
        gap: 20px; /* Зменшуємо простір на малих екранах */
    }

    .card-image {
        width: 80%; /* Збільшуємо відносну ширину на малих екранах */
        max-width: none;
        margin-right: 0;
        margin-bottom: 20px;
    }

    .trailer-section {
        width: 80%; /* Трейлер займає більшу частину ширини */
        margin-left: 0;
        margin-top: 30px; /* Додаємо більший відступ зверху */
    }
}
```

content.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Інформація про картку</title>
   <link rel="stylesheet" href="content.css">
   <link rel="preconnect" href="https://fonts.googleapis.com">
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
   <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&family=Roboto:wght
@400;500&display=swap" rel="stylesheet">

</head>
<body>

   <div class="card-info-page">
      </div>

<script src="content.js"></script>
<script src="header.js"></script>

</body>
</html>
```

content.js

```
const contentData = {
   1: {
      title: "Атака титанів",
      type: "Аніме",
      genre: "Бойовик, Фентезі",
      director: "Tetsurō Araki",
      description: "Людство бореться за виживання проти гігантських гуманоїдних істот.",
      image: "image/aot.png",
      trailer: '<iframe width="560" height="315" src="https://www.youtube.com/embed/LV-
nazLVmgo?si=a0-e1K2fMz4HO0Pr" title="YouTube video player" frameborder="0"
allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture;
web-share" referrerpolicy="strict-origin-when-cross-origin" allowfullscreen></iframe>',
      episodes: "87",
      status: "Завершено",
      year: "2013-2023",
      studio: "MAPPA, Wit Studio"
      },
      2: {
      title: "Клинок, що розтинає демонів",
      type: "Аніме",
```

genre: "Бойовик, Надприродне",
director: "Haruo Sotozaki",
description: "Хлопець стає винищувачем демонів, щоб помститися за свою родину та вилікувати свою сестру.",
image: "image/demonS.png",
trailer: "https://www.youtube.com/watch?v=tLn895P5jfU",
episodes: "44",
status: "Триває",
year: "2019-теперішній час",
studio: "ufotable"
},
3: {
title: "Зошит смерті",
type: "Аніме",
genre: "Трилер, Детектив",
director: "Tetsurō Araki",
description: "Студент знаходить зошит, який дозволяє йому вбивати будь-кого.",
image: "image/death-note.png"
},
4: {
title: "Твоє ім'я",
type: "Аніме",
genre: "Романтика, Надприродне",
director: "Makoto Shinkai",
description: "Двоє підлітків загадково міняються тілами та формують унікальний зв'язок.",
image: "image/your-name.png"
},
5: {
title: "Наруто",
type: "Аніме",
genre: "Бойовик, Пригоди",
director: "Hayato Date",
description: "Юний ніндзя прагне стати лідером свого села.",
image: "image/naruto.png"
},
6: {
title: "Віднесені привидами",
type: "Аніме",
genre: "Фентезі, Пригоди",
director: "Hayao Miyazaki",
description: "Дівчинка потрапляє у світ духів, щоб врятувати своїх батьків і знайти дорогу додому.",
image: "image/spirited-away.png"
},
7: {
title: "Ван Піс",
type: "Аніме",
genre: "Пригоди, Фентезі",
director: "Konosuke Uda",
description: "Хлопець з гумовими здібностями пливе, щоб стати Королем Піратів.",

```
        image: "image/onepiece.png"
        },
        8: {
        title: "Сталевий алхімік: Братство",
        type: "Аніме",
        genre: "Бойовик, Драма",
        director: "Yasuhiro Irie",
         description: "Двоє братів використовують алхімію, щоб спробувати відновити те, що вони втратили.",
         image: "image/fma.png"
         },
        9: {
         title: "Токійський гуль",
        type: "Аніме",
        genre: "Жахи, Надприродне",
       director: "Shuhei Morita",
        description: "Студент коледжу стає напівгулем і бореться зі своєю новою ідентичністю.",
        image: "image/tokyoghoul.png"
        },
        10: {
        title: "Магічна битва",
        type: "Аніме",
        genre: "Бойовик, Надприродне",
        director: "Sunghoo Park",
       description: "Студент бореться з прокляттями силою особливого духа.",
         image: "image/jujutsukaisen.png"
         },
         11: {
        title: "Сім'я шпигуна",
        type: "Аніме",
       genre: "Бойовик, Комедія, Повсякденність",
        director: "Kazuhiro Furuhashi",
         description: "Шпигун створює фальшиву сім'ю для місії, не знаючи, що його дружина — вбивця, а дочка — телепат.",
        image: "image/spyxfamily.png"
       },
         12: {
       title: "Створений у безодні",
        type: "Аніме",
        genre: "Пригоди, Фентезі, Драма",
        director: "Masayuki Kojima",
        description: "Юна дівчина спускається в таємничу безодню, щоб знайти свою зниклу матір, розкриваючи темні таємниці на своєму шляху.",
        image: "image/madeinabyss.png"
       }
};

function getContentId() {
   const params = new URLSearchParams(window.location.search);
   return params.get("id");
}
```

```
function loadContent() {
   const contentId = getContentId();
   const cardInfoPage = document.querySelector('.card-info-page');

   if (contentData[contentId]) {
      const content = contentData[contentId];

      cardInfoPage.innerHTML = `
         <div class="card-details">
            <div class="card-image">
               <img src="${content.image}" alt="${content.title}">
            </div>
            <div class="card-text">
               <h1 class="card-title">${content.title}</h1>
               <p class="card-description">${content.description}</p>

               <div class="details-section">
                  <h2>Деталі</h2>
                  <ul class="details-list">
                     <li><strong>Тип:</strong> <span>${content.type}</span></li>
                     <li><strong>Жанр:</strong> <span>${content.genre}</span></li>
                     <li><strong>Режисер:</strong> <span>${content.director}</span></li>
                     ${content.year ? `<li><strong>Рік виходу:</strong>
<span>${content.year}</span></li>` : ''}
                     ${content.episodes ? `<li><strong>Кількість епізодів:</strong>
<span>${content.episodes}</span></li>` : ''}
                     <li><strong>Статус:</strong> <span>${content.status}</span></li>
                     ${content.studio ? `<li><strong>Студія:</strong>
<span>${content.studio}</span></li>` : ''}
                  </ul>
               </div>
            </div>
         </div>
         ${content.trailer ? `
         <div class="trailer-section">
            ${content.trailer}
         </div>` : ''}
      `;
   } else {
      cardInfoPage.innerHTML = "<h1>Контент не знайдено</h1>";
   }
}

window.onload = loadContent;
```

header.js

```
function checkAuthStatus() {
   const user = JSON.parse(localStorage.getItem('user'));
```

```
return `
<header>
  <nav class="navbar">
    <div class="logo">
      <a href="index.html"><img src="image/logo.png" alt="Logo"></a>
    </div>
    <ul class="nav-links" id="nav-links">

      <div class="search-box">
        <input type="text" id="search-input" placeholder="Пошук.." />
        <div class="search-icon">
          <svg viewBox="0 0 24 24" width="20" height="20" fill="none" stroke="white" stroke-width="2" stroke-linecap="round" stroke-linejoin="round">
          <circle cx="11" cy="11" r="8"></circle>
          <line x1="21" y1="21" x2="16.65" y2="16.65"></line>
          </svg>
        </div>
      </div>
      <p id="searchMessage" class="search-message" style="display: none; color: white;">Введіть хоча б 3 символи для пошуку</p>


      <li><a href="index.html">Головна</a></li>
      ${user ? `
        <li><a href="wishlist.html">Список бажаного</a></li>
        <li>
          <a href="profile.html" style="display: flex; align-items: center;">
            ${user.avatar ? `<img src="${user.avatar}" alt="Аватар" style="width: 30px; height: 30px; border-radius: 50%; margin-right: 5px;">` : ''}
            ${user.username}
          </a>
        </li>
        <li><button  class="Btn" id="logout-btn">

          <div class="sign"><svg viewBox="0 0 512 512"><path d="M377.9 105.9L500.7 228.7c7.2 7.2 11.3 17.1 11.3 27.3s-4.1 20.1-11.3 27.3L377.9 406.1c-6.4 6.4-15 9.9-24 9.9c-18.7 0-33.9-15.2-33.9-33.9l0-62.1-128 0c-17.7 0-32-14.3-32-32l0-64c0-17.7 14.3-32 32-32l128 0 0-62.1c0-18.7 15.2-33.9 33.9-33.9c9 0 17.6 3.6 24 9.9zM160 96L96 96c-17.7 0-32 14.3-32 32l0 256c0 17.7 14.3 32 32 32l64 0c17.7 0 32 14.3 32 32s-14.3 32-32 32l-64 0c-53 0-96-43-96-96L0 128C0 75 43 32 96 32l64 0c17.7 0 32 14.3 32 32s-14.3 32-32 32z"></path></svg></div>

          <div class="text">Вийти</div>
        </button></li>
      ` : `
        <li><a href="wishlist.html">Список бажаного</a></li>
        <li><button onclick="window.location.href='login.html'" class="Btn-login">

          <div class="sign"><svg viewBox="0 0 512 512"><path d="M217.9 105.9L340.7 228.7c7.2 7.2 11.3 17.1 11.3 27.3s-4.1 20.1-11.3 27.3L217.9 406.1c-6.4 6.4-15 9.9-24 9.9c-18.7 0-33.9-15.2-33.9-33.9l0-62.1L32 320c-17.7 0-32-14.3-32-32l0-64c0-17.7 14.3-32 32-32l128 0 0-
```

```
62.1c0-18.7 15.2-33.9 33.9-33.9c9 0 17.6 3.6 24 9.9zM352 416l64 0c17.7 0 32-14.3 32-32l0-256c0-17.7-14.3-32-32-32l-64 0c-17.7 0-32-14.3-32-32s14.3-32 32-32l64 0c53 0 96 43 96 96l0 256c0 53-43 96-96 96l-64 0c-17.7 0-32-14.3-32-32s14.3-32 32-32z"></path></svg></div>

                <div class="text">Увійти</div>
              </button></li>
            `}
          </ul>
        </nav>
    </header>
    `;
}

function loadHeader() {
    const headerContainer = document.getElementById('header-placeholder');
    if (!headerContainer) return;

    headerContainer.innerHTML = checkAuthStatus();

    const logoutBtn = document.getElementById('logout-btn');
    if (logoutBtn) {
        logoutBtn.addEventListener('click', (e) => {
            e.preventDefault();
            localStorage.removeItem('user');
            window.location.href = 'index.html';
        });
    }
}

document.addEventListener('DOMContentLoaded', loadHeader);
```

index.css

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

body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  color: #333;
}

/* Navigation Bar (залишено без змін) */
header {
  background-color: #362cfccb;
  padding: 10px;
  color: white;
```

```
  position: sticky;
  top: 0;
  z-index: 1000;
}

nav {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin: 0 auto;
  width: 100%;
  position: relative;
}

.logo {
  overflow: hidden;
  width: 100px;
  display: flex;
  align-items: center;
  justify-content: center;
}
.logo img {
  max-width: 50px;
  object-fit: contain;
  transform: scale(4);
}

