

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ
КОРИСТУВАЦЬКОГО СПИСКУ ЧИТАННЯ
DEVELOPMENT OF A WEB APPLICATION FOR
USER'S READING LIST TRACKING

Виконав(ла): студент(ка) 4 курсу, групи КНТ-210

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ІНДИК А.А.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ІНДИКА Андрія Андрійовича
(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для відстеження користувацького списку читання. Development of a Web Application for User's Reading List Tracking
керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір і обґрунтування структури системи. 3. Основні рішення реалізації компонентів системи. 4. Експлуатація та тестування.

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

6. Консультанти розділів проєкту (роботи)

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

7. Дата видачі завдання “15” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір і обґрунтування структури системи.	2 тиждень	Розділ 2
4	Основні рішення реалізації компонентів системи.	3-4 тижні	Розділ 3
5	Експлуатація та тестування.	5 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
7	Нормоконтроль та рецензування.	7 тиждень	
8	Захист роботи.	8 тиждень	

Студент(ка)

підпис Андрій ІНДИК
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

Тетяна ФЕДОРОНЧАК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра:
112 с., 6 табл., 42 рис., 3 дод., 15 джерела.

СПИСОК ЧИТАННЯ, ВЕБЗАСТОСУНОК, ВІДСТЕЖЕННЯ,
КЕРУВАННЯ, РОЗПОДІЛЕННЯ, ЛІТЕРАТУРА, JAVASCRIPT.

Об'єкт дослідження – програмні засоби для підтримки користувацьких хобі.

Предмет дослідження – програмні засоби для управління списками літератури.

Мета роботи – створення вебзастосунка для покращення слідкування за власним списком читання.

Матеріали, методи та технічні засоби: мови розмітки сторінки HTML та стилів CSS, мова програмування JavaScript, бібліотека React на її основі, середовище виконання Node.js версії 20.12.2, система керування базами даних MySQL, персональний комп'ютер з процесором Intel Core I7-3770 під управлінням ОС Microsoft Windows 10, вебпереглядач на основі Chromium версії 124.

Результати. Створено вебзастосунок, що керується мовою JavaScript, який дає користувачу можливість керувати списком читання.

Висновки. Розроблено систему керування особистим списком читання користувача, до якої можна додавати нову літературу, інформацію про неї, розподіляти її за станами читання та категоріями.

Галузь використання – покращення та структуризація хобі читання, створення списку прочитаного, відслідковування поточної зайнятості та планування читання майбутньої літератури.

ABSTRACT

Explanatory note to the diploma qualifying work of a bachelor: 112 pages, 6 tables, 42 figures, 3 appendixes, 15 sources.

READING LIST, WEB APPLICATION, TRACKING, MANAGEMENT, DISTRIBUTION, LITERATURE, JAVASCRIPT.

Object of research is software tools for supporting user hobbies.

The subject of research is software tools for managing reading lists.

The purpose of the study is to create a web application to improve the tracking of user's reading list.

Materials, methods and tools: HTML and CSS page markup languages, JavaScript programming language, React library based on it, Node.js runtime version 20.12.2, MySQL database management system, a personal computer with an Intel Core I7-3770 processor running Microsoft Windows 10, a web browser based on Chromium version 124.

Results. A JavaScript-based web application has been created that allows the user to manage the reading list.

Conclusions. A system for managing the user's personal reading list has been developed, which allows adding new literature, information about it, and dividing it into reading states and categories.

The field of application is improving and structuring reading hobbies, creating a reading list, tracking current activity and planning the reading of future literature.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз предметної області	10
1.1 Предметна область.....	10
1.2 Огляд аналогів.....	11
1.3 Постановка завдання.....	21
1.4 Висновки з розділу 1.....	23
2 Вибір і обґрунтування структури системи	24
2.1 Вибір інструментів розробки.....	24
2.2 Створення структурної схеми вебзастосунку	36
2.3 Висновки з розділу 2.....	41
3 Основні рішення реалізації компонентів системи.....	42
3.1 Функціональне проєктування	42
3.2 Проєктування бази даних	50
3.3 Проєктування інтерфейсу.....	51
3.4 Висновки з розділу 3.....	57
4 Експлуатація та тестування.....	58
4.1 Опис проєкту	58
4.2 Умови функціонування проєкту.....	58
4.3 Розгортання проєкту	59
4.4 Експлуатація проєкту	59
4.4 Тестування проєкту.....	61
4.5 Висновки з розділу 4.....	63
Висновки	64

Перелік джерел посилань	65
Додаток А Технічне завдання	67
Додаток Б Текст програми	71
Додаток В Слайди презентації.....	104

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CSS	– Cascading Style Sheets;
DOM	– Document Object Model;
JSX	– JavaScript XML;
HTML	– Hypertext Markup Language;
npm	– Node Package Manager;
ГБ	– гігабайт;
ОЗП	– оперативний запам'ятовуючий пристрій;
ОС	– операційна система;
СКБД	– система керування базами даних.

ВСТУП

Актуальність цієї роботи полягає в тому, що книги вже десятки сотень років являються головними джерелами знань та інформації, що написані одними людьми для читання іншими. І хоч форма представлення письмової інформації змінювалася з роками, починаючи свій шлях з глиняних табличок, папірусу, пергаменту, дійшла до звичного нам паперу, з яким людство паралельно починає використовувати електронні книги та аудіокниги, які можна читати на будь-якому носії інформації, який призначений для її відтворення, це може бути телефон, комп'ютер або навіть окремий девайс для читання, який і називається електронна книга. Це показує нам, що хоч ми і відходимо від звичного сприйняття паперових книг, самі книги не зникають і ніколи не зникнуть. Папір замінюється планшетами, бібліотеки – інтернет-репозиторіями. В світі, де актуальність книг безперечна і це навряд чи зміниться у майбутньому, необхідні засоби для зручного контролю того, що вже прочитано, та що стоїть у черзі для прочитання у майбутньому.

Метою цієї роботи є створення програмного забезпечення у вигляді вебзастосунку, основною ціллю якого буде спрощення процесу відслідковування списку читання користувача, допомога в його структуризації та підбитті підсумків читання.

Задача цієї роботи полягає у розробці проєкту для відстеження користувацького списку читання, створення його зовнішньої складової, його програмна реалізація у вигляді вебзастосунку, побудованого на мові програмування JavaScript та її бібліотеці React.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область

За оцінками компанії Google, вже майже п'ятнадцять років тому у світі існувало близько 130 мільйонів різноманітних книжкових праць, ця кількість продовжує зростати [1].

Книги існували століттями як джерело інформації та знань, відтворюючи важливі тексти та твори, створені людьми для використання та аналізу. Від глиняних табличок, папірусу та пергаменту до звичайного паперу. Водночас книги не втратили своєї актуальності, навіть коли електронні книги та аудіокниги стали доступними на різноманітних носіях, включаючи мобільні телефони, комп'ютери та спеціалізовані пристрої для читання.

Перш за все, книги залишаються невичерпним джерелом знань та інформації, створених однією людиною для сприйняття і розуміння іншою. Книги є основою для передачі знань і культурних цінностей від покоління до покоління і забезпечують безперервний розвиток суспільства.

По-друге, книги є свідками еволюції людської думки та ідей, відображаючи розмаїття культур, світоглядів і способів мислення. Вони є важливими реліквіями нашої культурної спадщини, сприяють діалогу між сьогоденням і минулим та допомагають зберегти пам'ять про наше коріння і цінності.

По-третє, книги є фундаментальними інструментами для критичного мислення, самоосвіти та саморозвитку. Книги стимулюють мислення, допомагають розібратися в складних проблемах і знайти можливі рішення, надихають на досягнення.

Таким чином, книги залишаються актуальним і необхідним джерелом знань, мудрості та натхнення в сучасному світі. Важливість книги полягає в комунікації, яка сприяє збереженню та передачі культурної спадщини, особистісному зростанню та безперервному розвитку суспільства.

Розробка ефективних інструментів для структурування літературних творів важлива для забезпечення організованості та послідовності в процесі сприйняття та аналізу інформації, яку ми споживаємо. Історія літератури багата на розмаїття жанрів, галузей, тем і авторів, тому можливість вести списки літератури є важливим компонентом для кожного читача.

Інструменти управління списком літератури допоможуть не загубитися у величезному морі знань, які може запропонувати людство. Завдяки цим інструментам ми можемо структурувати свій читацький досвід, розставляючи пріоритети та плануючи майбутні роботи.

Особливо важливим є можливість формувати особисті враження та висновки про прочитані твори. Зберігаючи ці враження, користувач може періодично повертатися до них, щоб оновити пам'ять про свої знання та розуміння прочитаних робіт.

Тому використання інструментів для контролю та організації списку читання є важливим елементом у сьогодишньому дедалі різноманітнішому інформаційному світі. Інструменти допомагають нам не лише зберігати та структурувати знання, але й активно використовувати їх для розвитку та збагачення нашого світогляду.

1.2 Огляд аналогів

Перед створенням проєкту на тему відстеження списку читання, обов'язковим кроком є огляд аналогічних успішних проєктів.

Перше, на що варто звернути увагу, – це функціональність інших платформ. Визначення їх сильних та слабких сторін є важливим пунктом для подальшого аналізу та порівняння. Як приклад сильних сторін можна виділити інтуїтивно зрозумілий інтерфейс, можливість додавання різних форматів контенту та ефективну систему фільтрації та сортування. Як приклад слабких сторін можна виділити обмежену можливість ділитися списками літератури,

нестабільну роботу мобільних додатків і погану інтеграцію з іншими сервісами.

По-друге, важливо порівняти доступні функції з тими, що пропонують різні альтернативи. Якщо в одному проєкті відсутня функція, яка є стандартною на інших схожих платформах, це може бути гарною можливістю покращити функціонал. Наприклад, якщо проєкти пропонують можливість залишати коментарі або оцінювати літературу, то це один з елементів, який потенційно може привернути увагу майбутніх користувачів. Якщо ж інші проєкти з якоїсь причини не мають певних функцій, треба визначити, наскільки вони необхідні та чи впливають на зацікавленість у підсумку.

Першим проєктом для порівняння і, напевно, фаворитом серед всіх оберемо вебсайт Goodreads.

Goodreads – це американський вебсайт, який слугує платформою соціального каталогу книг, надаючи користувачам можливість шукати та переглядати різні книги, анотації, уривки та рецензії в базі даних Goodreads. Він використовується для створення особистих бібліотек і списків для читання, а також для того, щоб користувачі могли відстежувати свої літературні інтереси і плани [2].

Крім того, користувачі можуть створювати власні групи для обговорення книг, обміну рекомендаціями та думками, голосування, ведення блогів та участі в різноманітних дискусіях. Ці можливості роблять Goodreads не лише інструментом для ведення особистої історії читання, але й платформою для спілкування та обміну думками з іншими користувачами.

На сайті Goodreads користувачі можуть додавати книги на свої книжкові полиці, оцінювати книги, бачити, що читають їхні друзі та автори, брати участь у дискусіях та групах на різні теми, а також отримувати рекомендації щодо майбутніх книг на основі вже прочитаних. Додавши друзів до свого профілю, користувачі можуть переглядати їх книжкові полиці та рецензії, а також коментувати сторінки. Goodreads пропонує систему оцінювання де одна зірка є найгіршою оцінкою, а 5 зірок – найкращою, з

можливістю написання рецензії. На сайті є стандартні книжкові полиці, такі як "Книги, які я прочитав", "Книги, які я читаю" і "Книги, які я хочу прочитати", а також можливість для користувачів створювати власні полиці, щоб класифікувати свої книги за категоріями.

Користувачі отримують регулярну розсилку з новими книгами, книжковими рекомендаціями, інтерв'ю з авторами та віршами. Якщо користувач пише твір, він потрапляє на сторінку профілю автора, а також публікується в авторському блозі.

Користувачі можуть додавати один одного в друзі та обмінюватися відгуками, повідомленнями, рекомендаціями щодо книг, повідомленнями тощо. Goodreads присутній у Facebook, Pinterest, Twitter та інших соціальних мережах. Якщо підключити свій обліковий запис до соціальних мереж, таких як Facebook, з'явиться можливість імпортувати контакти зі своїх сторінок у соціальних мережах до Goodreads і розширити список друзів Goodreads. Існують також налаштування, які дозволяють Goodreads автоматично публікувати публікації в соціальних мережах, наприклад, сповіщати друзів у Facebook про прочитані користувачем книги або про оцінки, які користувач поставив книгам.

Goodreads має кілька переваг як інструмент для запису прочитаного. По-перше, він має велику кількість користувачів та інтегрований з Facebook, що дозволяє користувачам швидко знаходити своїх друзів на платформі. Це створює можливості для обміну пропозиціями та думками про книги. Це також полегшує пошук нових книг для читання на основі інтересів друзів та книг, які вони зараз читають або планують прочитати. Goodreads також сприяє взаємодії та обговоренню літературних творів, дозволяючи користувачам розпочинати розмови на основі коментарів своїх друзів. Платформа дозволяє користувачам відстежувати книги, які вони хочуть прочитати, але не мають, а також книги у власній бібліотеці.

Однак Goodreads має кілька недоліків. По-перше, він не може відстежувати інші матеріали, такі як відеоігри чи фільми. Це обмежує його

функціональність для тих, хто хоче керувати різними культурними інтересами на одній платформі. Крім того, відсутність налаштувань приватності може створити проблеми для тих, хто хоче контролювати доступ до своїх даних і рецензій. Крім того, Goodreads не пов'язаний з місцевими бібліотеками, тому необхідно вжити спеціальних заходів для перевірки наявності матеріалів у бібліотеках. Дизайн платформи можна вважати незручним і неефективним для використання на мобільних пристроях, що ускладнює її використання на ходу і в мобільних форматах.

На рис. 1.1 зображено інтерфейс вебсайту Goodreads.

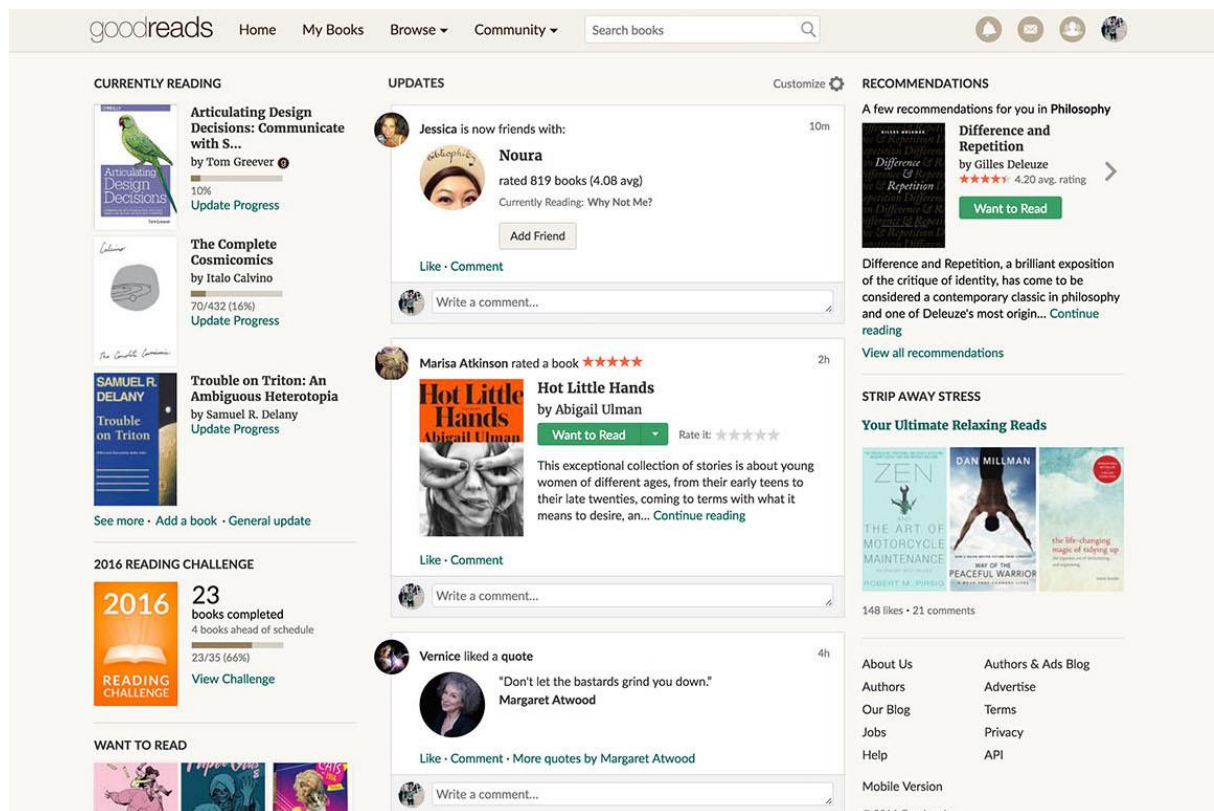


Рисунок 1.1 – Інтерфейс вебсайту Goodreads [3]

Наступним на розгляд проєктом і, напевно, головним конкурентом Goodreads є The StoryGraph.

TheStoryGraph – це вебплатформа соціального каталогу книжок, яка конкурує з Goodreads, яким володіє Amazon. StoryGraph набув популярності після того, як Book Riot опублікував статтю про переваги платформи, такі як персоналізовані рекомендації, налаштовані опції оцінювання та незалежність

від Amazon. StoryGraph використовує безкоштовну модель з деякими функціями, доступними лише через платну підписку [4].

У порівнянні з Goodreads, StoryGraph пропонує користувачам більш персоналізовані рекомендації та рейтинги. Його незалежність від Amazon також приваблює тих, хто хоче уникнути "амазонізації" книжкової індустрії і віддає перевагу альтернативним платформам. Однак, як і у випадку з багатьма безкоштовними сервісами, деякі функції доступні лише за умови сплати передплати [4].

The StoryGraph у багатьох відношеннях схожий на Goodreads, також будучи вебплатформою, призначеною для читачів з метою відстеження прочитаних книг і використання метаданих для створення профілів книг. На відміну від Goodreads, StoryGraph зосереджується на відстеженні книжкових колекцій і не має соціальної домашньої сторінки, як у Goodreads. Платформа StoryGraph дозволяє читачам реєструватися та оцінювати книги, спілкуватися з друзями на сторінках спільноти сайту та ставити читацькі завдання. Платформа надає книжкові рекомендації на основі аналізу читацьких звичок користувачів. Прокручуючи сторінку профілю, користувачі можуть оцінити свою онлайн-бібліотеку, класифікуючи її за настроєм, темпом, тривалістю, жанром і рейтингом. Ця функція може бути оновлена за щомісячну плату і пропонує більш розширену статистику; на відміну від Goodreads, StoryGraph пропонує можливість давати книгам півзірки або чверть зірки.

Тепер давайте розглянемо переваги The StoryGraph як інструменту для запису читання.

The StoryGraph пропонує унікальну систему каталогу на основі настрою, яка дозволяє знайти потрібну книгу відповідно до вашого настрою. Це дозволяє знайти потрібний літературний твір для будь-якого дня або настрою, спрощуючи процес пошуку нових творів.

Варто також відзначити, що StoryGraph дозволяє користувачам знаходити твори, яких немає в їхній бібліотеці. Це розширює кругозір і

дозволяє відстежувати не лише придбані книги, а й ті, які ви плануєте прочитати в майбутньому.

Платформа також надає статистику про ваш стиль читання, ваш настрій і швидкість, з якою ви любите читати. Це дозволяє краще зрозуміти ваші читацькі вподобання та швидше знаходити нові та цікаві книги.

Витончений дизайн StoryGraph доступний як на ПК, так і на мобільних пристроях, що робить його простим у використанні в будь-який час і в будь-якому місці.

Ще однією важливою особливістю є те, що ви можете отримувати рекомендації щодо наступної книги з вашого особистого списку "до прочитання". Це сприяє безперервному розвитку вашого списку читання і забезпечує постійний потік нової літератури.

Оглянемо недоліки StoryGraph. На відміну від Goodreads, StoryGraph не може відстежувати нічого, окрім книг, наприклад, відеоігри чи фільми. Це може бути обмеженням для тих, хто хоче відстежувати різні аспекти свого культурного життя на одній платформі.

Також немає налаштувань конфіденційності, що може бути проблемою для тих, хто хоче контролювати доступ до своїх даних і відгуків.

На рис. 1.2 зображено інтерфейс вебсайту The StoryGraph.

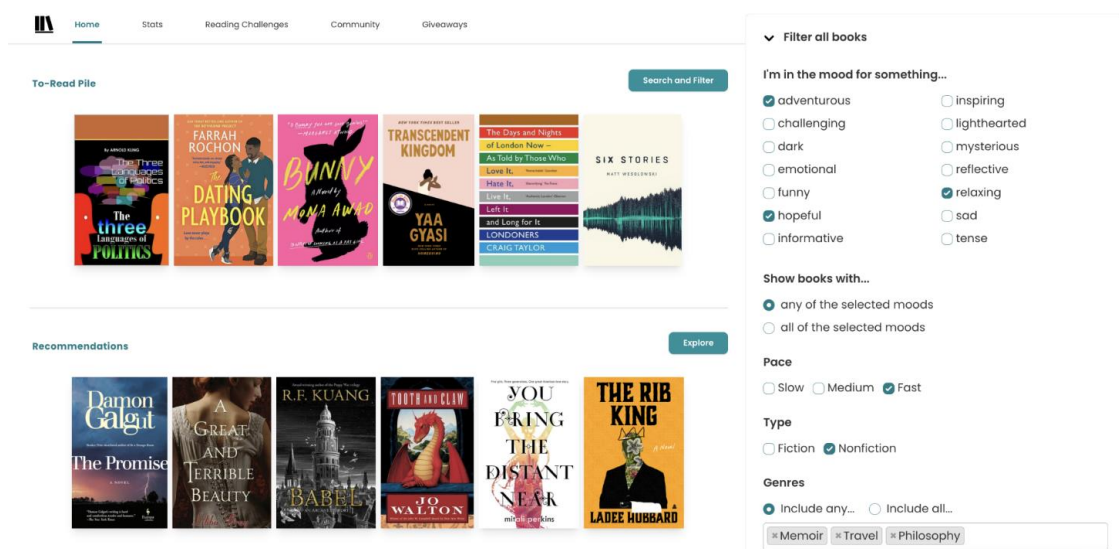


Рисунок 1.2 – Інтерфейс вебсайту The StoryGraph [5]

Останнім в огляді альтернатив розглянемо вебзастосунок LibraryThing.

LibraryThing – це вебзастосунок соціального каталогу, призначений для обміну, а також зберігання книжкових каталогів та різноманітних метаданих про книги. Його використовують автори, окремі користувачі, бібліотеки та видавці [6].

Основна функція LibraryThing – каталогізація книг, фільмів, музики та інших медіа шляхом передачі даних з бібліотек і шести магазинів Amazon.com через з'єднання Z39.50. Бібліотечні джерела надаватимуть LibraryThing записи у форматі Dublin Core та MARC. Користувачі можуть імпортувати інформацію з понад 2000 бібліотек, включаючи Британську бібліотеку, Національний каталог Канади, Бібліотеку Конгресу, Національну бібліотеку Австралії та Єльський університет. Якщо записів немає в жодному з цих джерел, інформацію про книгу можна ввести вручну, використовуючи чистий бланк [6].

Кожен твір може мати різні видання, переклади, друковані та аудіоверсії. Учасники заохочуються додавати загальнодоступні рецензії, описи, загальні відомості та іншу інформацію про твір. Рейтинги, колекції та теги допомагають у категоризації. Також заохочуються дискусії на форумі.

Теми класифікуються з використанням Десяткової системи класифікації Мелвілла. Вона базується на неавторському виданні класифікації Дьюї 1922 року, стандартному написанні назв розділів (на відміну від оригінальних назв, які були написані відповідно до рекомендацій реформи правопису Дьюї) та модифікаціях сучасної термінології [6].

LibraryThing дозволяє користувачам шукати книги, обирати улюблені та додавати їх до особистої бібліотеки. На головній сторінці є меню "Додати книгу", щоб спростити цей процес. Крім того, користувачі можуть вибрати джерело, з якого шукати книги. Наприклад, вони можуть обрати Бібліотеку Конгресу США або Amazon. Платформа має понад 700 джерел даних, які можна відфільтрувати за мовою та країною.

Сайт має відведену сторінку для статистики. Вона відображає кількість книг у вашій бібліотеці, статистику за оригінальними мовами (наприклад, кількість книг, які спочатку були написані французькою, іспанською, давньогрецькою тощо), стовпчикову діаграму середніх оцінок, стовпчикові діаграми років видання вашої бібліотеки та інше. Також вона відображає всі серії у вашій бібліотеці й показує, чи є ваша серія "завершеною".

На сторінці статистики також доступні такі можливості:

- можливість перегляду книг на основі подій у них, персонажів, місць. Наприклад, можна знайти книги, які містять події, що відбуваються в певному місці;
- можливість перегляду статистики за авторами, таку як статистика за статевою приналежністю, статусом життя (живий/померлий) та національності, розкладеної за секторами кругової діаграми.

На сайті спостерігається значне використання текстового матеріалу, що стосується як домашньої сторінки, так і інших сторінок ресурсу. Цей контент охоплює різноманітні аспекти, такі як презентація авторів, інформація для рецензентів, розіграші та інше. Проте для деяких користувачів ці дані можуть бути неактуальними та відволікати увагу від основного завдання. Крім того, звернення до текстового вмісту без належного оформлення може створювати враження застарілості та неадаптованості до сучасних стандартів дизайну.

Хоча існують можливості для налаштування домашньої сторінки, слід враховувати, що процес цього налаштування може вимагати часу та зусиль від користувача. Тому, незважаючи на наявність таких можливостей, складність використання вебсайту може впливати на загальний досвід користувача та вимагати додаткової уваги з боку користувача для його оптимізації.

Система сортування відгуків на платформі LibraryThing відкриває користувачам можливість здійснення більш ефективного пошуку та оцінки літературних творів. Можливість сортування за мовою дозволяє зосередитися на відгуках у власній мові, сприяючи зручності взаємодії з платформою для користувачів з різних культурних та мовних середовищ. Крім того, функція

сортування за часом публікації відгуків надає можливість користувачам отримувати актуальну інформацію про нові літературні твори та тренди.

Сортування відгуків за кількістю "голосів" дозволяє виділити найбільш важливі та впливові відгуки, що сприяє збагаченню користувачів авторитетною інформацією. З урахуванням меншої кількості користувачів на платформі, відгуки можуть бути менш масовими, проте це може забезпечити більшу увагу до кожного відгуку та більш індивідуалізований підхід до оцінки книги.

LibraryThing – це інструмент для серйозних користувачів, які прагнуть мати докладний, об'єктивний та розгорнутий каталог книг. Обговорення та групи на платформі відзначаються вищим рівнем інтелектуальності, і взаємодія з авторами менш виражена, що може розглядатися як позитивний або негативний аспект, залежно від вподобань користувача. LibraryThing сприймається як інструмент для читачів, а не як платформа для реклами та просування видавців і авторів. Його інтерфейс менш інтуїтивний, менш привабливий з естетичної точки зору та менш зручний для користування порівняно з іншими аналогічними сервісами. Проте, його система статистики та тегів є більш розвиненою. LibraryThing розроблено для тих, хто надає перевагу розміщенню своєї фізичної бібліотеки на перше місце, а спілкування з іншими читачами – на друге.

На рис. 1.3 зображено інтерфейс вебсайту LibraryThing.

В заключення, порівняємо три розглянуті сервіси, випишемо їх функціональні та інші параметри і спробуємо звести їх у порівняльну таблицю 1.1.

Виходячи з таблиці можна зробити висновки, що Goodreads представляє собою найпопулярніший та найзвичніший варіант з середнім функціоналом та оформленням. The StoryGraph є наймолодшим та найсучаснішим з усіх, втілює креативні нові ідеї та з'являється у новітньому легкому інтерфейсі, система настроїв книг виділяється з поміж кандидатів. LibraryThing, хоч і є менш популярним за інших, має застарілий дизайн,

незрозумілий інтерфейс, але представляє найширший функціонал з усіх, а також дає доступ до оцінки різного виду контенту.