.popular {
margin-top: 30px;
text-align: center;
}

.menu {
  display: flex;
  align-items: center;
  margin-left: 0 auto; /* Ймовірно, тут мало бути margin: 0 auto; або видалити? */
}

ul {
  display: flex;
  list-style: none;
  align-items: center;
  /* 'ul' не має padding за замовчуванням, але додамо для безпеки */
  padding-left: 0;
}

ul li {
  margin: 0 15px;
}

ul li a {
```

```
  color: white;
  text-decoration: none;
  font-weight: bold;
}

ul li a:hover {
  color: #f39c12;
}

.navbar .nav-links li img.profile-avatar-header {
  width: 30px;
  height: 30px;
  border-radius: 50%;
  vertical-align: middle;
  margin-right: 5px;
}

/* Carousel (залишено без змін) */
#carousel {
  margin: 10px 0 30px 0;
  padding: 10px 0;
  background-color: #ecf0f1;
}

.carousel-container {
  display: flex;
  overflow: hidden;
  width: 100%;
  justify-content: center;
  position: relative;
}

.carousel-item {
  width: 100%;
  max-width: 300px;
  margin: 0 15px;
  transition: transform 0.5s ease;
  text-align: center;
}

.carousel-item img {
  width: 100%;
  border-radius: 8px;
}

/* ----- СТИЛІ ДЛЯ КАРТОК ----- */

/* Секція, що містить список карток (може бути #anime-cards або інша) */
/* Залишаємо тут тільки загальні стилі для секції, якщо потрібно */
#anime-cards { /* Якщо у вас є батьківська секція */
  padding: 20px; /* Наприклад, зовнішні відступи для всієї секції */
```

```
  margin: 10px;
}

/* Контейнер, куди JS додає картки */
#content-list {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
  gap: 20px; /* Відстань між картками */
  padding: 0; /* Забираємо внутрішній padding, якщо він був у #anime-cards */
}


/* Стилі для КОЖНОЇ окремої картки */
.anime-card {
  background-color: white;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  overflow: hidden; /* Важливо для border-radius зображення */
  text-align: center;
  transition: transform 0.3s ease, box-shadow 0.3s ease; /* Додано box-shadow до transition */
  display: flex; /* Використовуємо Flexbox для внутрішнього компонування картки */
  flex-direction: column; /* Розміщуємо елементи картки (img, h3, p, button) в стовпчик */
  height: 100%; /* Дозволяє карткам мати однакову висоту в рядку гріда */
}

.anime-card img {
  width: 100%;
  height: 300px; /* Фіксована висота зображення */
  object-fit: cover; /* Масштабує зображення, зберігаючи пропорції та обрізаючи зайве */
  aspect-ratio: 10 / 16;
  display: block;
  /* border-bottom: 2px solid #ddd; */ /* Можна прибрати, якщо не потрібна лінія */
}

.anime-card h3 {
  padding: 10px 15px 5px 15px; /* Зменшено нижній padding */
  font-size: 1.1rem; /* Трохи зменшено шрифт */
  margin: 0; /* Забираємо можливі зовнішні відступи */
  flex-grow: 1; /* Дозволяє блоку h3 займати доступний простір, щоб кнопки були внизу */
}

.anime-card h3 a {
  text-decoration: none;
  color: #333;
}
/* Додано стилі для параграфів всередині картки */
.anime-card p {
  font-size: 0.9rem;
  color: #666;
  padding: 0 15px 5px 15px; /* Невеликі відступи */
  margin: 0;
```

```
}


.anime-card:hover {
  transform: translateY(-5px); /* Зменшено ефект підйому */
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15); /* Трохи змінено тінь */
}




/* СПИСОК БАЖАНОГО КНОПКА (залишено без змін, але переміщено нижче стилів картки)
*/
.heart-wrapper {
display: flex;
justify-content: flex-end; /* Кнопка буде справа */
padding: 5px 10px 10px 10px; /* Зменшено верхній padding */
margin-top: auto; /* Притискає кнопку до низу картки, якщо .anime-card має display: flex */
}

.heart-container {
  --heart-color: rgb(255, 91, 137);
  position: relative;
  width: 30px;
  height: 30px;
  transition: .3s;
}

.heart-container .checkbox {
  position: absolute;
  width: 100%;
  height: 100%;
  opacity: 0;
  z-index: 20;
  cursor: pointer;
  top: 0;
  left: 0;
}

.heart-container .svg-container {
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.heart-container .svg-outline,
        .heart-container .svg-filled {
  fill: var(--heart-color);
```

```
  position: absolute;
}

.heart-container .svg-filled {
  animation: keyframes-svg-filled 1s;
  display: none;
}

.heart-container .svg-celebrate {
  position: absolute;
  animation: keyframes-svg-celebrate .5s;
  animation-fill-mode: forwards;
  display: none;
  stroke: var(--heart-color);
  fill: var(--heart-color);
  stroke-width: 2px;
}

.heart-container .checkbox:checked~.svg-container .svg-filled {
  display: block
}

.heart-container .checkbox:checked~.svg-container .svg-celebrate {
  display: block
}

@keyframes keyframes-svg-filled {
  0% {
    transform: scale(0);
  }

  25% {
    transform: scale(1.2);
  }

  50% {
    transform: scale(1);
    filter: brightness(1.5);
  }
}

@keyframes keyframes-svg-celebrate {
  0% {
    transform: scale(0);
  }

  50% {
    opacity: 1;
    filter: brightness(1.5);
  }
```

```
  100% {
    transform: scale(1.4);
    opacity: 0;
    display: none;
  }
}

/* Інші стилі (кнопки, пошук) залишено без змін */
/* ... (решта ваших стилів .Btn, .Btn-login, .search-box) ... */

.Btn {
  display: flex;
  align-items: center;
  justify-content: flex-start;
  width: 45px;
  height: 45px;
  border: none;
  border-radius: 50%;
  cursor: pointer;
  position: relative;
  overflow: hidden;
  transition-duration: .3s;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.199);
  background-color: rgb(255, 65, 65);
}

/* plus sign */
.sign {
  width: 100%;
  transition-duration: .3s;
  display: flex;
  align-items: center;
  justify-content: center;
}

.sign svg {
  width: 17px;
}

.sign svg path {
  fill: white;
}
/* text */
.text {
  position: absolute;
  right: 0%;
  width: 0%;
  opacity: 0;
  color: white;
  font-size: 1.2em;
  font-weight: 600;
```

```
  transition-duration: .3s;
}
/* hover effect on button width */
.Btn:hover {
  width: 125px;
  border-radius: 40px;
  transition-duration: .3s;
}

.Btn:hover .sign {
  width: 30%;
  transition-duration: .3s;
  padding-left: 20px;
}
/* hover effect button's text */
.Btn:hover .text {
  opacity: 1;
  width: 70%;
  transition-duration: .3s;
  padding-right: 10px;
}
/* button click effect*/
.Btn:active {
  transform: translate(2px ,2px);
}



/* ВХІД КНОПКА*/
.Btn-login {
  display: flex;
  align-items: center;
  justify-content: flex-start;
  width: 45px;
  height: 45px;
  border: none;
  border-radius: 50%;
  cursor: pointer;
  position: relative;
  overflow: hidden;
  transition-duration: .3s;
  box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.199);
  background-color: rgb(163, 142, 255);
}

/* plus sign */
/* .sign - вже визначено вище */

/* text */
/* .text - вже визначено вище */
```

```
/* hover effect on button width */
.Btn-login:hover {
  width: 125px;
  border-radius: 40px;
  transition-duration: .3s;
}

.Btn-login:hover .sign {
  width: 30%;
  transition-duration: .3s;
  padding-left: 20px;
}
/* hover effect button's text */
.Btn-login:hover .text {
  opacity: 1;
  width: 70%;
  transition-duration: .3s;
  padding-right: 10px;
}
/* button click effect*/
.Btn-login:active {
  transform: translate(2px ,2px);
}


.search-box {
  position: relative;
  width: 250px;
  margin: 10px;
}

.search-box input {
  width: 100%;
  padding: 10px 35px 10px 10px;
  background: transparent;
  border: none;
  border-bottom: 2px solid #fff;
  color: white;
  font-size: 16px;
  outline: none;
}

.search-box input::placeholder {
  color: #aaa;
}

.search-box .search-icon {
  position: absolute;
  right: 10px;
  top: 50%;
  transform: translateY(-50%);
```

```
  pointer-events: none; /* щоб клік ішов по інпуту */
}

.search-message {
  text-align: center;
  font-size: 14px;
  margin-top: 10px;
  color: white; /* Змінено з black на white */
  opacity: 0.7;
  display: none; /* Додано, щоб приховати за замовчуванням */
}


/* Стилі для модального контейнера */
.modal {
  display: none;
  position: fixed;
  z-index: 1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgba(0,0,0,0.4);
  /* Анімація появи (розмиття) */
  animation-name: modalFadeIn;
  animation-duration: 0.3s;
}

/* Стилі для вмісту модального вікна */
.modal-content {
  background-color: #fefefe;
  width: 50%; /* Можете змінити на потрібне значення */
  border-radius: 10px; /* Можете змінити на потрібне значення */
  padding: 20px;
  border: 1px solid #888;

  /* Центрування модального вікна */
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}

/* Стилі для кнопки закриття */
.close-button {
  color: #aaa;
  position: absolute; /* Абсолютне позиціонування всередині .modal-content */
  top: 10px; /* Відступ зверху */
  right: 15px; /* Відступ справа */
  font-size: 28px;
```

```
  font-weight: bold;
  cursor: pointer;
}

.close-button:hover,
.close-button:focus {
  color: black;
  text-decoration: none;
}

/* Анімація появи (розмиття) для всього модального вікна */
@keyframes modalFadeIn {
  from {opacity: 0;}
  to {opacity: 1;}
}
```

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Anime & Ranobe</title>
    <link rel="stylesheet" href="index.css">
</head>

<body>

    <script src="header.js"></script>
    <script src="index.js"></script>
    <script src="wishlist.js"></script>
    <script src="profile.js"></script>
    <div id="header-placeholder"></div>

    <div>
        <h2 class="popular">Популярне</h2>
    </div>

    <!-- Carousel -->
    <section id="carousel">
        <div class="carousel-container">
            <div class="carousel-item" id="item1">
                <img src="image/soul.png" alt="Carousel Image 1">
                <p>Пожирач душ</p>
            </div>
            <div class="carousel-item" id="item2">
                <img src="image/diary.png" alt="Carousel Image 2">
                <p>Монолог травниці</p>
```

```
        </div>
        <div class="carousel-item" id="item3">
          <img src="image/april.png" alt="Carousel Image 3">
          <p>Твоя квітнева брехня</p>
        </div>

      </div>
    </section>

    <section id="anime-cards">
      <div id="content-list"></div>
    </section>


    <div id="search-message-display" class="modal">
      <div class="modal-content">
        <span class="close-button">&times;</span>
        <p>За вашим запитом нічого не знайдено.</p>
      </div>
    </div>
</body>

</html>
```