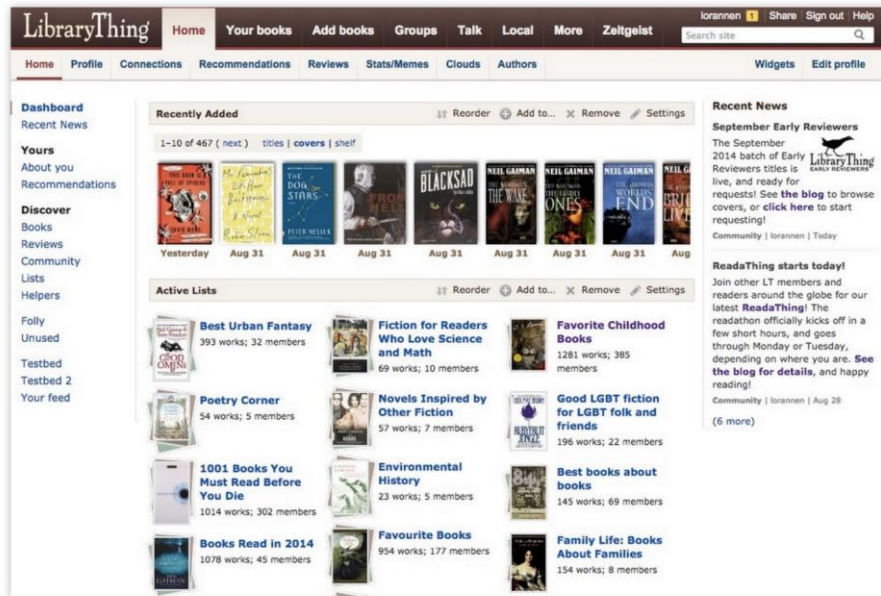


Рисунок 1.3 – Інтерфейс вебсайту LibraryThing [6]

Таблиця 1.1 – Порівняння альтернатив

Параметр	Goodreads	The StoryGraph	LibraryThing
1	2	3	4
Інтерфейс	Класичне стримане оформлення, але є простір для модернізації.	Сучасний, не навантажений	Застарілий, місцями не має оформлення
Оцінювання	П'ятизірковий варіант оцінювання, що може сковувати читачів.	Оцінювання п'ятизіркове, але є можливість ставити половину та чверть зірки.	П'ятизіркове оцінювання з можливістю ставити половину зірки.
Фільтрування	Пошук не має фільтрів, але після введення запиту можна буде обрати за яким критерієм шукати (книги, групи, цитати, люди, назва або автор).	Пропонуються фільтри за настроєм книги, її темпом, переліком жанрів, трьома типами розмірів та датою публікації.	Пошук не має початкових фільтрів, але результати запиту можна відфільтрувати за типом речі, що шукають.
Додаткові можливості пошуку	Після введення запиту, пропонуються певні результати на нього, а також, на основі цих результатів, користувачу пропонуються полиці жанрів, де він може знайти схожі роботи.	Додаткових можливостей немає, але це компенсується багатими фільтрами і великою системою настрою книг.	Сервіс має унікальну систему тегів, які кожен користувач сам додає до кожної книги. Є можливість шукати книги за цими тегами.

Продовження таблиці 1.1

1	2	3	4
Статистика	Надається статистика про середній рейтинг прочитаних книг, кількість книг та сторінок на рік.	Надається велика кількість статистики у вигляді діаграм, а саме настроїв книг, темпу, кількості сторінок, типу книг, жанрів, форматів, авторів, мовам, графіку кількості книг та сторінок та оцінкам	Надається широка статистика з діаграмами по типу книг, жанрам, інформації про авторів (живі/мертві, статі), власній оцінці, мові, обкладинці та її кольору.
Рецензія	Надається можливість поставити оцінку, написати ємну рецензію та призначити книгу на певні власні полиці за категоріями. Рецензії можна сортувати та фільтрувати за мовами.	Пропонує коротку вижимку думок читача, його оцінку. Також читач проходить коротке опитування для того, щоб сформувати категорії під які підпадає книга, її настрої.	Користувачі не обмежені в обсязі рецензії, призначається оцінка та користувач обирає теги до книги. Рецензії можна сортувати та фільтрувати за мовами.
Додавання нових книг	Потрібно створювати запит на додавання книги, якщо її неможливо зайти на сервісі.	Є можливість ручного додавання книги, яку потім можуть затвердити.	Є можливість додати книгу за номером або вручну, якщо він відсутній.

1.3 Постановка завдання

Після вивчення предметної області та аналізу наявних успішних реалізацій для вирішення її проблем, сформулюємо технічне завдання.

Метою цієї роботи є розробка програмного забезпечення у формі вебзастосунку, основною ціллю якого буде спрощення процесу відслідковування списку читання користувача, допомога в його структуризації та підбитті підсумків читання.

Основними функціональними вимогами до проєкту є:

- додавання нових книг до читацької бази;
- додавання та редагування будь-якої інформації про створені книги;
- розподілення книг за списками, які позначають поточний стан її читання;
- пошук книг у списках за назвою або автором;

- додавання рецензії на книгу, можливість редагування рецензії;
- можливість оцінити книгу при перенесенні її до стану прочитаних.

Проект повинен бути розроблений з можливістю подальшої модернізації та масштабування.

При створенні інтерфейсу важливо врахувати потреби користувача і переконатися, що користувачеві комфортно працювати з інтерфейсом. Це означає, що інтерфейс повинен бути не тільки простим і зрозумілим, але й ефективним. Він повинен дозволяти користувачам легко виконувати необхідні дії, не заважаючи їм зосередитися на основній меті використання програми або системи.

Занадто простий інтерфейс може справляти враження застарілого або не мати достатньої функціональності. Тому важливо знайти баланс між простотою і достатнім рівнем функціональності та дизайну, щоб він виглядав сучасним і привабливим для користувачів.

Однак також важливо не перевантажувати інтерфейс додатковими деталями, які можуть відволікати увагу користувача від основного завдання. Чистий і простий дизайн допомагає користувачеві зосередитися на важливих діях та інформації.

У склад програмної документації повинні входити такі елементи як опис програмного забезпечення та описи основних програмних модулів, тексти реалізації цих модулів.

Тестування повинно бути виконано з оглядом на те, що всі сценарії, які будуть надані користувачу повинні виконуватись без нарікань з боку застосунку.

1.4 Висновки з розділу 1

У цьому розділі була розглянута предметна область розроблюваного проєкту, а також проблеми, які має на меті вирішити цей проєкт. Досліджено актуальність даної роботи як у минулому, так і у майбутньому.

Для більш глибокого розуміння контексту були розглянуті три аналогічні, успішні проєкти. Проведений аналіз їх функціоналу, виділено сильні та слабкі сторони кожного з них. В результаті цього дослідження отримано базове розуміння для подальшого розвитку та вдосконалення розроблюваного проєкту.

Було створено технічне завдання до проєкту, в якому описано ключові функціональні можливості, які мають бути реалізовані в проєкті, вимоги до інтерфейсу, складові компоненти та програмне забезпечення, яке необхідно для редагування проєкту, його подальшої модифікації, вказані системні специфікації з якими створюється проєкт, описані складові програмної документації.

2 ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Вибір інструментів розробки

Як мову розробки функціональної частини вебзастосунку було обрано JavaScript.

JavaScript, у своєму статусі високорівневої скриптової мови програмування, широко використовується у веброзробці для створення динамічних інтерактивних вебзастосунків і сайтів. Завдяки своїй гнучкості і потужності, JavaScript забезпечує функціональність і зручність вебсторінок для користувачів.

Основні застосування JavaScript включають анімацію, яка прикрашає вебсторінки та забезпечує їх привабливість для відвідувачів, а також валідацію форм, яка гарантує правильність введених даних користувачем. Більш того, JavaScript може динамічно оновлювати вміст сторінки без перезавантаження, забезпечуючи плавний і неперервний перегляд.

Ще однією ключовою можливістю JavaScript є можливість реагувати на дії користувача, такі як наведення курсору миші на елемент або натискання на кнопку. Це дозволяє створювати інтерактивні елементи, які взаємодіють з користувачем, змінюючи свій вигляд або поведінку у реальному часі.

За опитуваннями розробників, JavaScript є не тільки найпопулярнішою мовою для веброзробки, але найпопулярнішою мовою в цілому серед всіх мов програмування. Це підтверджує дослідження Лайонела Вейлшері, що охоплює тренди мов програмування на 2023 рік. Статистика мов зі статті зображена на рис. 2.1.

Діаграма показує, що майже 64% опитуваних зазначили JavaScript, як свою основну мову програмування.

Якщо не брати до уваги мови HTML/CSS, що прийнято вважати мовами розмітки, а не програмування, а також Python, який, хоч і використовується для веброзробки, найбільше застосовується в датааналітиці,

то найпопулярнішою мовою після JavaScript у веброзробці є TypeScript з майже 39% відповідей серед опитуваних.

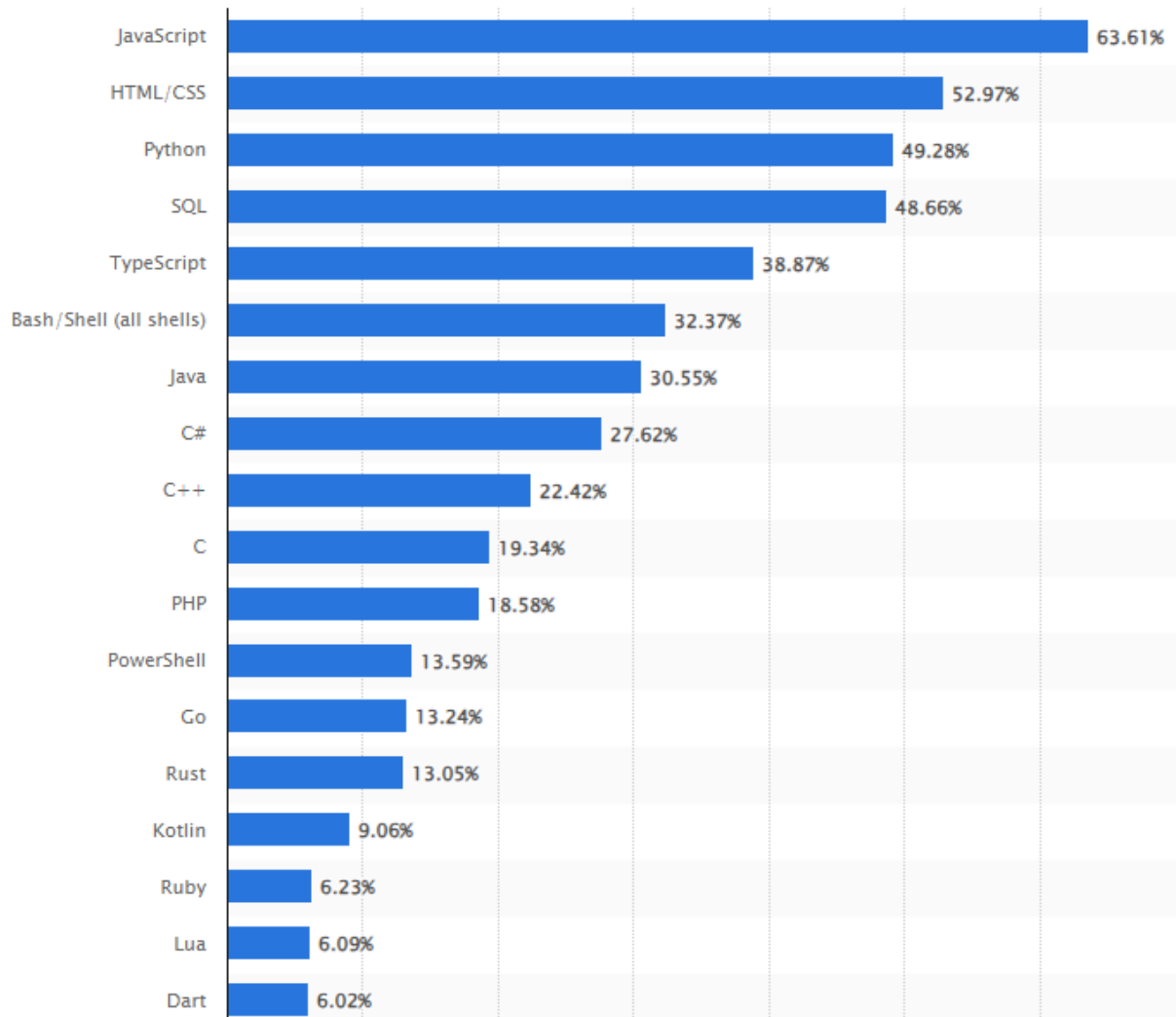


Рисунок 2.1 – Найпоширеніші мови програмування серед розробників у всьому світі станом на 2023 рік [7]

Мова TypeScript є відкритою мовою програмування, розробленою для створення великих і складних застосунків. TypeScript був створений одним із великих технологічних гігантів – компанією Microsoft – у 2012 році. Основною метою створення TypeScript було полегшення розробки та підтримки великих проєктів програмного забезпечення [8].

Основна особливість TypeScript полягає в тому, що він є синтаксичним розширенням мови JavaScript, але при цьому додає статичну типізацію. Це дозволяє розробникам забезпечити більшу стабільність та безпеку великих

програм, виявляючи помилки на етапі компіляції, що сприяє вирішенню проблем на ранніх етапах розробки.

Однією з ключових переваг TypeScript є можливість використання сучасних функціональних можливостей, таких як асинхронне програмування та робота з промісами, що спрощує розробку складних асинхронних застосунків.

Завдяки поєднанню можливостей JavaScript та переваг статичної типізації, TypeScript став популярним інструментом серед розробників для створення надійних та масштабованих програмних продуктів.

Порівнюючи TypeScript з JavaScript, варто зазначити, що TypeScript надає всі можливості JavaScript, а також додаткові можливості та функції, які полегшують розробку складних програмних продуктів. Оскільки TypeScript базується на JavaScript, весь коректний код JavaScript також є коректним у TypeScript, що робить TypeScript надмножиною JavaScript.

Можливість інтеграції з існуючим кодом JavaScript робить TypeScript привабливим варіантом для розширення функціональності існуючих проєктів. Крім того, TypeScript має додаткову функцію статичної типізації, що дозволяє виявляти помилки на етапі розробки, підвищуючи надійність та підтримку програмного забезпечення.

З використанням TypeScript розробник отримує ряд переваг. Уникнення прихованих помилок, таких як помилка "undefined" не є функцією. Код може бути перероблений зі збереженням його функціональності. Орієнтація в складних системах більше не є проблемою. За результатами досліджень, TypeScript виявляє понад 15% всіх помилок у JavaScript [8].

Динамічне типування часто спричиняє помилки, що знижують ефективність розробника та можуть ускладнити розробку через збільшення витрат на додавання нового коду.

Отже, через відсутність типів та перевірок помилок на етапі компіляції, JavaScript може бути невдалим вибором для серверного коду в організаціях та великих кодових базах.

TypeScript містить різні базові типи, такі як число, масив, кортеж, булевий тип, рядок і багато інших. Деякі з цих типів недоступні в JavaScript.

TypeScript підтримує як неявні, так і явні типи. Якщо ви не пишете типи явно, компілятор увімкне виведення типів, щоб визначити типи, які ви використовуєте.

З іншого боку, їх написання явно надає переваги, такі як допомога іншим членам команди, які читають код, і перевірка того, що те, що бачите ви, відповідає тому, що бачить компілятор.

У базовому JavaScript перевірка типів виконується під час виконання. Однак це призводить до збільшення часу виконання, якого можна уникнути, виконуючи перевірку під час компіляції.

TypeScript чудово працює у великих проєктах або коли багато розробників працюють разом, завдяки своїм сильним системам типів, які допомагають у виявленні помилок на етапі компіляції і полегшують спільну роботу над кодом. Однак, у невеликих проєктах з невеликою кількістю коду використання TypeScript може бути зайвим, оскільки воно може ускладнити процес розробки, додавши додаткову складність та затримки у випуску продукту. У таких випадках, використання JavaScript може бути більш виправданим, дозволяючи швидше розробляти та випускати програмне забезпечення.

Зведемо інформацію про ці мови програмування у таблицю 2.1, порівняємо їх параметри.

Таблиця 2.1 – Порівняння JavaScript та TypeScript

Параметр	TypeScript	JavaScript
1	2	3
Типізація	Сильна типізація. TypeScript підтримує як статичну, так і динамічну типізацію.	Слабка типізація. JavaScript підтримує лише динамічну типізацію.
Компіляція	TypeScript потребує компіляції.	JavaScript не потребує компіляції.

Продовження таблиці 2.1

1	2	3
Прив'язка даних	TypeScript використовує такі поняття, як інтерфейси та типи для визначення даних, що використовуються.	У JavaScript таких понять немає.
Крива навчання	TypeScript має жорстку криву навчання. Крім того, він потребує знань зі створення сценаріїв.	Це проста у вивченні і гнучка мова для написання вебскриптів.
Спільнота	TypeScript має меншу спільноту розробників програмного забезпечення.	JavaScript має велику спільноту розробників програмного забезпечення.
Створення прототипів	Функція прототипування доступна в TypeScript.	JS не підтримує прототипування.

Використання JavaScript для дипломного проєкту має свої переваги. Перш за все, це відповідає розміру проєкту. Для невеликого обсягу роботи, яким зазвичай є дипломна робота, TypeScript може виявитися занадто потужним і складним в управлінні. JavaScript, з іншого боку, забезпечує простоту і швидкість розробки завдяки своїй прямій і лаконічній синтаксичній структурі.

Крім того, використання JavaScript дозволяє використовувати широкий спектр доступних бібліотек і фреймворків. Це дає можливість розширити функціональність проєкту і використовувати вже наявні рішення для різноманітних задач, що спрощує процес розробки і зменшує час витрачений на написання коду.

Не останнім чинником є широка підтримка JavaScript у спільноті розробників. Завдяки своїй популярності, JavaScript має велику кількість активних розробників, які можуть надати підтримку, поради та відповіді на питання, що виникають під час роботи над проєктом.

Для створення вебзастосунку, обрання однієї лише мови програмування недостатньо для досягнення задовільного результату. Велику долю розробки займає також фреймворк, на якому побудовано застосунок.

В якості фреймворку для нашого проєкту обираємо React, який хоч формально і є бібліотекою до JavaScript, має функціонал дуже близький до альтернативних фреймворків і є найпопулярнішим варіантом для веброзробки.

React — JavaScript-бібліотека, що використовується для створення інтерфейсів користувача та відповідає за зовнішній вигляд вебзастосунку. Цей інструмент, розроблений корпорацією Facebook, є відкритим за кодом і активно розвивається спільнотою. Зокрема, React використовується в React Native — технології для створення нативних та гібридних додатків для операційних систем iOS та Android [9].

Бібліотека абсолютно не обмежує можливості в дизайні вебзастосунків. Якщо метою є створити зовсім особливий користувацький інтерфейс, то React найкраще з того, що можна обрати. Розробники лише вказують, як кожна деталь інтерфейсу повинна виглядати у всіх можливих варіантах, а бібліотека самостійно забезпечує їх втілення за вказівкою.

React не лише прикрашає вебзастосунок, але й забезпечує його продуктивність. Однією з ключових переваг бібліотеки є віртуальний DOM, який дозволяє оновлювати лише необхідні частини сторінки, зберігаючи в пам'яті вебзастосунку структуру попередньої версії для порівняння з новим станом інтерфейсу.

Обробка сторінки на серверній частині також має велике значення: завдяки йому швидкість, з якою компоненти з'являються на сторінці не залежить від потужності пристрою користувача.

React можна застосовувати у вебзастосунках будь-якого розміру, але він особливо ефективний у невеликих односторінкових проєктах з потенціалом для майбутнього зростання. Саме в таких випадках React виявляється найбільш вдалим вибором, зокрема, через його легку окупність, що приваблює стартапи. Крім того, вебсайти на React відзначаються високою оцінкою від пошукових систем, зокрема Google, завдяки їхньому зручному інтерфейсу та відмінному користувацькому досвіду.

Та все ж є вагомі причини, чому саме React обирають для створення вебзастосунків.

Перше, JavaScript XML – це розширений синтаксис, який спрощує роботу та полегшує виявлення помилок. Для розробників, що володіють JavaScript та HTML, які є простими та популярними мовами, об'єднання їх у JSX дає більше розуміння цих компонентів розробником. Це дає можливість уникати проблем з доопрацюванням, тестуванням та підтримкою коду.

Код, розроблений з використанням React, не лише легко піддається тестуванню, а й має потенціал для повторного використання. Завдяки декларативному підходу до представлення даних, він стає надзвичайно передбачуваним, особливо порівняно з альтернативними бібліотеками або фреймворками. Крім того, він надає можливість додавання власних станів до компонентів та ефективно інкапсулювати їх. JavaScript не обмежується шаблонами, що дозволяє компонентам обмінюватися даними з різних точок у програмі.

Засновники React пропонують розробникам використовувати бібліотеку настільки, наскільки це необхідно, і в тих місцях, де це необхідно. Наприклад у серверній частині, конкретніше в парі із Node.js, а також для створення мобільних додатків з використанням React Native. Крім того, можна легко додати декілька динамічних віджетів на HTML-сторінку, щоб зробити інтерфейс більш динамічним та сучасним. Якщо у майбутньому з'явиться потреба використовувати більше можливостей React, розширення його функціоналу не вимагатиме значних зусиль та серйозних змін у коді сторінки.

React є найпопулярнішим JavaScript-фреймворком. За опитуванням, його обирає понад 80% відсотків розробників. На рис. 2.2 зображено найпоширеніші JavaScript-фреймворки станом на 2022 рік.

З рисунку можна побачити, що основним конкурентом React є фреймворк Angular, який має популярність в районі 50%. Angular пропонує часто схожі, а інколи і зовсім інші рішення при створенні вебсайтів.

Проще кажучи, Angular – це інструмент для створення вебсайтів, з якими можна взаємодіяти. Це міцний фреймворк для фронтенду, який має всі необхідні засоби для швидкої розробки складних вебсайтів і застосунків [10].

Порівнюючи Angular і React, важливо враховувати ступінь їхньої складності. Angular, який є повноцінним фреймворком, може мати більшу крутість кривої навчання, особливо для початківців у веброботі. З іншого боку, React, спрямований на компоненти, надає більш плавний старт для новачків у цій галузі.

Кожен з цих фреймворків має свої особливості та унікальний стиль. Angular є потужним і повноцінним, ідеально підходить для великих проєктів і досвідчених команд. В свою чергу, React відомий своєю легкістю, гнучкістю і швидкістю, що робить його ідеальним вибором для невеликих за розміром проєктів, а також для команд, які віддають перевагу роботі з компонентами.

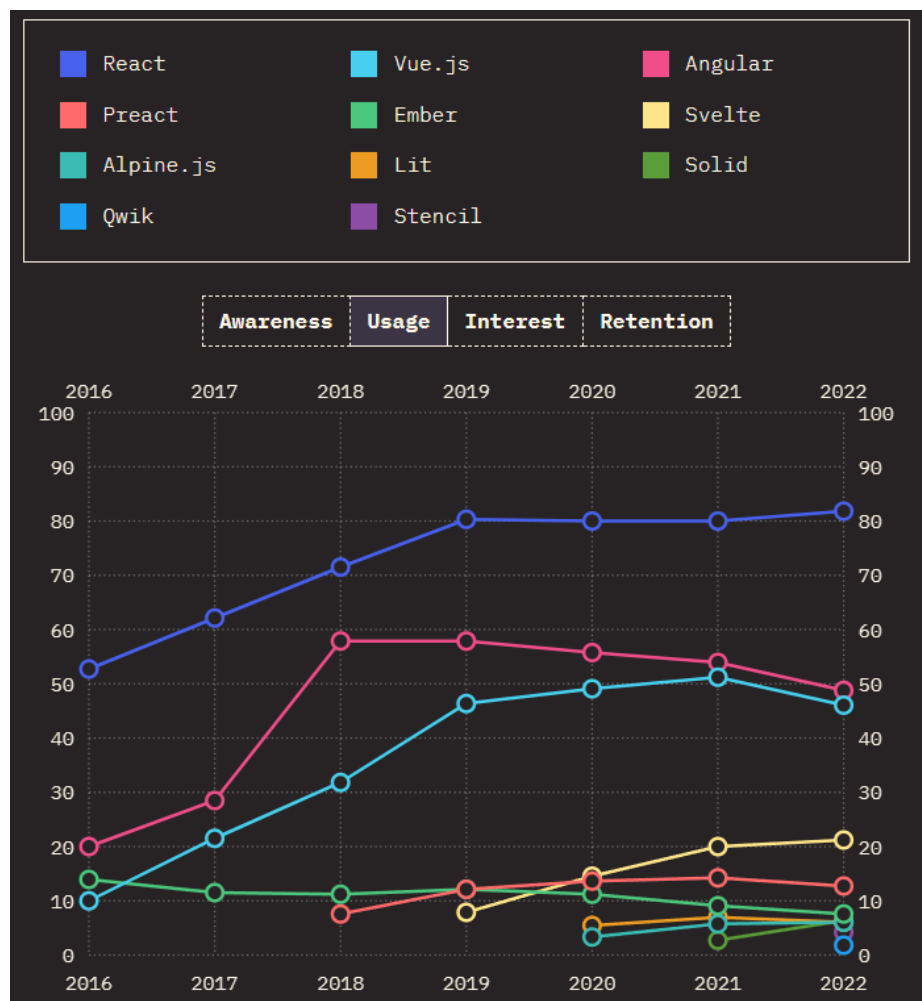


Рисунок 2.2 – Найпоширеніші JavaScript-фреймворки станом на 2022 рік [11]

Проведемо аналіз параметрів React та Angular, зведемо розглянуті параметри у порівняльну таблицю 2.2 [12].

Таблиця 2.2 – Порівняння Angular та React [12]

Параметр	Angular	React
Тип	Фреймворк	Бібліотека
Двостороннє зв'язування даних	Автоматична синхронізація даних інтерфейсу та коду	Необхідно використовувати додаткові інструменти
Компонентний підхід	Використовуються компоненти для створення користувацьких елементів	Компоненти – основа створення користувацького інтерфейсу
Складність початку роботи	Потрібен певний час для освоєння і налаштування	Простий старт і швидке створення компонентів з JSX
Розширена екосистема	Великий набір інструментів і бібліотек для різних завдань	Деякі функціональності вимагають додаткових бібліотек
Роутінг і управління станом	Вбудовані інструменти для маршрутизації та управління станом	Потрібно використовувати сторонні бібліотеки для роутингу та управління станом
Розмір і продуктивність	Більший розмір через включені інструменти, хороша продуктивність у великих додатках	Менший розмір, швидка і плавна робота, особливо на невеликих і середніх проєктах
Суспільство та підтримка	Велика й активна спільнота розробників і хороша підтримка	Величезна й активна спільнота, безліч рішень і допомоги

Отже, у підсумку можна сказати, що React є найбільш популярною бібліотека JavaScript. Дані з опитувань розробників на Stack Overflow, активність на GitHub, а також статистика пошукових запитів, – усе це підтверджує постійний ріст інтересу до цієї технології та збільшення попиту на розробників, які володіють нею. React, що можна описати як гнучкий, легкий та продуктивний, використовують компанії різного масштабу – від

Facebook до відомих платформ, таких як Netflix. Він ефективно адаптує контент для телевізорів з низькою продуктивністю, а також успішно впорається з викликами, такими як геолокація, у платформі Instagram.

Пакетний менеджер `npm` та виконувач пакетів `prx` використовуватимуться для початкової генерації та подальшої побудови вебзастосунку, підключенні бібліотек до нього, тестуванні. Їх встановлення відбувається разом із середовищем виконання `Node.js`.

`Node.js` є впливовою платформою, що відобразила революційні зміни у сфері розробки вебзастосунків за останні роки. Її ключова мета полягає в трансформації вже широко використовуваної мови програмування `JavaScript` в універсальну мову, яка може ефективно застосовуватися як для клієнтської, так і для серверної частини, відкриваючи нові перспективи для розробників у створенні вебзастосунків.

Основним принципом `Node.js` є модель асинхронного програмування. Це означає, що розробники можуть створювати ефективні та адаптивні програми, які можуть обробляти багато запитів одночасно. `Node.js` також відомий своєю простою та швидкою поведінкою, що дозволяє ефективно використовувати ресурси сервера.

Однією з ключових особливостей `Node.js` є його асинхронна подійно-орієнтована модель програмування. Замість того, щоб блокувати виконання коду під час очікування операцій введення-виведення, `Node.js` використовує неблокувальні операції, щоб дозволити програмам обробляти кілька запитів одночасно. Це робить `Node.js` дуже ефективним для створення масштабованих і адаптивних серверних програм [13].