index.js

```
// Ваш масив даних про контент
const contentData = [
    { id: 1, title: "Атака Титанів", type: "Аніме", genre: "Екшн, Фентезі", image: "image/aot.png" },
    { id: 2, title: "Вбивця демонів", type: "Аніме", genre: "Фентезі, Пригоди, Екшн", image: "image/demonS.png" },
    { id: 3, title: "Зошит Смерті", type: "Аніме", genre: "Трилер, Містика, Надприродне", image: "image/death-note.png" },
    { id: 4, title: "Твоє ім’я", type: "Аніме", genre: "Романтика, Надприродне, Драма", image: "image/your-name.png" },
    { id: 5, title: "Наруто", type: "Аніме", genre: "Екшн, Пригоди, Фентезі", image: "image/naruto.png" },
    { id: 6, title: "Віднесені привидами", type: "Аніме", genre: "Фентезі, Пригоди, Надприродне", image: "image/spirited-away.png" },
    { id: 7, title: "Ван Піс", type: "Аніме", genre: "Пригоди, Фентезі, Екшн", image: "image/onepiece.png" },
    { id: 8, title: "Сталевий алхімік: Братство", type: "Аніме", genre: "Екшн, Драма, Фентезі, Пригоди", image: "image/fma.png" },
    { id: 9, title: "Токійський гуль", type: "Аніме", genre: "Жахи, Надприродне, Екшн, Драма", image: "image/tokyoghoul.png" },
    { id: 10, title: "Магічна битва", type: "Аніме", genre: "Екшн, Надприродне, Фентезі", image: "image/jujutsukaisen.png" },
    { id: 11, title: "Сім’я Шпигуна", type: "Аніме", genre: "Екшн, Комедія, Повсякденність", image: "image/spyxfamily.png" },
    { id: 12, title: "Створено в Безодні", type: "Аніме", genre: "Пригоди, Фентезі, Драма, Містика", image: "image/madeinabyss.png" },
```

```
    { id: 13, title: "тест", type: "Аніме", genre: "Пригоди, Фентезі, Драма, Містика", image: "image/madeinabyss.png" }
];


/**
 * @param {Array} data - Масив об'єктів контенту для відображення.
 */

function displayContent(data) {
    const contentList = document.getElementById("content-list");
    if (!contentList) {
        console.error("Елемент #content-list не знайдено! Картки не можуть бути додані.");
        return;
    }
    contentList.innerHTML = "";

    if (data.length === 0) {
        contentList.innerHTML = "<p></p>";
        return;
    }
    data.forEach(item => {
        const card = document.createElement("div");
        card.classList.add("anime-card");

        card.dataset.id = item.id;
        card.dataset.title = item.title;
        card.dataset.image = item.image;

        card.innerHTML = `
            <img src="${item.image}" alt="${item.title}">
            <h3><a href="content.html?id=${item.id}">${item.title}</a></h3>
            <p class="genre">Жанр: ${item.genre}</p>
            <p class="type">Тип: ${item.type}</p>
            `;
        contentList.appendChild(card);
    });

    initializeWishlistButtons();
    updateWishlistCount(); // Оновлюємо кількість бажаного після відображення контенту
}

function updateWishlistCount() {
    const currentUser = JSON.parse(localStorage.getItem("user"));
    const wishlistCountElement = document.getElementById("wishlist-item-count"); // Припускаємо, що у вас є елемент з id "wishlist-count"

    if (currentUser && wishlistCountElement) {
        const wishlist = JSON.parse(localStorage.getItem("wishlist_" + currentUser.email)) || [];
        wishlistCountElement.textContent = wishlist.length;
    } else if (wishlistCountElement) {
```

```
        wishlistCountElement.textContent = 0; // Якщо користувач не авторизований або немає бажаного
    }
}

function initializeWishlistButtons() {
    const currentUser = JSON.parse(localStorage.getItem("user"));
    if (!currentUser) {
        console.warn("Користувач не авторизований, функціонал Wishlist може бути недоступний.");
        return;
    }

    let wishlist = JSON.parse(localStorage.getItem("wishlist_" + currentUser.email)) || [];

    document.querySelectorAll("#content-list .anime-card").forEach(card => {
        if (card.querySelector('.heart-wrapper')) {
            return;
        }

        const id = card.dataset.id;
        const title = card.dataset.title;
        const image = card.dataset.image;

        if (!id || !title || !image) {
            console.warn("Картка не має всіх необхідних data-атрибутів (id, title, image):", card);
            return;
        }

        const heartWrapper = document.createElement("div");
        heartWrapper.classList.add("heart-wrapper");

        const heart = document.createElement("div");
        heart.classList.add("heart-container");
        heart.setAttribute("title", "Додати до бажаного");

        heart.innerHTML = `
            <input type="checkbox" class="checkbox">
            <div class="svg-container">
                <svg viewBox="0 0 24 24" class="svg-outline" xmlns="http://www.w3.org/2000/svg">
                    <path d="M17.5,1.917a6.4,6.4,0,0,0-5.5,3.3,6.4,6.4,0,0,0-5.5-3.3A6.8,6.8,0,0,0,0,8.967c0,4.547,4.786,9.513,8.8,12.88a4.974,4.974,0,0,0,6.4,0C19.214,18.48,24,13.514,24,8.967A6.8,6.8,0,0,0,17.5,1.917Zm-3.585,18.4a2.973,2.973,0,0,1-3.83,0C4.947,16.006,2,11.87,2,8.967a4.8,4.8,0,0,1,4.5-5.05A4.8,4.8,0,0,1,11,8.967a1,1,0,0,0,2,0,4.8,4.8,0,0,1,4.5-5.05A4.8,4.8,0,0,1,22,8.967C22,11.87,19.053,16.006,13.915,20.313Z"></path>
                </svg>
                <svg viewBox="0 0 24 24" class="svg-filled" xmlns="http://www.w3.org/2000/svg">
                    <path d="M17.5,1.917a6.4,6.4,0,0,0-5.5,3.3,6.4,6.4,0,0,0-5.5-3.3A6.8,6.8,0,0,0,0,8.967c0,4.547,4.786,9.513,8.8,12.88a4.974,4.974,0,0,0,6.4,0C19.214,18.48,24,13.514,24,8.967A6.8,6.8,0,0,0,17.5,1.917Z"></path>
```

```
            </svg>
            <svg class="svg-celebrate" width="100" height="100"
xmlns="http://www.w3.org/2000/svg">
              <polygon points="10,10 20,20"></polygon> <polygon points="10,50
20,50"></polygon> <polygon points="20,80 30,70"></polygon> <polygon points="90,10
80,20"></polygon> <polygon points="90,50 80,50"></polygon> <polygon points="80,80
70,70"></polygon>
            </svg>
          </div>`;
    heartWrapper.appendChild(heart);



    const checkbox = heart.querySelector(".checkbox");

    if (wishlist.find(item => String(item.id) === String(id))) {
      checkbox.checked = true;
    }

    checkbox.addEventListener("change", () => {
      wishlist = JSON.parse(localStorage.getItem("wishlist_" + currentUser.email)) || [];
      const numericId = parseInt(id, 10);

      if (checkbox.checked) {
        if (!wishlist.find(item => item.id === numericId)) {
          wishlist.push({ id: numericId, title, image });
        }
      } else {
        wishlist = wishlist.filter(item => item.id !== numericId);
      }
      localStorage.setItem("wishlist_" + currentUser.email, JSON.stringify(wishlist));
      updateWishlistCount(); // Оновлюємо кількість бажаного при зміні стану кнопки
    });



    card.appendChild(heartWrapper);
  });
}


// --- Запуск при завантаженні DOM ---
document.addEventListener("DOMContentLoaded", () => {
  // Відображаємо весь контент з масиву contentData при завантаженні сторінки
  displayContent(contentData);
  updateWishlistCount(); // Оновлюємо кількість бажаного при завантаженні сторінки


  const searchInput = document.getElementById("search-input");
  const searchMessageDisplay = document.getElementById("search-message-display");
  const contentListContainer = document.getElementById("content-list");
```

```
if (searchInput && contentListContainer) {
    searchInput.addEventListener("input", () => {
        const searchTerm = searchInput.value.trim().toLowerCase();

        let filteredData;

        if (searchTerm === "") {
            filteredData = contentData;

            if (searchMessageDisplay) {
                searchMessageDisplay.style.display = "none";
            }
        } else {

            filteredData = contentData.filter(item => {

                return item.title && item.title.toLowerCase().includes(searchTerm);
            });
            if (searchMessageDisplay) {
                if (filteredData.length === 0) {
                    searchMessageDisplay.style.display = "block";
                } else {
                    searchMessageDisplay.style.display = "none";
                }
            }
        }
        displayContent(filteredData, searchTerm !== "");




        const modal = document.getElementById("search-message-display");
        const closeButton = document.querySelector(".close-button");
        function showNoResultsModal() {
            modal.style.display = "block";
        }

        function hideNoResultsModal() {
            modal.style.display = "none";
        }

        closeButton.addEventListener("click", hideNoResultsModal);

        window.addEventListener("click", function(event) {
            if (event.target == modal) {
                hideNoResultsModal();
            }
        });

    });
```

```
    }

});
```

login.css

```
/* General Reset and Styling */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: #ecf0f1;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    /* Full viewport height */
}

/* Authentication Forms Container */
.auth-container {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: 100%;
    /* Full viewport height */
    position: absolute;
    /* Makes it fill the entire screen */
}

.auth-box {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 50px;
    /* Increased padding */
    width: 100%;
    max-width: 500px;
    /* Increased max-width */
    margin: auto;
    /* Automatically adjust margins to center the form */
}

h2 {
    text-align: center;
    margin-bottom: 20px;
```

```
    font-size: 2.5rem;
    /* Increased font size */
}

.auth-form .form-group {
    margin-bottom: 20px;
}

.auth-form label {
    display: block;
    font-size: 1rem;
    margin-bottom: 5px;
}

.auth-form input {
    width: 100%;
    padding: 12px;
    /* Increased padding for input fields */
    font-size: 1.1rem;
    /* Increased font size for inputs */
    border-radius: 5px;
    border: 1px solid #ccc;
}

.auth-form input:focus {
    border-color: #f39c12;
    outline: none;
}

.auth-btn {
    width: 100%;
    padding: 14px;
    /* Increased padding for button */
    background-color: #f39c12;
    color: white;
    border: none;
    border-radius: 5px;
    font-size: 1.2rem;
    /* Increased font size for button */
    cursor: pointer;
}

.auth-btn:hover {
    background-color: #e67e22;
}

.auth-footer {
    text-align: center;
    margin-top: 20px;
}
```

```
.auth-footer a {
    color: #f39c12;
    text-decoration: none;
}

.auth-footer a:hover {
    text-decoration: underline;
}

.back-home {
    display: block;
    text-align: left;
    font-size: 1rem;
    margin-bottom: 15px;
    color: #f39c12;
    text-decoration: none;
}

.back-home:hover {
    text-decoration: underline;
}

#notification-container {
    position: fixed; /* Щоб сповіщення залишалися видимими при прокрутці */
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    z-index: 1000; /* Щоб сповіщення були над іншими елементами */
}

.notification {
    background-color: #4CAF50; /* Зелений колір для успіху */
    color: white;
    padding: 15px;
    border-radius: 5px;
    margin-bottom: 10px;
    opacity: 0.9;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.notification.error {
    background-color: #f44336; /* Червоний колір для помилки */
}

.notification.success {
    background-color: #4CAF50;
}
```

login.html

```
<!DOCTYPE html>
```

```
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login</title>
   <link rel="stylesheet" href="login.css">
   <script src="login.js" defer></script>

</head>

<body>

   <div class="auth-container">
      <div class="auth-box">
         <a href="index.html" class="back-home">&larr; На головну</a> <!-- Back to home link --
>
         <h2>Вхід</h2>
         <form class="auth-form">
            <div class="form-group">
               <label for="email">Email</label>
               <input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
               <label for="password">Пароль</label>
               <input type="password" id="password" name="password" required>
            </div>
            <button type="submit" class="auth-btn">Увійти</button>
         </form>
         <div class="auth-footer">
            <p>Немає облікового запису? <a href="register.html">Зареєструйтесь тут</a></p>
         </div>
      </div>
   </div>

   <div id="notification-container"></div>
</body>

</html>
```

login.js

```
document.addEventListener("DOMContentLoaded", function () {
   const form = document.querySelector(".auth-form");

   form.addEventListener("submit", function (e) {
      e.preventDefault();

      const email = document.getElementById("email").value.trim();
      const password = document.getElementById("password").value;

      const users = JSON.parse(localStorage.getItem("users")) || [];
```

```
        const currentUser = users.find(user => user.email === email && user.password === password);

        if (!currentUser) {
            showNotification("Невірний email або пароль.", 'error');
            return;
        }

        // Зберігаємо поточного користувача в localStorage для хедера
        localStorage.setItem("user", JSON.stringify({
            username: currentUser.username,
            email: currentUser.email,
            password: currentUser.password // Додаємо пароль
        }));

        showNotification(`Вітаю, ${currentUser.username}!`, 'success');

        window.location.href = "profile.html";
    });
});

function showNotification(message, type = 'success') {
    const notificationContainer = document.getElementById('notification-container');
    const notification = document.createElement('div');
    notification.classList.add('notification');
    notification.classList.add(type);
    notification.textContent = message;
    notificationContainer.appendChild(notification);

    // Автоматичне зникнення через 3 секунди
    setTimeout(() => {
        notification.remove();
    }, 3000);
}
```

profile.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
    min-height: 100vh;
    display: flex;
    flex-direction: column;
}

.profile-container {
    width: 100%;
    height: 100%;
    max-width: none;
```

```
    text-align: left;
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    box-sizing: border-box;
    padding: 30px;
}

.profile-box {
    width: 80%;
    max-width: 960px;
    margin: 20px auto;
}

.profile-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
    width: 100%;
}

.avatar-username {
    display: flex;
    align-items: center;
    gap: 10px;
}

.profile-avatar {
    width: 140px;
    height: 140px;
    border-radius: 50%;
    overflow: hidden;
    border: 2px solid #ddd;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    cursor: pointer;
}

.profile-avatar img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.avatar-username h2 {
    margin: 0;
    color: #333;
    font-size: 2rem;
    display: flex;
    align-items: center;
    gap: 10px;
```

```
}

.edit-username-btn {
    background: none;
    border: none;
    cursor: pointer;
    font-size: 1.2rem;
    color: #f39c12;
}

.edit-username-btn:hover {
    color: #e67e22;
}

.wishlist-count {
    font-size: 1rem;
    color: #555;
}

.wishlist-count span {
    font-weight: bold;
}

.profile-info {
    margin-bottom: 20px;
    width: 100%;
}

.profile-info p {
    margin: 10px 0;
    font-size: 16px;
    color: #555;
}

.profile-info p strong {
    font-weight: bold;
    color: #333;
}

.profile-actions {
    display: flex;
    flex-direction: column;
    gap: 10px;
    width: 300px;
}

.profile-btn {
    display: inline-block;
    padding: 10px 20px;
    background-color: #f39c12;
    color: white;
```

```
    text-decoration: none;
    border-radius: 5px;
    font-size: 1.1rem;
    font-weight: bold;
    text-align: center;
}

.profile-btn:hover {
    background-color: #e67e22;
}

header {
    background-color: #2c3e50;
    padding: 20px;
    color: white;
    position: sticky;
    top: 0;
    z-index: 1000;
    width: 100%;
    box-sizing: border-box;
    position: relative;
}

nav {
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    position: relative;
}

.logo img {
    max-width: 100px;
}

.search-container {
    display: flex;
    align-items: center;
    justify-content: center;
    position: absolute;
    left: 50%;
    transform: translateX(-50%);
}

#search {
    padding: 8px;
    border-radius: 5px;
    border: 1px solid #ccc;
    margin-right: 10px;
}
```

```
#search-btn {
    padding: 8px 15px;
    background-color: #f39c12;
    border: none;
    border-radius: 5px;
    color: white;
    cursor: pointer;
}

#search-btn:hover {
    background-color: #e67e22;
}

.menu {
    display: flex;
    align-items: center;
    margin-left: 0 auto;
}

ul {
    display: flex;
    list-style: none;
}

ul li {
    margin: 0 15px;
}

ul li a {
    color: white;
    text-decoration: none;
    font-weight: bold;
}

ul li a:hover {
    color: #f39c12;
}

/* Modal Styles */
#change-password-modal {
    display: none;
    position: fixed;
    z-index: 2000;
    left: 0;
    top: 0;
    width: 100vw;
    height: 100vh;
    background-color: rgba(0, 0, 0, 0.4);
    justify-content: center;
    align-items: center;
}
```

```
#change-password-modal.active {
    display: flex;
}

.modal-content {
    background-color: #fff;
    padding: 30px;
    border-radius: 8px;
    width: 100%;
    max-width: 400px;
    position: relative;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}

.modal-close {
    position: absolute;
    top: 10px;
    right: 15px;
    font-size: 20px;
    cursor: pointer;
    color: #aaa;
}

.modal-close:hover {
    color: #333;
}

.modal-content h3 {
    margin-top: 0;
    font-size: 1.5rem;
    color: #333;
    margin-bottom: 20px;
}

.modal-content input[type="password"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
}

.modal-content button {
    width: 100%;
    padding: 10px;
    background-color: #f39c12;
    border: none;
    color: #fff;
    font-weight: bold;
```

```
    font-size: 1rem;
    border-radius: 5px;
    cursor: pointer;
}

.modal-content button:hover {
    background-color: #e67e22;
}

/* Загальний контейнер модального вікна */
#change-username-modal,
#change-password-modal {
    display: none; /* Спочатку приховано */
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.1);
    z-index: 1001;
    justify-content: center;
    align-items: center;
}

/* Коли модаль активний — показати */
#change-username-modal.active,
#change-password-modal.active {
    display: flex;
}

/* Вміст модального вікна */
.modal-content {
    background: white;
    padding: 20px;
    border-radius: 10px;
    min-width: 300px;
    position: relative;
}

/* Кнопка закриття */
.modal-close {
    position: absolute;
    top: 10px;
    right: 15px;
    font-size: 20px;
    cursor: pointer;
}

.modal-content input[type="text"] {
    width: 100%;
    padding: 10px;
```

```
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-sizing: border-box;
    font-size: 1rem;
}

#notification-container {
    position: fixed; /* Щоб сповіщення залишалися видимими при прокрутці */
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    z-index: 1000; /* Щоб сповіщення були над іншими елементами */
}

.notification {
    background-color: #4CAF50; /* Зелений колір для успіху */
    color: white;
    padding: 15px;
    border-radius: 5px;
    margin-bottom: 10px;
    opacity: 0.9;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.notification.error {
    background-color: #f44336; /* Червоний колір для помилки */
}

.notification.success {
    background-color: #4CAF50;
}
```

profile.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Профіль</title>
    <link rel="stylesheet" href="profile.css">
    <link rel="stylesheet" href="index.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css">
</head>
<body>
<div id="header-placeholder"></div>
<main>
    <div class="profile-container">
```

```
      <div class="profile-box">
        <div class="profile-header">
          <div class="avatar-username">
            <div class="profile-avatar" onclick="document.getElementById('avatar-upload').click();">
              <img id="profile-avatar-img" src="image/avatarplaceholder.png" alt="Аватар користувача">
              <input type="file" id="avatar-upload" accept="image/*" style="display: none;">
            </div>
            <h2 id="profile-username-header"></h2>
          </div>
          <div class="wishlist-count">
            <span>Бажане:</span> <span id="wishlist-item-count">0</span>
          </div>
        </div>
        <div class="profile-info">
          <p><strong>Email:</strong> <span id="profile-email"></span></p>
        </div>
        <div class="profile-actions"></div>
      </div>
    </div>
</main>

<!-- Модальне вікно зміни паролю -->
<div id="change-password-modal" class="modal"></div>
<div id="notification"></div>

<script src="header.js"></script>
<script src="profile.js"></script>
<script src="index.js"></script>

</body>
</html>
```

profile.js

```
document.addEventListener('DOMContentLoaded', function () {
    const user = JSON.parse(localStorage.getItem('user'));
    const users = JSON.parse(localStorage.getItem('users')) || [];
    const currentUserData = users.find(u => u.email === user?.email);

    const usernameEl = document.getElementById('profile-username-header');
    const emailEl = document.getElementById('profile-email');
    const avatarImg = document.getElementById('profile-avatar-img');
    const avatarUpload = document.getElementById('avatar-upload');

    const notificationContainer = document.createElement('div');
    notificationContainer.id = 'notification-container';
    document.body.appendChild(notificationContainer);

    function showNotification(message, type = 'success') {
```

```
        const notification = document.createElement('div');
        notification.className = `notification ${type}`;
        notification.textContent = message;
        notificationContainer.appendChild(notification);
        setTimeout(() => {
            notification.remove();
        }, 3000);
    }

    if (!user) {
        window.location.href = 'login.html';
        return;
    }

    // Заповнення профілю
    if (usernameEl) usernameEl.textContent = user.username;
    if (emailEl) emailEl.textContent = user.email;

    if (currentUserData && currentUserData.wishlist) {
        document.getElementById('wishlist-item-count').textContent =
currentUserData.wishlist.length;
    }

    if (currentUserData && currentUserData.avatar) {
        avatarImg.src = currentUserData.avatar;
    }

    // Завантаження нового аватара
    avatarUpload.addEventListener('change', function (event) {
        const file = event.target.files[0];
        if (file) {
            const reader = new FileReader();
            reader.onload = function (e) {
                avatarImg.src = e.target.result;

                const updatedUsers = users.map(u => {
                    if (u.email === user.email) {
                        return { ...u, avatar: e.target.result };
                    }
                    return u;
                });
                localStorage.setItem('users', JSON.stringify(updatedUsers));
                localStorage.setItem('user', JSON.stringify({ ...user, avatar: e.target.result }));
                showNotification('Аватар оновлено', 'success');
            };
            reader.readAsDataURL(file);
        }
    });

    // Модальне вікно зміни юзернейму
    const usernameModal = document.createElement('div');
```

```
usernameModal.id = 'change-username-modal';
usernameModal.innerHTML = `
   <div class="modal-content">
      <span class="modal-close" id="close-username-modal">&times;</span>
      <h3>Зміна імені користувача</h3>
      <input type="text" id="new-username-input" placeholder="Нове ім’я користувача">
      <button id="save-username-btn">Зберегти</button>
   </div>
`;
document.body.appendChild(usernameModal);

// Додавання кнопки редагування юзернейму
const editBtn = document.createElement('button');
editBtn.innerHTML = '✏️';
editBtn.classList.add('edit-username-btn');
usernameEl.appendChild(editBtn);

editBtn.addEventListener('click', () => {
   document.getElementById('new-username-input').value = user.username;
   usernameModal.classList.add('active');
});

document.getElementById('close-username-modal').addEventListener('click', () => {
   usernameModal.classList.remove('active');
});

document.getElementById('save-username-btn').addEventListener('click', () => {
   const newUsername = document.getElementById('new-username-input').value.trim();
   if (newUsername) {
      usernameEl.childNodes[0].nodeValue = newUsername;

      const updatedUsers = users.map(u => {
         if (u.email === user.email) {
            return { ...u, username: newUsername };
         }
         return u;
      });
      localStorage.setItem('users', JSON.stringify(updatedUsers));
      localStorage.setItem('user', JSON.stringify({ ...user, username: newUsername }));
      showNotification('Ім’я користувача змінено', 'success');
      usernameModal.classList.remove('active');
   } else {
      showNotification('Ім’я не може бути порожнім', 'error');
   }
});

// Модальне вікно зміни пароля
const changePasswordModal = document.createElement('div');
changePasswordModal.id = 'change-password-modal';
changePasswordModal.innerHTML = `
```

```
    <div class="modal-content">
        <span class="modal-close" id="close-password-modal">&times;</span>
        <h3>Зміна паролю</h3>
        <input type="password" id="old-password" placeholder="Старий пароль">
        <input type="password" id="new-password" placeholder="Новий пароль">
        <input type="password" id="confirm-password" placeholder="Повторіть новий пароль">
        <button id="save-password-btn">Зберегти</button>
    </div>
`;
document.body.appendChild(changePasswordModal);

const changePasswordBtn = document.createElement('a');
changePasswordBtn.href = '#';
changePasswordBtn.textContent = 'Змінити пароль';
changePasswordBtn.classList.add('profile-btn');
document.querySelector('.profile-actions').appendChild(changePasswordBtn);

changePasswordBtn.addEventListener('click', () => {
    changePasswordModal.classList.add('active');
});

document.getElementById('close-password-modal').addEventListener('click', () => {
    changePasswordModal.classList.remove('active');
});

document.getElementById('save-password-btn').addEventListener('click', () => {
    const oldPass = document.getElementById('old-password').value;
    const newPass = document.getElementById('new-password').value;
    const confirmPass = document.getElementById('confirm-password').value;

    console.log('Введений старий пароль:', oldPass);
    console.log('Пароль з об\'єкта user:', user.password);

    if (oldPass !== user.password) {
        showNotification('Неправильний старий пароль', 'error');
        return;
    }
    if (newPass.length < 6) {
        showNotification('Новий пароль має містити щонайменше 6 символів', 'error');
        return;
    }
    if (newPass !== confirmPass) {
        showNotification('Нові паролі не співпадають', 'error');
        return;
    }

    const updatedUsers = users.map(u => {
        if (u.email === user.email) {
            return { ...u, password: newPass };
        }
        return u;
```

```
        });
        localStorage.setItem('users', JSON.stringify(updatedUsers));
        localStorage.setItem('user', JSON.stringify({ ...user, password: newPass }));
        showNotification('Пароль успішно змінено', 'success');
        changePasswordModal.classList.remove('active');
    });
});
```