У рамках переваг `Node.js` слід відзначити його розгалужену екосистему модулів, яка представлена вбудованим менеджером пакетів `npm`. `npm` є одним з найбільших та найбільш розгалужених сховищ програмного забезпечення, де розробники можуть широко користуватися тисячами попередньо зібраних модулів і бібліотек для максимального прискорення процесу розробки [13].

Node.js відмінно взаємодіє з іншими інструментами веброзробки, такими як Express.js, популярна структура для створення вебзастосунків у Node.js. Express.js надає набір інструментів і функцій, що роблять розробку вебзастосунків швидкою і зручною.

Однак середовище не обмежується розробкою вебсерверів. За допомогою Node.js можна створювати багато програм, включаючи вебсервери, мікросервіси, інструменти командного рядка та настільні програми. Node.js завоював заслужену репутацію однієї з найпопулярніших платформ для розробки серверних програм, завдяки своїй надзвичайній гнучкості та потужним функціональним можливостям.

Node.js має багато унікальних функцій, які роблять його привабливим для розробників. Опишемо деякі з основних функцій Node.js.

Висока швидкість виконання. Node.js базується на механізмі Google V8 JavaScript і забезпечує швидку та ефективну обробку коду. Це дозволяє Node.js швидко виконувати операції. Це особливо важливо при створенні високопродуктивних серверних додатків.

Універсальність і гнучкість. Node.js дозволяє розробляти програми на стороні сервера та клієнта, використовуючи JavaScript як мову програмування. Це дозволяє розробникам використовувати одну мову та обмінюватися кодом між сторонами сервера та клієнта. Завдяки такій універсальності Node.js полегшує розробку повноцінних вебзастосунків і спрощує командну взаємодію.

Node.js має значну кількість модулів і бібліотек у своїй екосистемі, які доступні за допомогою вбудованого менеджера пакетів npm. Розробники можуть використовувати ці ресурси для розширення функціональності своїх програм та прискорення процесу розробки.

Node.js сумісний з усіма сучасними ОС, включаючи Linux, MacOS та найпопулярнішу Windows, що дозволяє розробникам вибирати оптимальну операційну систему для свого проєкту. Ця універсальність дає вільну руку у виборі середовища для розгортання програмного забезпечення.

Розглядання альтернатив Node.js не є необхідним у цій роботі, оскільки Node.js має занадто широке призначення, як для фронтенд, так і для бекенд розробки. Його альтернативами виступали б окремі мови та окремі фреймворки цих мов, головне призначення яких була б підтримка серверної частини вебзастосунка, що не є основною темою проєкту.

Дані у застосунку повинні зберігатися у базі даних серверу, для того щоб користувач не повинен був зберігати їх самостійно та мав доступ до них в будь який час, з будь-якого пристрою, якщо той має доступ до бази. В якості системи керування для цієї бази вирішено використовувати MySQL.

MySQL – вільна система керування реляційними базами даних, яка була розроблена для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних вебсторінок, оскільки має чудову підтримку з боку різноманітних мов програмування [14].

MySQL часто рекомендується для малих і середніх застосунків. Код сервера може бути скомпільований для різних платформ. Максимальний функціонал сервера виявляється у системах сімейства UNIX, де підтримується багатопоточність, що сприяє підвищенню продуктивності.

Можливості сервера MySQL [14]:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Також, MySQL було обрано для цієї роботи з тієї причини, що саме MySQL є одною з найпопулярніших СКБД, а нещодавно була найпопулярнішою.

Ця СКБД може вже є і не найкращим варіантом, але, зважаючи на розмір проєкту відносно крупних компаній і рівню знань у галузі баз даних, її вибір є досить логічним, бо коли справа доходить до деталей, професіонали обирають інші системи, в той час як MySQL найчастіше обирали ті, хто вивчає SQL.

Розподіл баз даних за популярністю наведено на рис. 2.3.

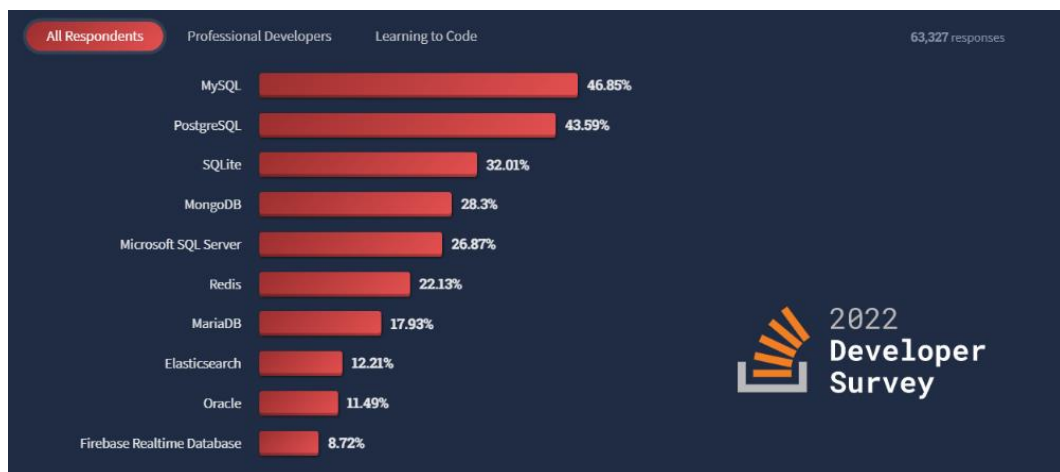


Рисунок 2.3 – Розподіл баз даних за популярністю [15]

2.2 Створення структурної схеми вебзастосунку

Перш ніж перейти до активної фази розробки вебзастосунку, важливо створити каркас, базову основу, яка визначає його структуру та функціональність. Структурні схеми, які ще називають вайрфреймами, допомагають зрозуміти взаємозв'язки між різними елементами вебзастосунку та організувати їх у найкращий спосіб. Цей етап важливий, оскільки він закладає основу для подальшої роботи і дозволяє розробникам зосередитися і полегшити робочий процес. Структурні схеми діють як візуальна інтерпретація концепції вебзастосунку, забезпечуючи єдність у сприйнятті завдань і допомагаючи уникнути несподіваних труднощів на більш пізніх етапах розробки.

Для створення вайрфрейму використовуватимемо сервіс Ninjamock, що безкоштовно надає достатній функціонал для цілей цієї роботи.

Усі вайрфрейми повинні реалізовувати раніше оголошені вимоги до застосунку, тому при створенні каркасів необхідно звертати на це найбільше уваги.

Почнемо створення з головної сторінки застосунку. За ідеєю проєкту, книга, як об'єкт, з яким взаємодіє користувач може мати три стани, а саме прочитана, читається зараз, або її читання заплановано на майбутнє. У застосунку усі книги повинні бути першочергово розподілені за списками, кожен з яких відповідає одному зі станів. Доступ до кожного цих списків повинен знаходитись на головній сторінці.

Окрім цього, сторінка повинна містити поле, за допомогою якого можна було б здійснювати пошук книг у цих списках за назвою, або автором. Сторінка повинна містити фільтри для елементів, що були знайдені пошуком, основні з них це жанри, сторінки, оцінка та наявність рецензії на книгу. Фільтрів може бути більше та вони можуть змінитися в процесі роботи, але цих буде цілком достатньо для представлення функціоналу фільтрів в цілому.

На сторінці списків, окрім самого перегляду книг у списках, повинна бути можливість змінювати їх приналежність до конкретного списку, а також елемент інтерфейсу, який би дозволяв додавати нові.

Додамо на сторінку елементи, що виконували б усі вище зазначені функції. Результат зображено на рис. 2.4.

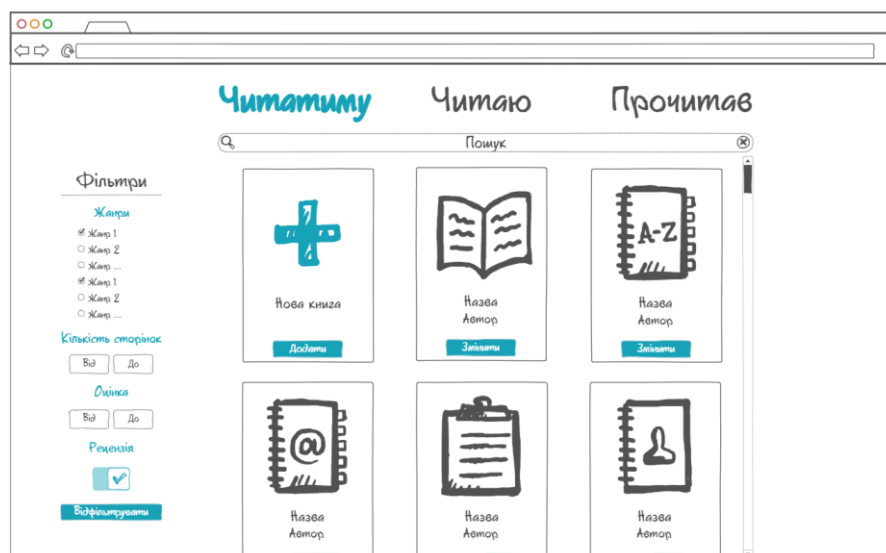


Рисунок 2.4 – Вайрфрейм сторінки списків

Після планування всього можливого функціоналу головної сторінки і додання його через елементи, необхідно спланувати інші сторінки, які б реалізовували незадіяний функціонал, що зазначено у вимогах, або логічно б продовжували вже описаний.

Наступною сторінкою, за таким принципом, створимо сторінку додання нової книги. Її ж можна буде використати пізніше для редагування інформації про вже існуючі книги.

Перехід на цю сторінку відбуватиметься, коли користувач захоче додати нову книгу, будучи на сторінці списків і натисне на елемент, який відповідає за створення.

На сторінці додавання нової книги для заповнення повинні бути доступні усі поля, які взагалі може мати книга як об'єкт, це і її назва, автор, жанри, які може містити книга, кількість сторінок, опис.

Надання книзі оцінки на етапі створення має сенс не в усіх випадках, бо користувач не має змоги оцінити книгу, яку він додає до списку тих, які він тільки бажає прочитати. Це поле є опціональним і дається для того випадку, якщо користувач захоче додати книгу яку він вже прочитав і хоче виставити їй оцінку без проведення декількох додаткових дій переходячи по різних сторінкам проекту. Окрім того, якщо використовувати цю форму повторно в редагуванні вже існуючих книг, які вже можуть мати оцінку, це рішення має ще більше сенсу.

Відповідно до того, що книга може мати оцінку в зазначених випадках, на етапі додання нової книги одразу впроваджуємо зручне для майбутнього користувача рішення вибору списку до якого додається книга, замість того щоб повертатися до списків, змінювати бажаний список і додавати до нього.

Додавання зображення іде окремим блоком, де надається можливість завантажити зображення обкладинки, побачити його вигляд на сторінці, або видалити, якщо користувача не влаштовує результат, який він бачить.

Результат описаного функціоналу у вигляді об'єктів на сторінці наведено на рис. 2.5.

Додавання До списків

Окладка

Завантажити зображення 100x300px

Завантажити

Видалити

Назва книги ...

Автор ...

Жанри

☒ Жанр 1 ☐ Жанр 2 ☐ Жанр ...

☒ Жанр 1 ☐ Жанр 2 ☐ Жанр ...

Кількість сторінок

255 - +

Оцінка (необов'язково)

★★★★☆

Опис

Додайте опис до книги

Читати

Читано

Против

Додати

Рисунок 2.5 – Вайрфрейм сторінки нової книги

Повертаючись до головної сторінки, наступна взаємодія, яку може провести користувач – переглянути детальну інформацію про будь яку книгу зі списків.

На сторінці з інформацією про конкретну книгу повинна бути наявна вся інформація, під яку ми створювали поля під час проєктування сторінки створення нової книги.

Як і раніше, користувачу буде набагато зручніше завжди мати можливість змінювати приналежність книги до стану на будь-якій сторінці, де він бачить книгу або інформацію про неї, це прискорить взаємодію з застосунком. Додаємо селектор і на цю сторінку.

В тому ж блоці, що і селектор, додаємо не менш важливі функції редагування та видалення книги, що переглядається. Якщо з видаленням все зрозуміло, воно повинно прибирати усі записи про цю книгу в застосунку, то редагування, в свою чергу, повинно переносити користувача до сторінки створення книги, за тим винятком, що в цьому випадку ця сторінка буде сторінкою редагування і її поля вже будуть заповнені даними з книги, яку користувач обрав для редагування.

Після всіх полів, що вже були розглянуті, ця сторінка міститиме дещо нове поле – «Рецензія». Це поле не додається при створенні нової книги, а залишається порожнім, користувач взаємодіє з ним тільки на цій сторінці, а саме редагувати його вміст. Поле містить оцінку, яку можна було бачити і раніше, і текст, що містить думки користувача про книгу, а не її загальний опис.

Вайрфрейм побудованої сторінки зображено на рис. 2.6.

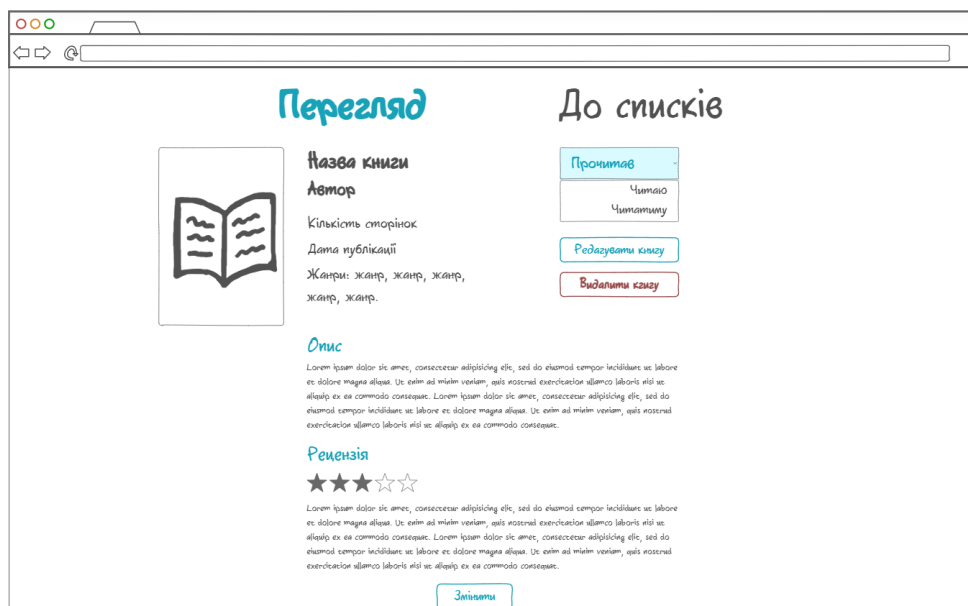


Рисунок 2.6 – Вайрфрейм сторінки інформації про книгу

Відходячи від головної сторінки, функціонал якої на даний момент було представлено на інших сторінках, що ще може змінитися вже на етапі не планування, а реалізації, переходимо до реалізації функціоналу інших сторінок, а саме сторінки інформації про книгу.