register.css

```
/* General Reset and Styling */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    background-color: #ecf0f1;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    /* Full viewport height */
}

/* Authentication Forms Container */
.auth-container {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    height: 100%;
    /* Full viewport height */
    position: absolute;
    /* Makes it fill the entire screen */
}

.auth-box {
    background-color: white;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 40px;
    width: 100%;
    max-width: 400px;
    margin: auto;
    /* Automatically adjust margins to center the form */
}

h2 {
    text-align: center;
```

```
    margin-bottom: 20px;
    font-size: 2rem;
}

.auth-form .form-group {
    margin-bottom: 20px;
}

.auth-form label {
    display: block;
    font-size: 1rem;
    margin-bottom: 5px;
}

.auth-form input {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
    border-radius: 5px;
    border: 1px solid #ccc;
}

.auth-form input:focus {
    border-color: #f39c12;
    outline: none;
}

.auth-btn {
    width: 100%;
    padding: 12px;
    background-color: #f39c12;
    color: white;
    border: none;
    border-radius: 5px;
    font-size: 1rem;
    cursor: pointer;
}

.auth-btn:hover {
    background-color: #e67e22;
}

.auth-footer {
    text-align: center;
    margin-top: 20px;
}

.auth-footer a {
    color: #f39c12;
    text-decoration: none;
}
```

```
.auth-footer a:hover {
    text-decoration: underline;
}

.back-home {
    display: block;
    text-align: left;
    font-size: 1rem;
    margin-bottom: 15px;
    color: #f39c12;
    text-decoration: none;
}

.back-home:hover {
    text-decoration: underline;
}

#notification-container {
    position: fixed; /* Щоб сповіщення залишалися видимими при прокрутці */
    top: 20px;
    left: 50%;
    transform: translateX(-50%);
    z-index: 1000; /* Щоб сповіщення були над іншими елементами */
}

.notification {
    background-color: #4CAF50; /* Зелений колір для успіху */
    color: white;
    padding: 15px;
    border-radius: 5px;
    margin-bottom: 10px;
    opacity: 0.9;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.notification.error {
    background-color: #f44336; /* Червоний колір для помилки */
}

.notification.success {
    background-color: #4CAF50;
}
```

register.html

```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link rel="stylesheet" href="register.css">
    <script src="register.js" defer></script>
</head>

<body>
    <div class="auth-container">
        <div class="auth-box">
            <a href="index.html" class="back-home">&larr; На головну</a> <!-- Back to home link --
>
            <h2>Реєстрація</h2>
            <form class="auth-form">
                <div class="form-group">
                    <label for="username">Ім'я користувача</label>
                    <input type="text" id="username" name="username" required>
                </div>
                <div class="form-group">
                    <label for="email">Email</label>
                    <input type="email" id="email" name="email" required>
                </div>
                <div class="form-group">
                    <label for="password">Пароль</label>
                    <input type="password" id="password" name="password" required>
                </div>
                <button type="submit" class="auth-btn">Зареєструватися</button>
            </form>
            <div class="auth-footer">
                <p>Уже маєте обліковий запис? <a href="login.html">Увійдіть тут</a></p>
            </div>
        </div>
    </div>

    <div id="notification-container"></div>
</body>

</html>
```

register.js

```
document.addEventListener("DOMContentLoaded", function () {
    const form = document.querySelector(".auth-form");

    form.addEventListener("submit", function (e) {
        e.preventDefault();

        const username = document.getElementById("username").value.trim();
        const email = document.getElementById("email").value.trim();
        const password = document.getElementById("password").value;
```

```
        if (!username || !email || !password) {
            showNotification("Будь ласка, заповніть усі поля.", 'error');
            return;
        }

        if (password.length < 6) {
            showNotification("Пароль повинен містити не менше 6 символів.", 'error');
            return;
        }

        const users = JSON.parse(localStorage.getItem("users")) || [];

        const emailExists = users.some(user => user.email === email);
        if (emailExists) {
            showNotification("Користувач з таким email вже існує.", 'error');
            return;
        }

        const usernameExists = users.some(user => user.username === username);
        if (usernameExists) {
            showNotification("Користувач з таким ім'ям вже існує.", 'error');
            return;
        }

        const defaultAvatarUrl = 'C:/Unik/4 курс 2 семестр/БКР/NoTitle/image/avatarplaceholder.png';
        const newUser = {
            username,
            email,
            password,
            avatar: defaultAvatarUrl,
            wishlist: []
        };

        users.push(newUser);
        localStorage.setItem("users", JSON.stringify(users));

        showNotification("Реєстрація успішна!", 'success');
        window.location.href = "login.html";
    });
});

function showNotification(message, type = 'success') {
    const notificationContainer = document.getElementById('notification-container');
    const notification = document.createElement('div');
    notification.classList.add('notification');
    notification.classList.add(type);
    notification.textContent = message;
    notificationContainer.appendChild(notification);
```

```
    // Автоматичне зникнення через 3 секунди
    setTimeout(() => {
        notification.remove();
    }, 3000);
}
```

wishlist.css

```
/* wishlist.css */

body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    color: #333;
}

header {
    background-color: #2c3e50;
    padding: 20px;
    color: white;
    position: sticky;
    top: 0;
    z-index: 1000;
}

nav {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    margin: auto;
}

.logo img {
    max-width: 100px;
}

ul.nav-links {
    display: flex;
    list-style: none;
    margin: 0;
    padding: 0;
}

ul.nav-links li {
    margin-left: 20px;
}

ul.nav-links a {
    color: white;
```

```
    text-decoration: none;
    font-weight: bold;
    text-transform: uppercase;
}

ul.nav-links a:hover {
    color: #f39c12;
}

.wishlist-container {
    padding: 20px 20px 40px 20px;
    margin: auto;
}

h1 {
    text-align: center;
    margin-bottom: 50px;
    margin-top: 0;
    color: #2c3e50;
    font-size: 2rem;
}

#wishlist-content {
    display: flex;
    flex-direction: column;
    gap: 20px;
}

.wishlist-card {
    display: flex;
    align-items: center;
    background-color: white;
    border-radius: 12px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    padding: 10px;
    transition: transform 0.3s ease;
    position: relative;
    margin-top: 20px;

}

.wishlist-card img {
    width: 120px;
    height: 160px;
    object-fit: cover;
    border-right: 2px solid #ddd;
    margin-right: 20px;
}

.wishlist-card h3 {
    margin: 0;
```

```
    font-size: 1.2rem;
    color: #34495e;
    flex-grow: 1;
    text-align: left;
}

.wishlist-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
}

/* Placeholder message */
p {
    text-align: center;
    font-size: 1.2rem;
    color: #888;
    padding: 50px;
}

/* Button (поки що схований, можеш використати пізніше) */
.remove-btn {
    background-color: #e74c3c;
    color: white;
    border: none;
    padding: 8px 12px;
    border-radius: 5px;
    font-size: 0.9rem;
    cursor: pointer;
    margin-left: auto;
    transition: background-color 0.3s ease;
}

.remove-btn:hover {
    background-color: #c0392b;
}
```

wishlist.html

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="wishlist.css">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Wishlist</title>
    <link rel="stylesheet" href="wishlist.css">
</head>

<body>
    <div class="wishlist-container" id="wishlist-container"></div>
```

```
    <script src="wishlist.js"></script>
    <script src="index.html"></script>
</body>

</html>
```

wishlist.js

```
document.addEventListener("DOMContentLoaded", () => {
    const user = JSON.parse(localStorage.getItem("user"));

    const container = document.getElementById("wishlist-container");

    if (!user) {
        container.innerHTML = "<p>Увійдіть в акаунт, щоб переглянути список бажаного.</p>";
        return;
    }

    if (user) {
        container.innerHTML = "<h1>Список бажаного</h1>";
    }

    const wishlistKey = `wishlist_${user.email}`;
    const wishlist = JSON.parse(localStorage.getItem(wishlistKey)) || [];

    if (wishlist.length === 0) {
        container.innerHTML = "<p>Список бажаного порожній.</p>";
        return;
    }

    wishlist.forEach(item => {
        const card = document.createElement("div");
        card.classList.add("wishlist-card");
        card.innerHTML = `
            <img src="${item.image}" alt="${item.title}">
            <h3>${item.title}</h3>
        `;
        container.appendChild(card);
    });
});
```

Додаток Г – Графічна частина

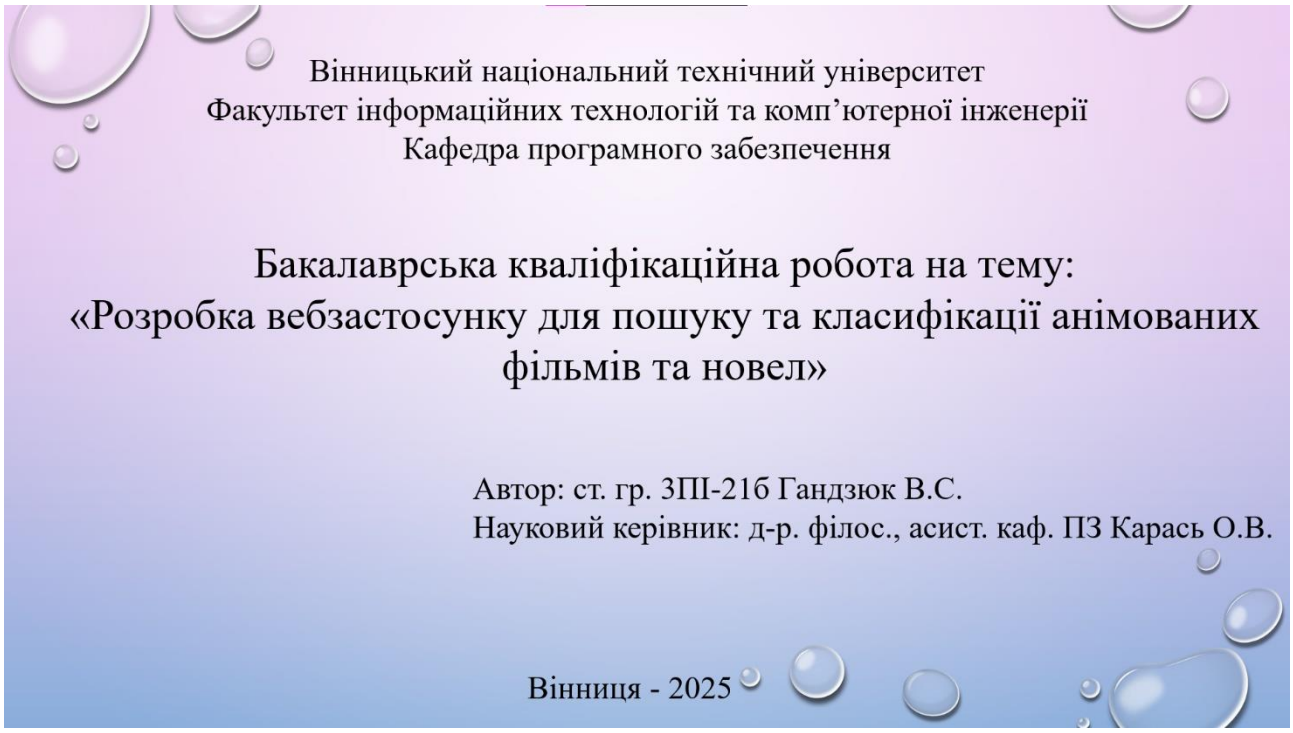

Рисунок Г.1 – Титульний аркуш

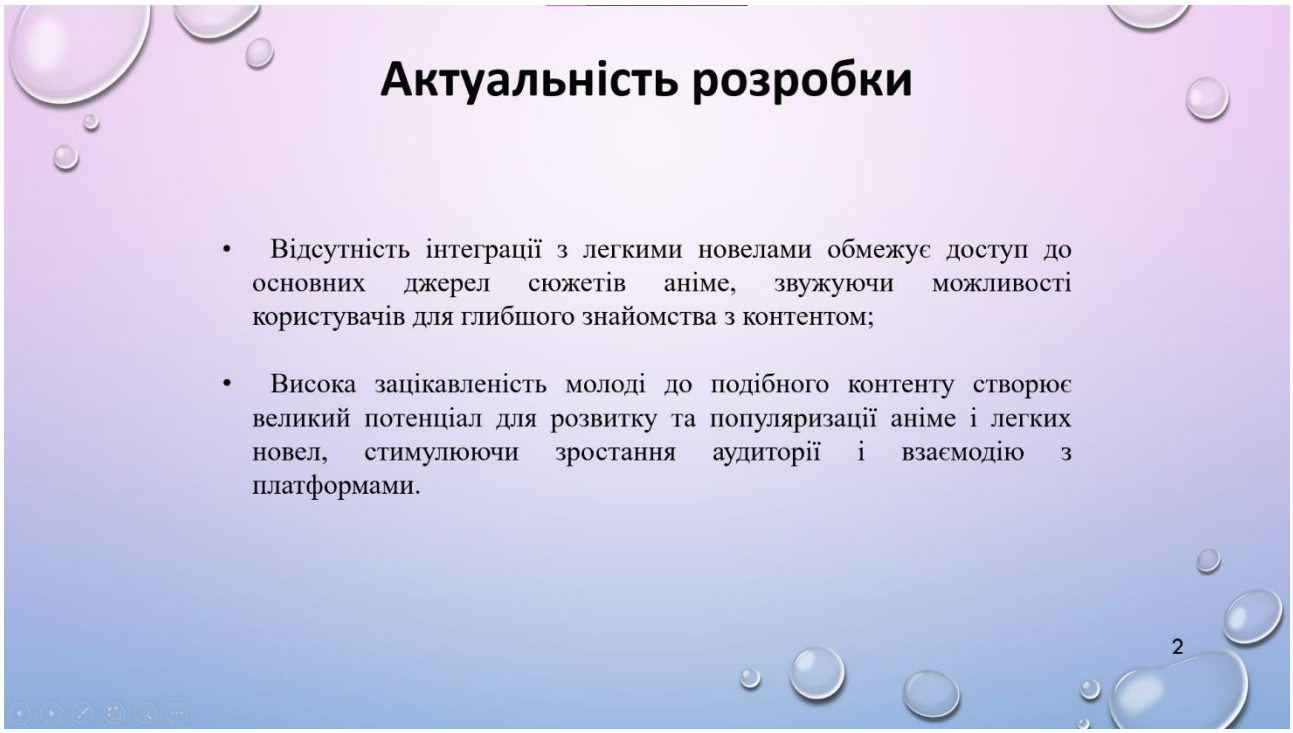

Рисунок Г.2 – Актуальність розробки