На сторінці перегляду інформації про конкретну книгу було описано таке поле як «Рецензія». Її додавання або редагування ще не було представлено на жодній сторінці проєкту, лише такий її елемент як оцінка можна було встановити раніше. Додамо ці функції на нову сторінку, яка буде повністю призначена цьому.

Сторінка повинна містити найголовнішу інформацію про книгу, щоб користувач не заплутався та не забув, до якої роботи він взагалі пише відгук.

Окрім цього, на сторінці повинен бути селектор вибору оцінки та поле для введення великого об'єму тексту для розгорнутої думки.

Розроблений вайрфрейм для рецензії зображено на рис. 2.7.

The wireframe shows a web browser window with a single tab. The address bar is empty. The page has two main sections: 'Рецензія' (Review) on the left and 'До списків' (To lists) on the right. Under 'Рецензія', there are labels for 'Назва книги' (Book name), 'Автор' (Author), 'Оцінка' (Rating), and 'Ваші враження' (Your impressions). The 'Оцінка' label is followed by a five-star rating system, with the first three stars filled. Below this is a large text area for 'Ваші враження' with the placeholder text 'Введіть свої враження від книги.' (Enter your impressions of the book.). At the bottom right of the form is a button labeled 'Зберегти' (Save).

Рисунок 2.7 – Вайрфрейм редагування рецензії

Розроблені вайрфрейми допомогли скласти приблизний зовнішній вигляд вебзастосунка та знайти місце для усіх його функціональних можливостей, які були зазначені у вимогах.

2.3 Висновки з розділу 2

У цьому розділі були розглянуті можливі альтернативи використаних засобів розробки, порівняно їх між собою і обґрунтовано використання обраних. Також було розроблено структурну схему застосунку у вигляді вайрфрейму.

3 ОСНОВНІ РІШЕННЯ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Функціональне проєктування

Перед початком фактичної розробки, необхідно більш конкретно представити загальну взаємодію користувача та застосунка, визначити їх функції. Гарним рішенням для відображення взаємодії стане розробка діаграми прецедентів.

Найголовнішою функцією користувача можна назвати перегляд списку книг на головній сторінці застосунку. Ця функція напряду пов'язана з функцією застосунку, що відповідає за відображення усіх книг з бази даних на сторінці.

Користувач може перемикає поточний список на один з трьох визначених: книги, які користувач планує читати в майбутньому, книги які користувач читає в поточний момент та книги, які користувач вже прочитав. Застосунок, в свою чергу, повинен при перемиканні списків виводити на сторінку лише ті книги, стан яких відповідає поточному списку.

Користувач може вводити запити у поле пошуку, на що застосунок повинен повертати лише ті книги, значення назви або автора яких містить слова з пошукового запиту користувача.

Користувач має можливість відфільтрувати книги, що виведені на сторінці. Це включає в себе такі фільтри як жанр, кількість сторінок, наявність оцінки та наявність рецензії. В залежності від виставлених фільтрів, застосунок повинен виводити на сторінку лише ті книги, поля яких підпадають під застосовані фільтри.

Користувач має можливість додати нову книгу до поточного списку. Застосунок, в свою чергу, повинен зберегти інформацію про додану книгу до бази даних та призначити створеній книзі ідентифікатор.

Користувач має можливість переглядати повну інформацію про обрану книгу. Застосунок повинен виводити на сторінку повну інформацію про книгу, що включає її назву, автора, жанр, сторінки, оцінку, рецензію, обкладинку.

Користувач має можливість редагувати інформацію про обрану книгу. Це включає в себе окреме редагування рецензії, редагування загальної інформації про книгу, додання нової, або видалення старої обкладинки книги. Застосунок повинен заносити відредаговані дані до полів цієї книги в базі даних. Також при видаленні обкладинки, застосунок повинен видаляти файл, що використовувався для неї з сервера. При додаванні нової обкладинки, застосунок повинен зберігати завантажений користувачем файл на сервері для подальшого відображення книги.

Користувач має можливість створити новий обліковий запис у застосунку. Застосунок повинен внести дані нового облікового запису до бази даних.

Користувач має можливість авторизуватися у застосунку, після чого в нього з'являється вийти з акаунту на будь-якій його сторінці. Застосунок повинен порівняти введені дані для входу з даними користувачів у базі даних і, при наявності введеного користувача, дати йому можливість далі працювати із застосунком.

Зобразимо описані взаємодії у вигляді діаграми прецедентів. Зображено на рис. 3.1.

Спроекуємо структуру вебзастосунку. Структура повинна відображати ієрархію елементів проєкту, їх знаходження відносно один одного. Також, на структурі повинно бути зображено, як елементи взаємодіють та звертаються один до одного. Елементи повинні мати реалізовані в них методи.

Розберемо побудовану структуру проєкту, опишемо її елементи їх взаємодію та основні методи.

Застосунок розділяється на клієнтську та серверну частини.

Серверна частина містить в собі скриптфайл під назвою `server.js`, який містить у собі усю логіку сервера. В ньому відбувається підключення до окремо локально піднятої бази даних. Розглянемо його методи.

`POST /items`. Відправляє на сервер масив книг для занесення його в таблицю бази даних.

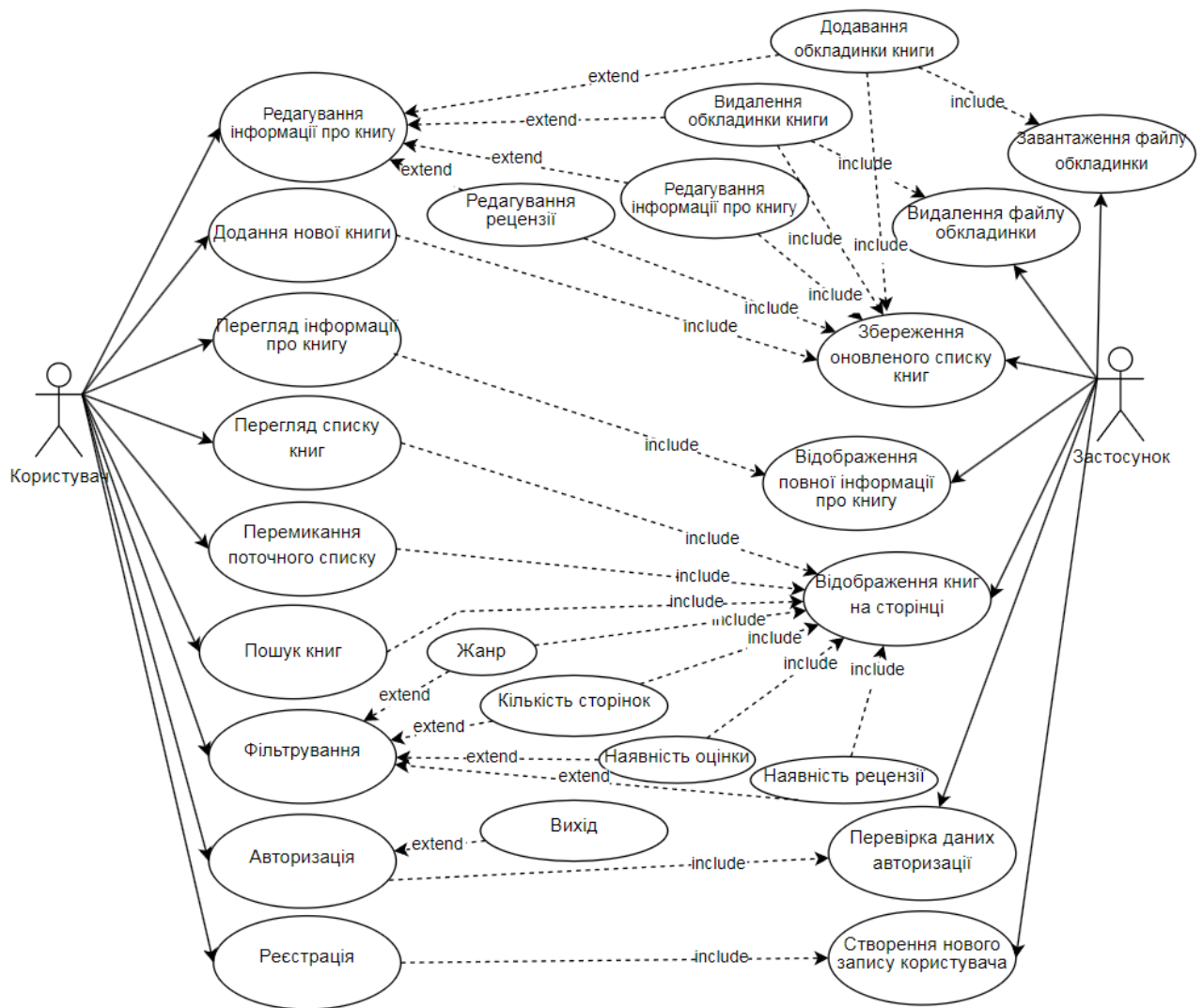


Рисунок 3.1 – Діаграма прецедентів

DELETE /items. Видаляє усі поля таблиці в базі даних, для того щоб оновити їх новим списком.

GET /items. Повертає усі поля таблиці з бази даних до клієнта.

POST /upload. Копіює завантажений користувачем файл зображення обкладинки до папки covers застосунку. У відповідь повертає посилання для доступу до цього файлу.

DELETE /delete-cover/:filename. Видаляє файл зображення обкладинки з папки covers застосунку.

Побудовану діаграму структури проєкту зображено на рис. 3.2.

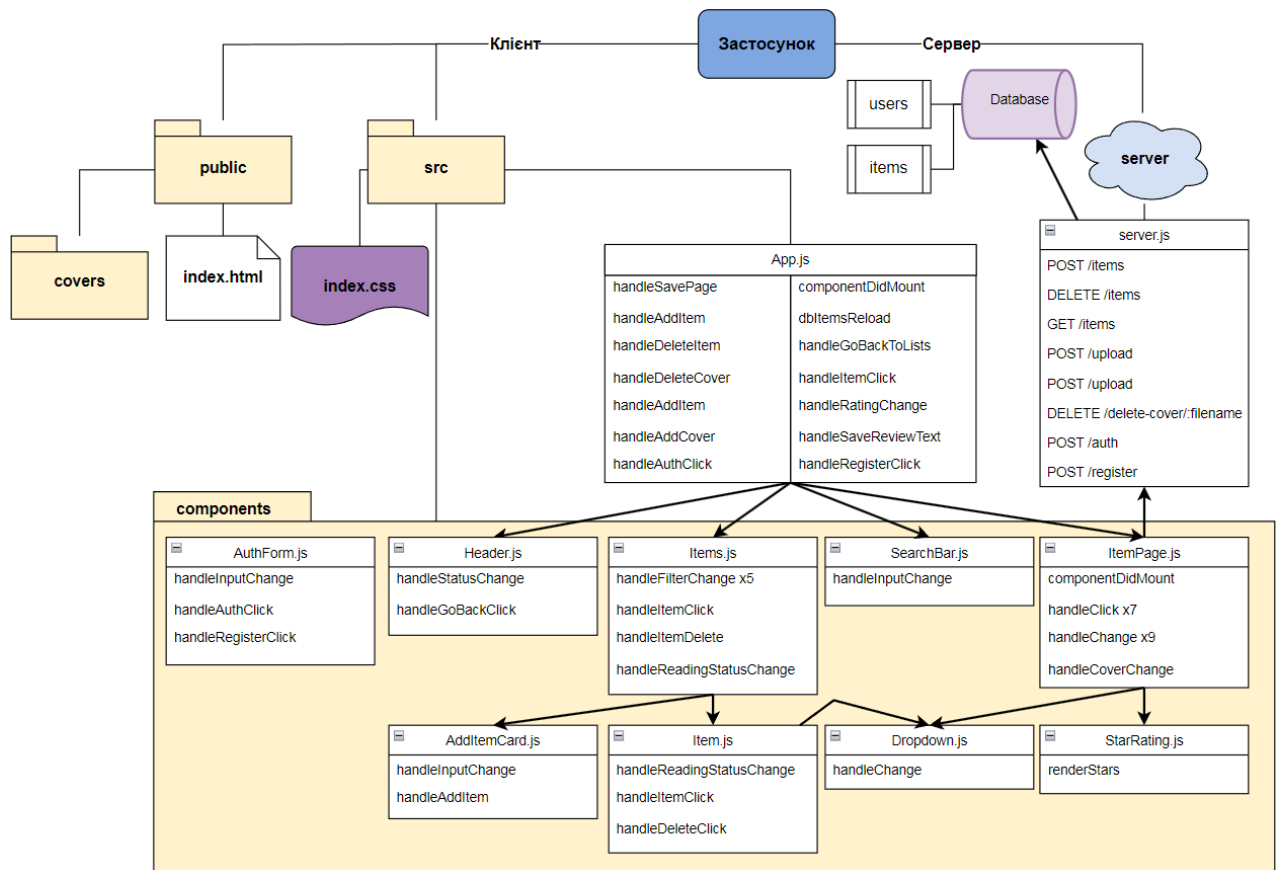


Рисунок 3.2 – Діаграма структури проєкту

POST /auth. Приймає запит від клієнтської частини з введеними логіном та паролем під час авторизації. Звертається до бази даних та перевіряє, чи існує користувач з такими даними. У відповідь до клієнта, повертає ідентифікатор цього користувача та його логін.

POST /register. Приймає запит від клієнтської частини з введеними логіном та паролем під час реєстрації. Звертається до бази даних та перевіряє, чи вже існує користувач з таким самим логіном, для того, щоб запобігти їх повторенню. Вставляє до таблиці користувачів бази даних нового користувача з даними, які сервер отримав від клієнтської частини.