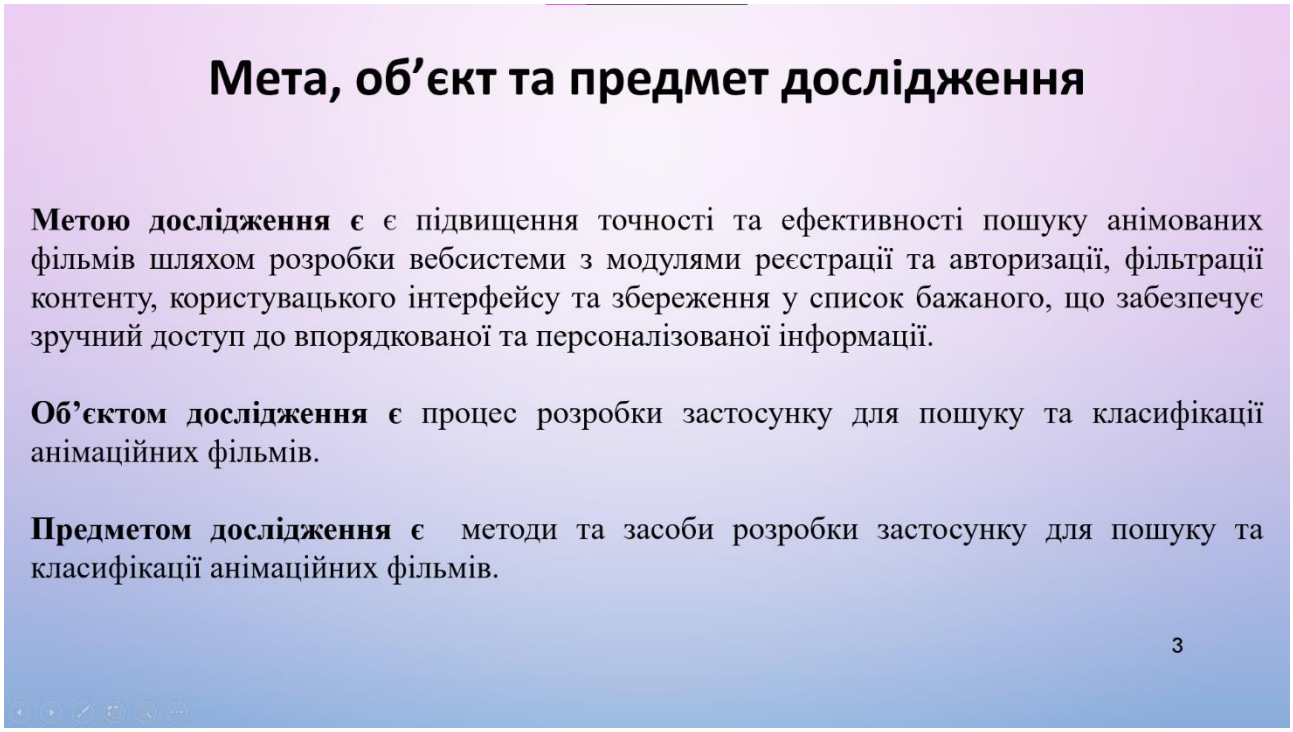

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

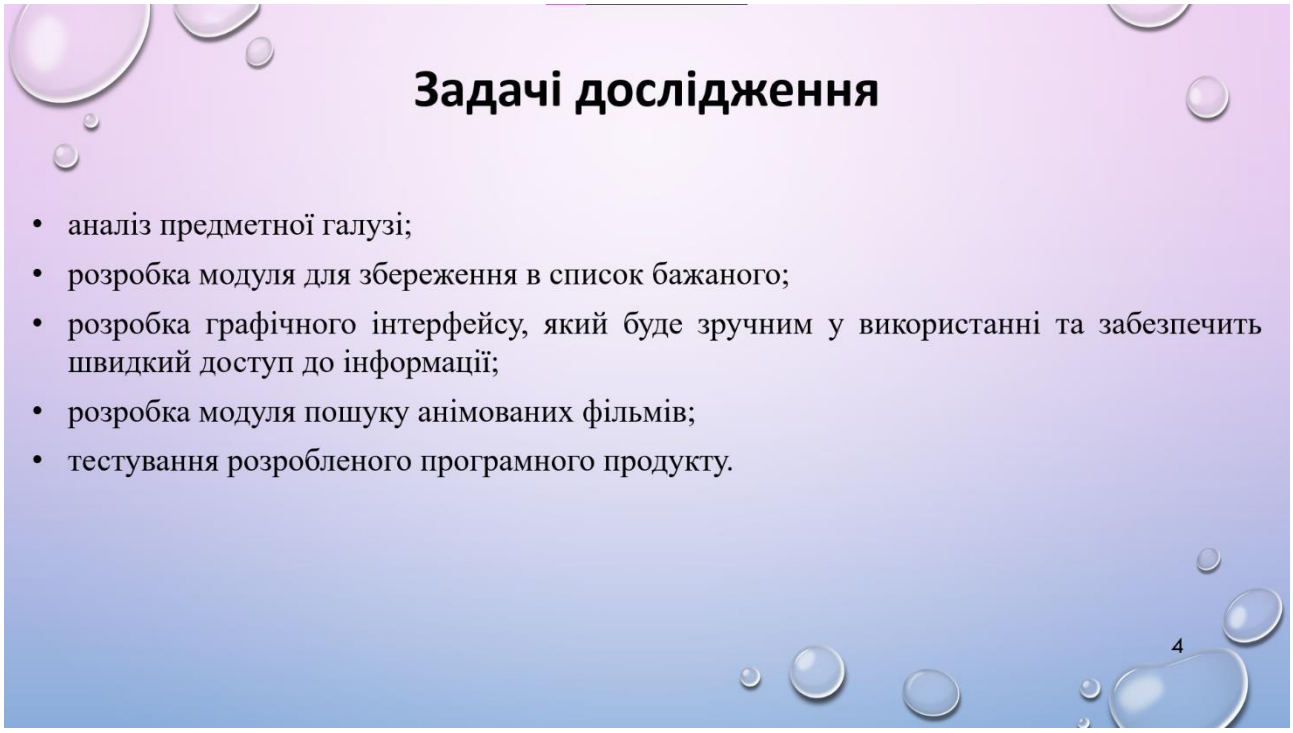

Рисунок Г.4 – Задачі дослідження

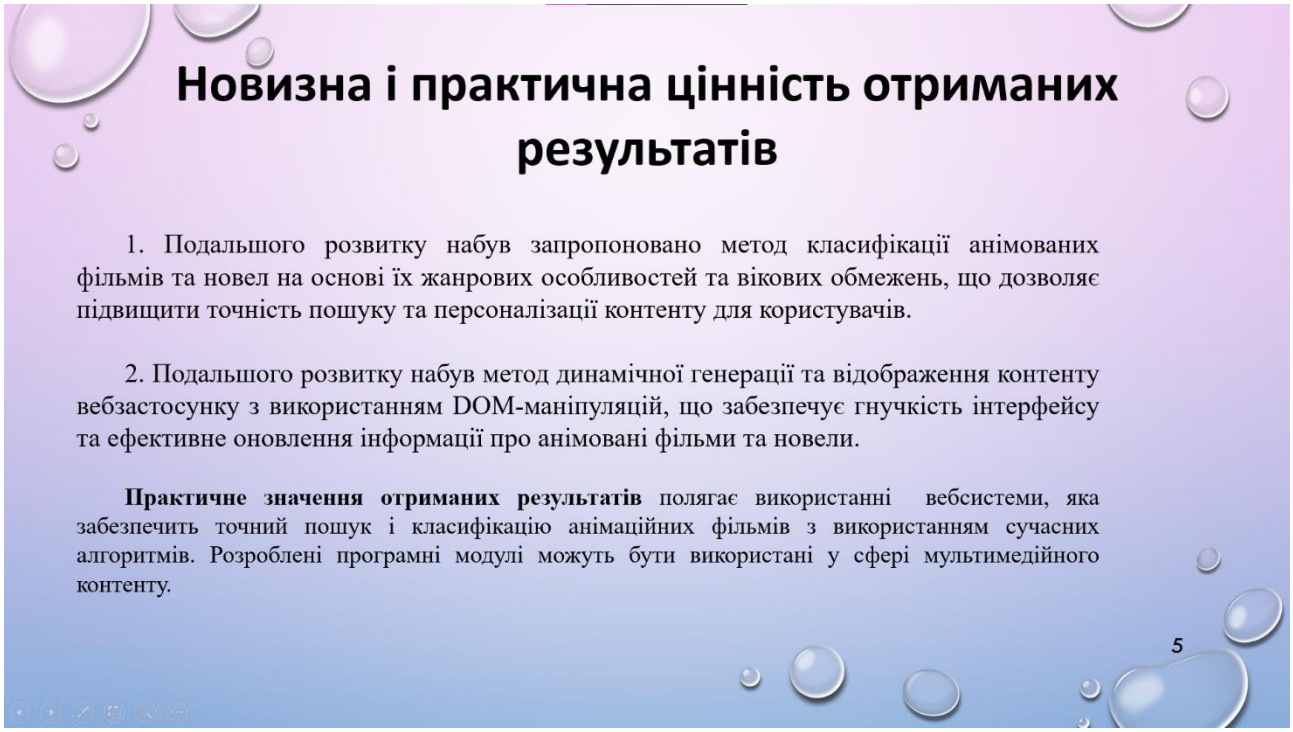

Рисунок Г.5 – Новизна і практична цінність отриманих результатів

Рисунок Г.6 – Порівняльний аналіз аналогів

Рисунок Г.7 – Загальний алгоритм роботи системи

Рисунок Г.8 – Алгоритм реєстрації користувача

Рисунок Г.9 – Діаграма алгоритму пошуку

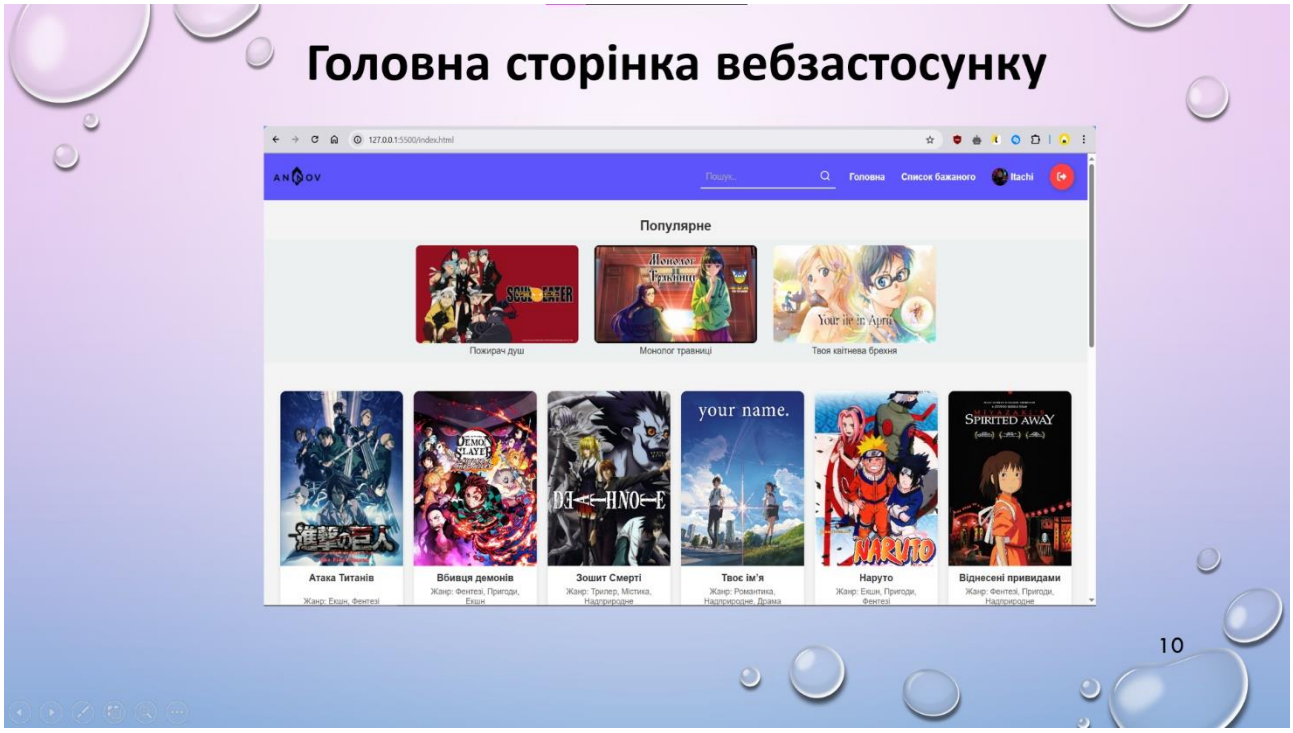

Рисунок Г.10 – Головна сторінка вебзастосунку

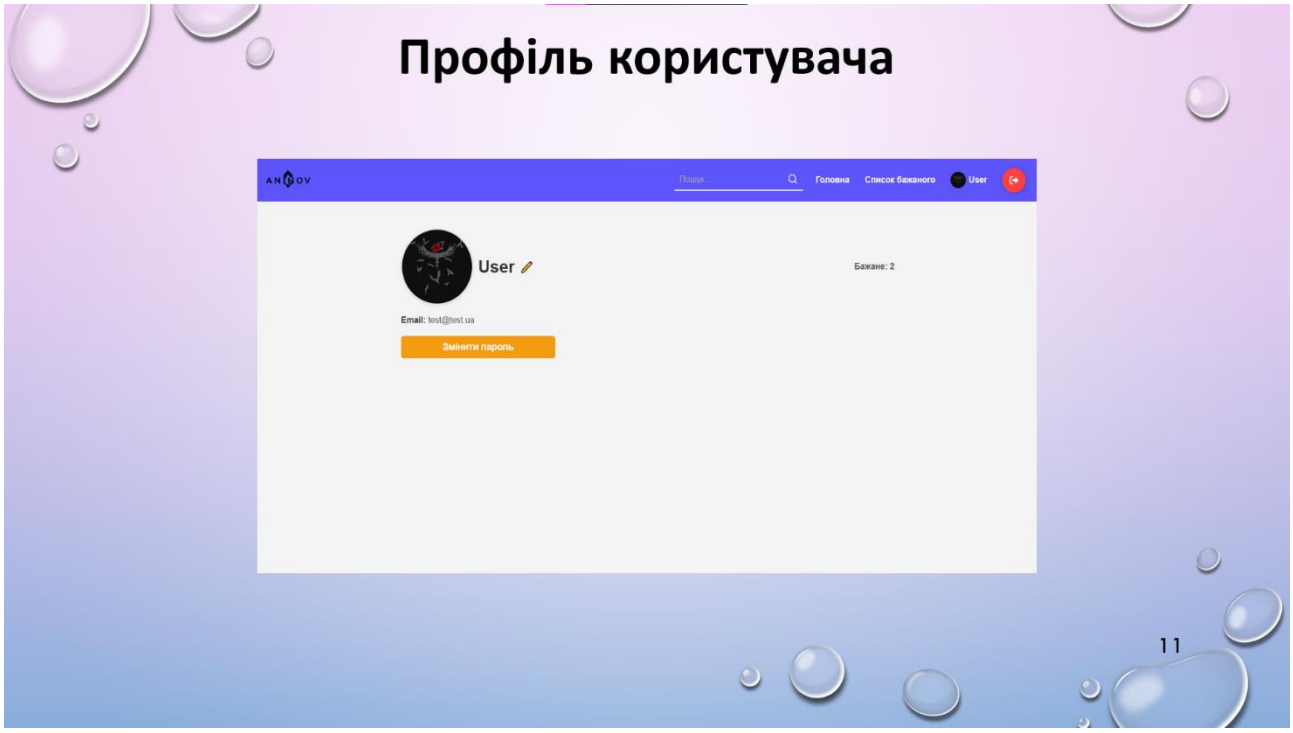

Рисунок Г.11 – Профіль користувача

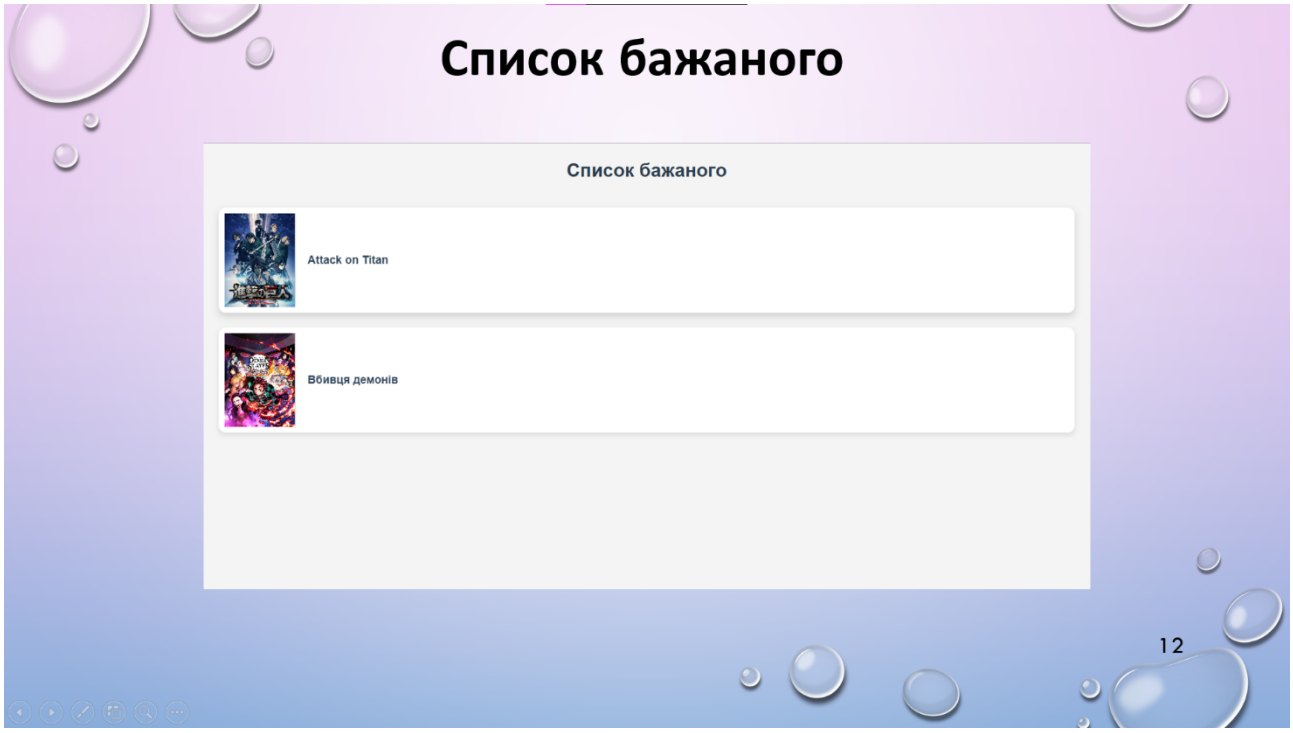

Рисунок Г.12 – Список бажаного

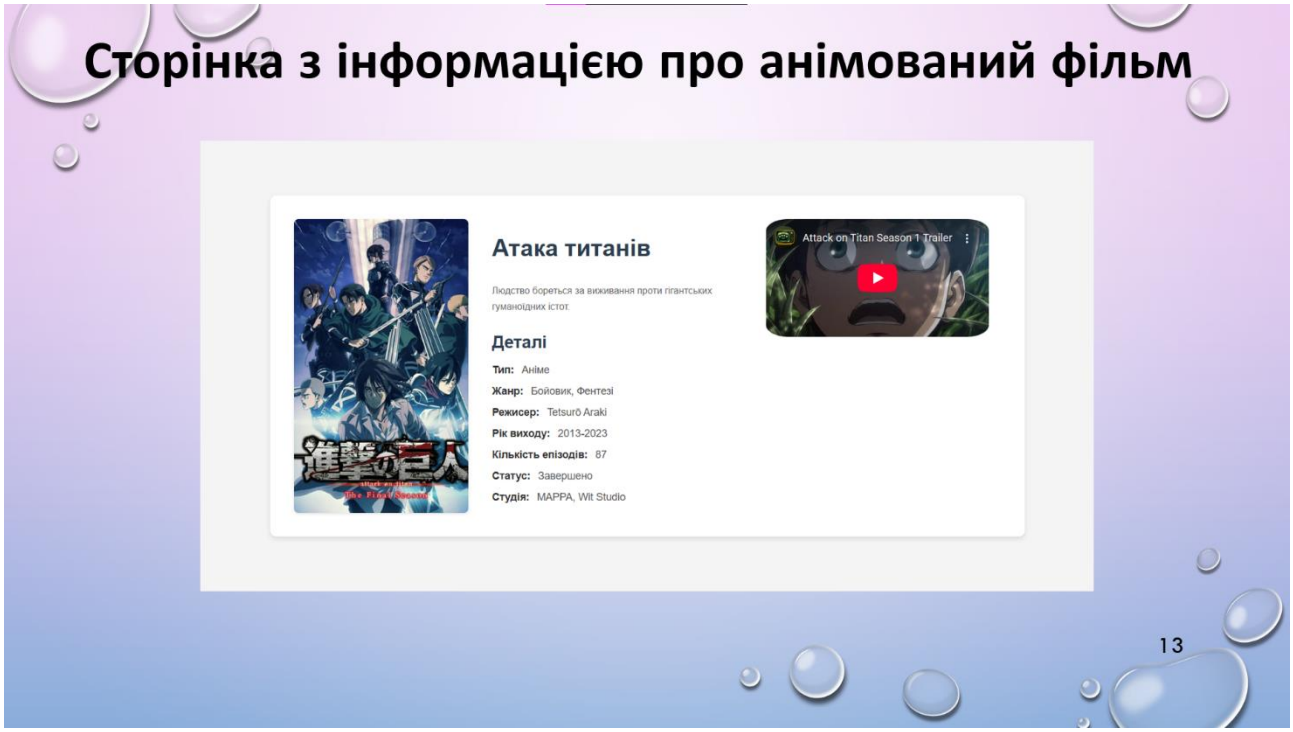

Рисунок Г.13 – Сторінка з інформацією про анімований фільм

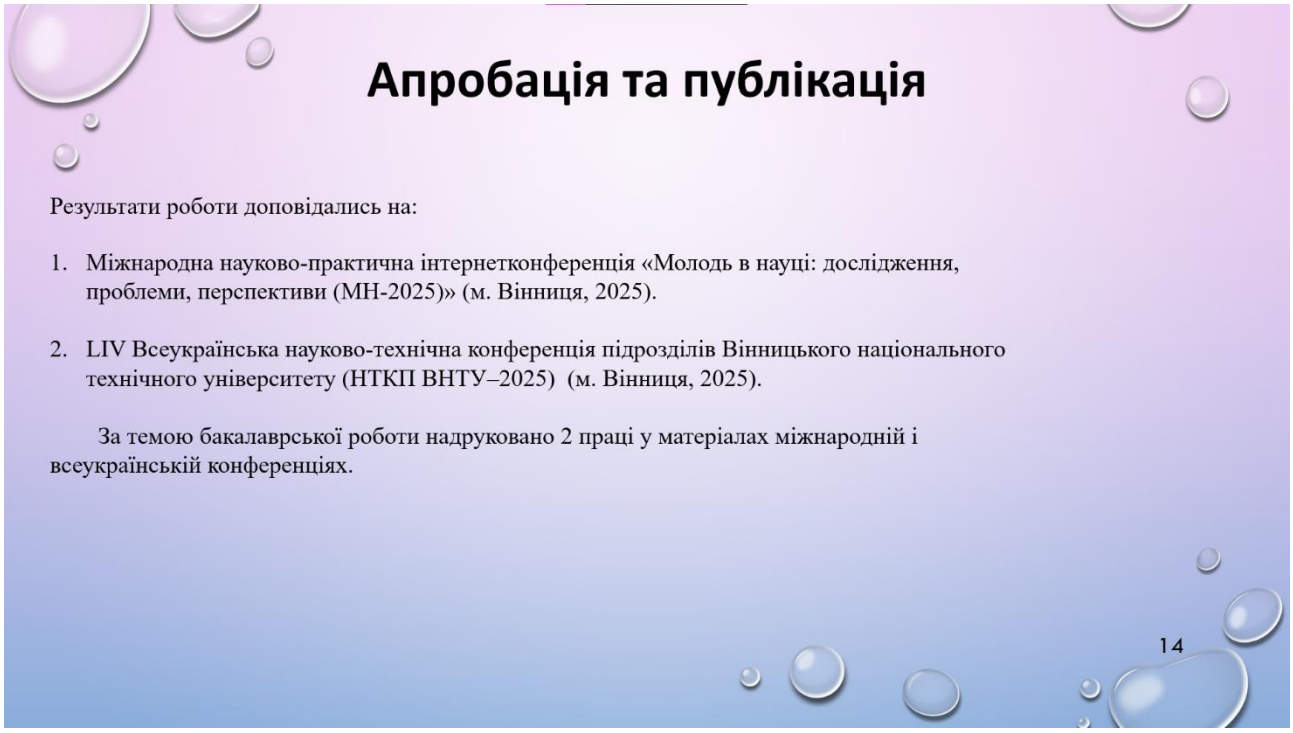

Рисунок Г.14 – Апробація та публікація

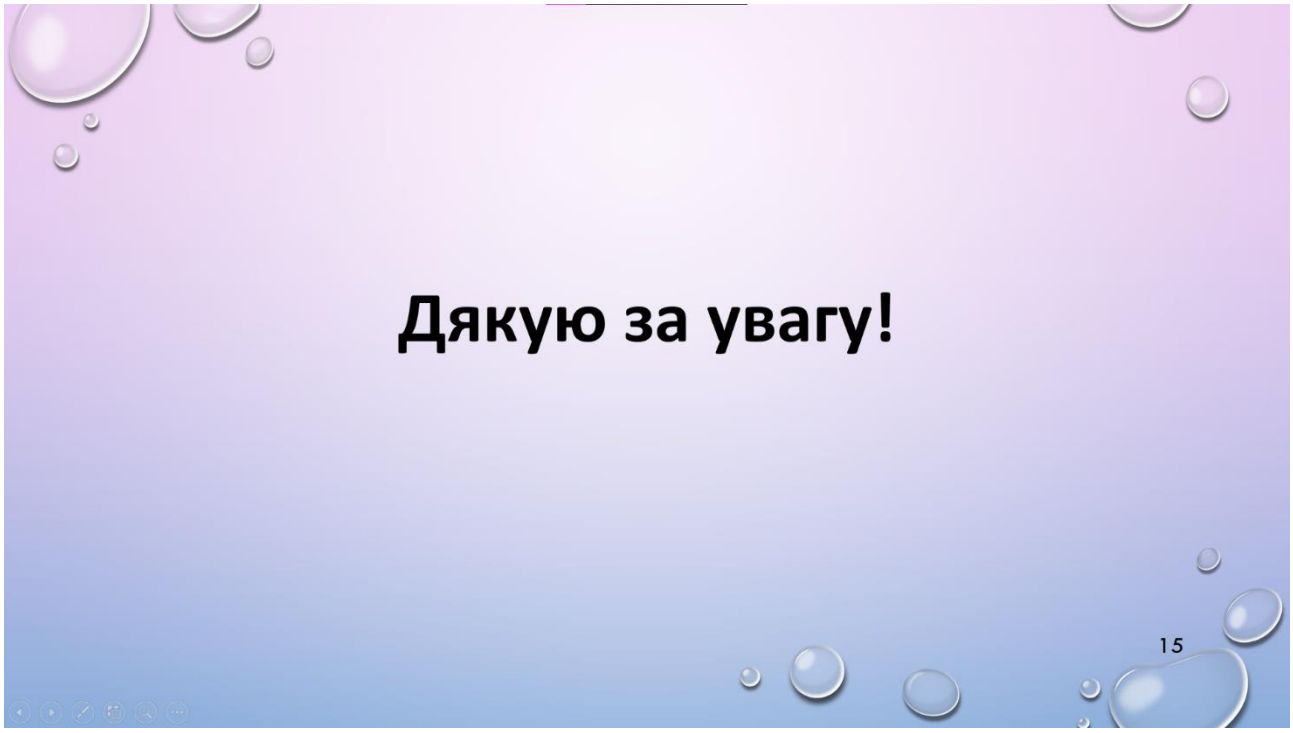

Рисунок Г.15 – Закінчення презентації