Клієнтська частина складається з двох тек, а саме public та src.

Тека public містить у собі сторінку html, до якої застосунок виводитиме вміст, за допомогою скриптфайлів. До виведення цей файл містить базову структуру, але з боку користувача він є порожнім. Також ця тека містить у собі ще одну теку covers, яка зберігає у собі файли зображень обкладинок книг, які завантажує користувач під час редагування книг, а також декілька

системних обкладинок, які користувач може побачити на картці додання книги або якщо він створив книгу без обкладинки.

Тека `src` містить у собі головні функціональні елементи застосунку, а саме головний скриптфайл `App.js`, теку `components`, та файл стилів сторінки `index.css`.

В теці `components` знаходяться компоненти у вигляді скриптфайлів, які визиваються у головному скриптфайлі, серед них такі компоненти як `Header.js`, `Items.js`, `SearchBar.js`, `ItemPage.js`, `AddItemCard.js`, `Item.js`, `Dropdown.js`, `StarRating.js`.

Опишемо методи та взаємодію усіх скриптфайлів від найнижчого рівня ієрархії до найвищого.

`StarRating.js`. Компонент відповідає за повернення 5 зірок оцінки і викликається в компоненті `ItemPage.js`. При створенні приймає значення оцінки.

Метод `renderStars` в циклі проходить 5 ітерацій і повертає золоті зірки кількістю прийнятого значення оцінки, а ті зірки, що залишилися повертає срібними.

`Dropdown.js`. Компонент відповідає за повернення спадаючого списку з вибором стану читання для книги. Використовується компонентами `ItemPage.js` та `Item.js`.

Метод `handleChange` повертає значення, що було обрано в списку до батьківського компонента.

`AddItemCard.js`. Компонент повертає картку створення нової книги. Використовується в компоненті `Items.js`.

Метод `handleAddItem` передає до батьківського компонента новий елемент масиву з заповненими полями назви книги та ім'я автора.

`SearchBar.js`. Компонент повертає поле пошуку. Використовується в компоненті `App.js`.

Метод `handleInputChange` передає до батьківського компоненту текст поля при його зміні.

Header.js. Компонент повертає шапку сторінки та приймає обрану книгу. Якщо книгу не обрано, то шапка містить три кнопки для перемикання між списками читання. Якщо книгу обрано, то шапка змінюється на кнопку повернення назад.

Метод `handleStatusChange` змінює поточний список та передає це значення батьківському компоненту.

Метод `handleGoBackClick` викликає метод батьківського компонента для виходу з перегляду обраної книги.

Item.js. Компонент повертає картку книги, що містить кнопку видалення, зображення, назву, автора та селектор стану читання. Використовується в компоненті `Items.js`.

Метод `handleReadingStatusChange` встановлює книзі статус зі спадаючого списку та повертає значення батьківському компоненту.

Метод `handleItemClick` повертає батьківському компоненту яку книгу встановити обраною.

Метод `handleDeleteClick` повертає батьківському компоненту ідентифікатор книги, яку треба видалити при натисканні на кнопку видалення.

Items.js. Компонент повертає блок фільтрів на сторінку, блок, що містить картки книг. Відповідає за відображення головної інформації на сторінці. Фільтрує книги, що бачить користувач за списком, пошуковим запитом і блоком фільтрів. Компонент викликається в головному скриптфайлі `App.js`.

Компонент містить 5 обробників подій для фільтрів.

Метод `handleReadingStatusChange` оновлює масив книг, змінюючи в ньому статус у книги, статус якої змінив користувач.

Декілька методів передаються від компонентів нащадків до батьківського.

ItemPage.js. Компонент повертає сторінку інформації про обрану користувачем книгу. На сторінці є можливість завантажити нову обкладинку, видалити стару, змінити будь-яку інформацію про обрану книгу, додати

рецензію та виставити оцінку. Компонент викликається в головному скриптфайлі App.js.

Компонент містить 9 обробників подій для зміни полів інформації та 7 обробників натискань на кнопки. Основна їх суть в тому, щоб змінювати поточний стан полів книги на сторінці, а також передавати ці значення до головного скриптфайлу для оновлення даних.

Метод `handleCoverChange` спрацьовує після того як користувач завантажує файл на сторінці. Метод викликає батьківський метод видалення обкладинки, а потім звертається до сервера за допомогою методу `POST /upload` і з його відповіді викликає батьківський метод встановлення нової обкладинки для книги з отриманим посиланням на завантажений файл.

`AuthForm.js`. Компонент повертає форму з полями для вводу логіну та паролю. Форма призначена для виконання авторизації та реєстрації.

Методи цього компоненту призначені для того, щоб змінювати поточний стан полів логіну та паролю та повертати їх до головного скриптфайлу для подальшої роботи.

`App.js`. Головний скриптфайл, відповідає за роботу застосунку і викликає усі компоненти, передаючи у них необхідні для роботи значення або методи. Якщо не вибрано певну книгу, то відображає списки з усіма книгами, якщо ж книгу обрано, тоді повертає користувачу сторінку з детальною інформацією про цю книгу.

Метод `dbItemsReload` звертається до сервера два рази для видалення старих даних та завантаження нових, змінених. Використовується в інших методах для оновленні бази даних.

Метод `componentDidMount` спрацьовує після повного завантаження сторінки. Звертається до сервера і отримує від нього таблицю з книгами для відображення.

Метод `handleItemsChange` працює як `dbItemsReload`, але оновлює масив `items` на клієнтській частині, коли отримує відповідь від сервера.

Метод `handleGoBackToLists` прибирає статус обрання книги та викликає `dbItemsReload`.

Метод `handleItemClick` призначає натиснутій картці статус активної книги.

Методи `handleRatingChange`, `handleSaveReviewText`, `handleSavePage` отримують від компонентів зміну книги та оновлюють масив книг спочатку на клієнті, а потім викликає `dbItemsReload`.

Метод `handleAddItem` генерує новий ідентифікатор і створює нову книгу в масиві в клієнті, а потім викликає `dbItemsReload`.

Метод `handleDeleteItem` фільтрує масив, прибираючи з нього книгу з отриманим ідентифікатором, викликає `dbItemsReload`.

Метод `handleDeleteCover` звертається до сервера, щоб видалити файл зображення обкладинки поточної книги з сервера, встановлює порожнє поле обкладинки в клієнті і викликає `dbItemsReload`.

Метод `handleAddCover` встановлює обраному елементу посилання на завантажену обкладинку в поле обкладинки, оновлює масив книг в клієнті, викликає `dbItemsReload`.

Метод `handleSearchTextChange` встановлює в клієнті пошуковий запит з пошукового рядку для фільтрування.

Метод `handleAuthClick` Робить запит до серверної частини, передаючи їй логін та пароль для авторизації. Після відповіді, змінює стан поточного користувача та його ідентифікатора в клієнтській частині.

Метод `handleRegisterClick` Робить запит до серверної частини, передаючи їй логін та пароль для реєстрації нового користувача. Після відповіді, змінює стан поточного користувача та його ідентифікатора в клієнтській частині. Також робить запит на авторизацію в новий обліковий запис з тими самими даними.

3.2 Проєктування бази даних

Спроекуємо структуру бази даних, що складається з таблиці користувачів та таблиці книг. Таблиці пов'язані між собою полем ідентифікатору користувача, що у таблиці книг відповідає за те, якому користувачу з таблиці користувачів належить книга з таблиці книг.

Структуру бази даних зображено на рис. 3.3.

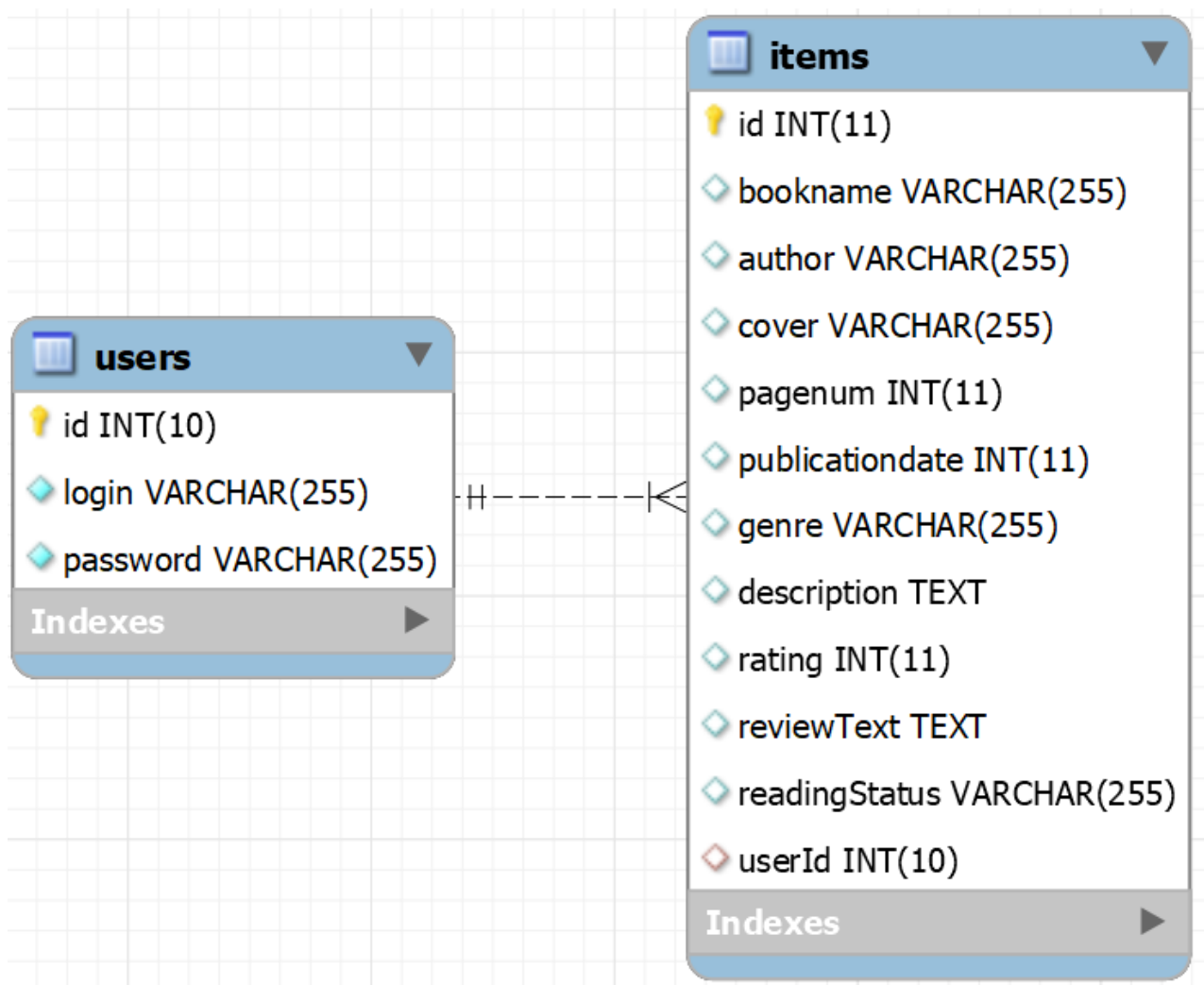


Рисунок 3.3 – Структура бази даних

Опишемо структуру таблиці книг **items** в базі даних. Структуру наведено у табл. 3.1.

Таблиця 3.1 – Структура таблиці items бази даних

Поле	Тип	Розмір	Коментар
id	INT	11	Ідентифікатор
bookname	VARCHAR	255	Назва
author	VARCHAR	255	Ім'я автора
cover	VARCHAR	255	Шлях до файлу обкладинки
pagenum	INT	11	Кількість сторінок
publicationdate	INT	11	Рік публікації
genre	VARCHAR	255	Жанр
description	TEXT	65535	Опис
rating	INT	11	Оцінка
reviewText	TEXT	65535	Текст рецензії
readingStatus	VARCHAR	255	Стан читання

Опишемо структуру таблиці користувачів users в базі даних. Структуру наведено у табл. 3.2.

Таблиця 3.2 – Структура таблиці users бази даних

Поле	Тип	Розмір	Коментар
id	INT	11	Ідентифікатор
login	VARCHAR	255	Логін
password	VARCHAR	255	Пароль

3.3 Проєктування інтерфейсу

Інтерфейс повинен бути не тільки простим і зрозумілим, але й ефективним. Він повинен дозволяти користувачам легко виконувати необхідні дії, не заважаючи їм зосередитися на основній меті використання програми або системи.

При розробці інтерфейсу вебзастосунку, була мета зробити його максимально прозорим та зрозумілим для користувача. Інтерфейс не повинен містити зайвих елементів, використання яких вводило б користувача у ступор. Усі елементи повинні бути очевидно відокремлені один від одного і самі по собі не представляти складності при користуванні. Інтерфейс повинен мати єдиний стиль оформлення, палітру.

Розглянемо основні взаємодії користувача з інтерфейсом.
На рис. 3.4 зображено сторінку авторизації та реєстрації.



Рисунок 3.4 – Сторінка авторизації та реєстрації

На рис. 3.5 зображено головну сторінку вебзастосунку.



Рисунок 3.5 – Головна сторінка

На рис. 3.6 зображено приклад взаємодії з карткою нової книги.



Рисунок 3.6 – Взаємодія з карткою нової книги

На рис. 3.7 зображено результат створення нової книги.

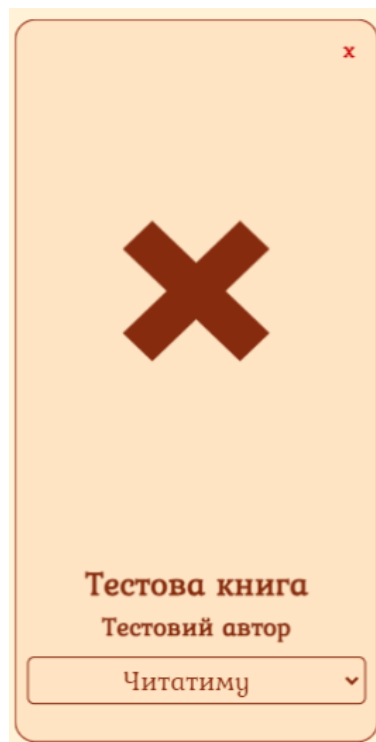


Рисунок 3.7 – Результат створення нової книги

На рис. 3.8 зображено взаємодію з селектором станів читання книги.



Рисунок 3.8 – Селектор станів читання

На рис. 3.9 зображено блок фільтрів та взаємодію з ним.

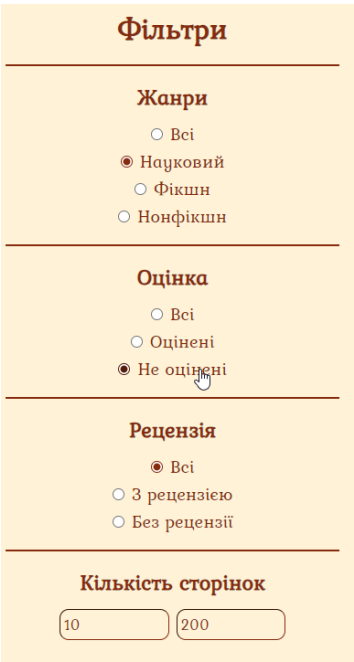


Рисунок 3.9 – Блок фільтрів

На рис. 3.10 зображено взаємодію з перемикачем списків книг.



Рисунок 3.10 – Перемикач списків книг

На рис. 3.11 зображено сторінку детальної інформації про обрану книгу.



Рисунок 3.11 – Сторінка детальної інформації про обрану книгу

На рис. 3.12 зображено редагування основної інформації про книгу.



Рисунок 3.12 – Редагування основної інформації про книгу

На рис. 3.13 зображена зміна оцінки книги.

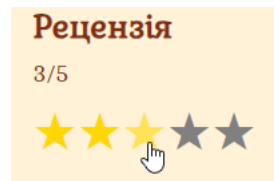


Рисунок 3.13 – Зміна оцінки книги

На рис. 3.14 зображено редагування рецензії.



Рисунок 3.14 – Редагування рецензії

На рис. 3.15 зображено додавання нової обкладинки.

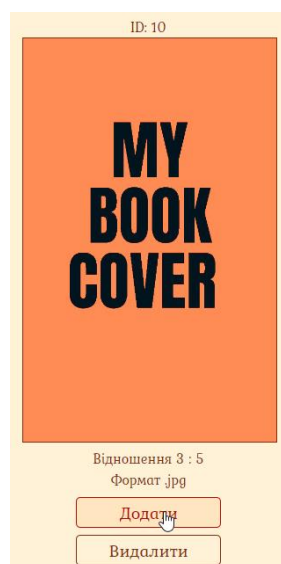


Рисунок 3.15 – Додавання нової обкладинки

3.4 Висновки з розділу 3

У цьому розділі було представлено загальну взаємодію користувача та застосунка, визначено та розібрано їх функції. Для цього було побудовано діаграму прецедентів. Також було спроектовано структуру застосунку та описано усі елементи цієї структури, розписано компоненти та їх основні методи. Було описано структуру бази даних, та її таблиць, їх поля, розміри та типи, опис.

Окрім цього, було описано підходи при створенні інтерфейсу та продемонстровано усі основні його елементи, з якими взаємодіє користувач.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ

4.1 Опис проєкту

Ціллю цього проєкту є створення вебзастосунку, призначенням якого є спрощення процесу відслідковування списку читання користувача, допомога в його структуризації та підбитті підсумків читання.

Розроблений вебзастосунок надає користувачу можливості переглядати книги, з якими він взаємодіє, розподіляти їх за станами читання, додавати свої думки та оцінки до книг, які він вже прочитав. Окрім цього, застосунок дозволяє оновлювати дані про книги, або розширювати наявну про них інформацію в будь-який час для зручності користувача.

4.2 Умови функціонування проєкту

Оптимальні системні вимоги до серверної частини, що будуть прийнятними для стабільного функціонування проєкту, але не є мінімальними або обмежуючими:

- операційна система сімейства Windows;
- 4-ядерний процесор з тактовою частотою 3.2 МГц;
- ОЗП ємністю 16 ГБ;
- npm версії 10.5.0;
- node версії 20.12.2.

Оптимальні системні вимоги до клієнтської частини:

- монітор роздільною здатністю 1920x1080 пікселів;
- будь-який вебпереглядач на основі Chromium версії 124.

Дотримання заявлених оптимальних вимог рекомендується для отримання взаємодії з застосунком, що запланована розробником і оптимізована під ці параметри.

4.3 Розгортання проєкту

Для розгортання проєкту, персональний комп'ютер на якому він розгортається повинен мати встановлене середовище виконання Node.js та менеджер пакетів npm. Для цього потрібно завантажити Node.js версії 20.12.2 з офіційного сайту, а менеджер пакетів npm буде встановлено разом з ним. Проєкт завантажується з GitHub-репозиторію.

Клієнтська частина запускається за допомогою консолі з кореневої папки проєкту, для цього прописуємо `npm run start` як команду у консоль. Пакетний менеджер виконує скрипт запуску застосунку за локальною адресою з портом 3000.

Серверна частина також запускається за допомогою консолі, але з кореневої папки треба поглибитись до папки `server` і виконати команду `node server.js`. Сервер запускається за локальною адресою з портом 3001 та робить спробу з'єднання з локальною базою даних за портом 3307.

База даних MySQL запускається будь-яким сервісом для запуску вебсерверів, в нашому випадку використовуємо XAMPP. Обираємо тип модуля MySQL, встановлюємо в параметрах порт 3307 та запускаємо локальний сервер для бази даних.

4.4 Експлуатація проєкту

Для того, щоб почати роботу з проєктом, необхідно виконати етап розгортання, що був описаний в минулому підрозділі і у вебпереглядачі звернутися за адресою <http://localhost:3000/>.

Перше, що користувач бачить перед собою – форма авторизації та реєстрації застосунку. Форма містить два поля, для того щоб ввести логін та пароль, та дві кнопки, в залежності від того, що користувач хоче зробити. Користувач може або створити новий обліковий запис з введеними логіном та паролем, або спробувати авторизуватись у вже існуючий.

Після авторизації або реєстрації, користувача зустрічає головна сторінка, на якій одразу можна побачити головне меню, в якому користувач може керувати списком книг, що відображається. Під головним меню знаходиться пошуковий рядок, за допомогою якого можна серед всього вмісту списку знайти необхідні книги. З лівого боку сторінки знаходиться блок фільтрів, який надає користувачу можливість керувати вмістом списку, що відображається, застосовуючи до нього бажані фільтри. Більшу частину сторінки займає контейнер, що містить у собі відображення поточного списку книг у вигляді карток з їх обкладинкою, назвою та автором. Ці картки надають можливість змінити приналежність книги до списку прямо на них, або взагалі видалити книгу цілком, видаливши цю картку при натисканні на хрестик у кутку. Окрім цього, в будь-якому списку, перша картка завжди відповідає за додання нової книги до списку. Картка створення нової книги має поля для назви та автора нової книги, але вони не є обов'язковими для створення, оскільки книги мають унікальні ідентифікатори та можуть мати одного автора чи однакову назву.

Для перегляду повної інформації про книги користувач має обрати одну з книг у списку та натиснути на її обкладинку, після чого відкриється сторінка, що містить детальну інформацію про обрану книгу. Сторінка має блок з обкладинкою книги, блок основної інформації, що містить назву, ім'я автора, кількість сторінок, рік публікації, жанр та опис книги, блок оцінки та рецензії на книгу.

Якщо користувач прагне редагувати інформацію про обрану книгу, або це нова книга, що потребує заповнення полів, сторінка має інструменти призначені для цієї задачі.

Керування обкладинкою відбувається в блоці під нею. Обкладинку можна завантажити у форматі jpg при натисканні на кнопку «Додати», вона, бажано, має бути співвідношенням сторін 3 до 5, інакше її зовнішній вигляд на сторінці інформації та сторінці списків буде викривлений. Якщо користувача

не влаштовує завантажена обкладинка, то її можна видалити кнопкою «Видалити».

З правого боку сторінки, користувачу доступний селектор для зміни стану читання поточної книги, не виходячи назад до списків.

Зміна інформації про книгу відбувається натисканням на кнопку «Редагувати» з правого боку сторінки. Після її натискання поля з інформацією стають доступними до редагування та обводяться контуром для розуміння того, що це поля, на місце жанру з'являється селектор з можливими варіантами. Після заповнення полів інформацією, користувач може зберегти результат натисканням на кнопку «Зберегти», що з'являється на місці кнопки «Редагувати» на час редагування.

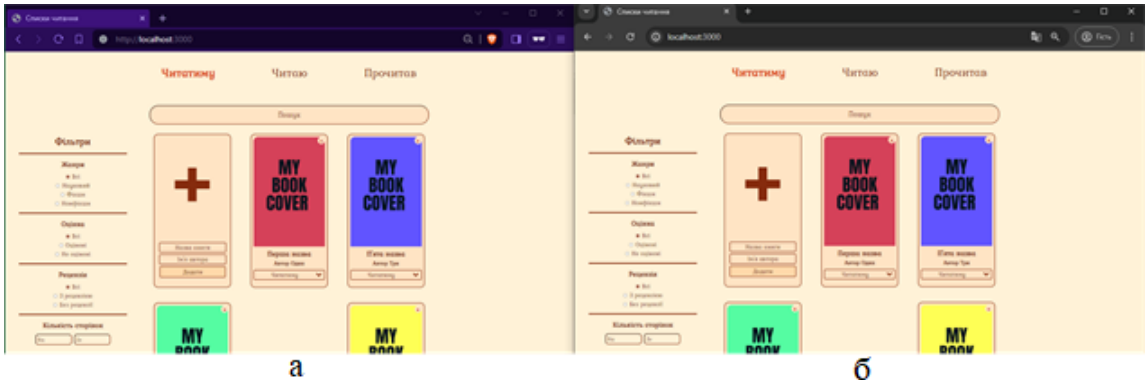
Під блоком основної інформації міститься блок з оцінкою та рецензією користувача на книгу. Користувач може виставити оцінку, натиснувши на необхідну йому кількість зірок, ніякого підтвердження для цього не потрібно. Редагування тексту рецензії потребує натискання на кнопку «Редагувати» під нею, після чого поле окреслюється контуром та стає доступним для написання думок про книгу. При натисканні на кнопку «Зберегти», редагування завершується.

Візуальний супровід до взаємодії з проєктом можна побачити на рисунках в попередньому розділі, а саме в підрозділі, що пов'язаний з інтерфейсом.

4.4 Тестування проєкту

Проведемо тестування розробленого проєкту у двох вебпереглядачах, а саме Brave та Google Chrome. Сформуємо спільний чекліст для перевірки функціональних можливостей, верстки та дизайну. Перевіримо сформований чекліст у кожному з вебпереглядачів та порівняємо отримані результати з очікуваними.

На рис. 4.1 зображено відображення головної сторінки вебзастосунку у двох вебпереглядачах паралельно. Результати ідентичні, що підтверджує те, що верстка та дизайн вебзастосунку відображаються коректно.



а – сторінка у Brave; б – сторінка у Google Chrome
Рисунок 4.1 – Головна сторінка у Brave та Google Chrome

В таблиці 4.1 наведено сформований чекліст.

Таблиця 4.1 – Чекліст

Параметр, що перевіряється	Google Chrome 124.0.6367.208	Brave 1.65.133 124.0.6367.208
Перегляд списку книг	Пройдено	Пройдено
Перемикання між доступними списками	Пройдено	Пройдено
Пошук за автором або назвою	Пройдено	Пройдено
Фільтрування за жанрами	Пройдено	Пройдено
Фільтрування за наявністю оцінки	Пройдено	Пройдено
Фільтрування за наявністю рецензії	Пройдено	Пройдено
Фільтрування за кількістю сторінок	Пройдено	Пройдено
Створення нової книги	Пройдено	Пройдено
Видалення картки книги	Пройдено	Пройдено
Зміна стану читання на картці книги	Пройдено	Пройдено
Перегляд сторінки детальної інформації	Пройдено	Пройдено
Редагування детальної інформації	Пройдено	Пройдено
Встановлення оцінки	Пройдено	Пройдено
Редагування рецензії	Пройдено	Пройдено
Зміна стану читання на сторінці інформації	Пройдено	Пройдено
Видалення книги на сторінці інформації	Пройдено	Пройдено
Додання нової обкладинки до книги	Пройдено	Пройдено
Видалення обкладинки книги	Пройдено	Пройдено

4.5 Висновки з розділу 4

У цьому розділі було описано призначення розробленого проєкту, його можливості. Було зазначено оптимальні умови, за яких проєкт може функціонувати без помилок. Описано процес розгортання проєкту, налаштування системи для його запуску та описано процес запуску локальних серверів для кожної його частини. Описано взаємодію користувача з проєктом та можливості, які має користувач під час цієї взаємодії, що супроводжується візуальними поясненнями. Проведено тестування застосунку у двох вебпереглядачах та визначено, що верстка, дизайн та функціональні можливості працюють коректно.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано предметну область розробки вебзастосунку для відстеження користувацького списку читання. Визначено актуальність книжкової тематики, а відповідно і засобів категоризації та сортування книг.

Було проведено огляд альтернатив у вигляді успішних проєктів на цю тематику, серед них Goodreads, TheStoryGraph та LibraryThing. Визначено слабкі та сильні сторони цих альтернатив, описано їх основний функціонал. Відмінністю розробленого застосунку від них можна назвати відкритий доступ до створення власних об'єктів книг, їх редагування, додання власних обкладинок з боку користувача, що є дуже ускладненим процесом в розглянутих аналогах, який займає багато часу, підтверджень та перевірок.

Функціональну частину розробки проєкту для відстеження користувацького списку читання було виконано мовою JavaScript з використанням бібліотеки React для побудови інтерфейсу та середовищем виконання Node.js для розгортання клієнтської та серверної частин проєкту. Зберігання даних реалізовано шляхом впровадження до проєкту бази даних, що підключена до сервера. Сервер має методи отримання і маніпулювання цими даними, які може використовувати клієнт, що до нього звертається. Для реалізації бази даних було використано СКБД MySQL.

В результаті отримано вебзастосунок, що надає користувачу можливості переглядати книги, з якими він взаємодіє, розподіляти їх за станами читання, додавати свої думки та оцінки до книг, які він вже прочитав. Окрім цього, застосунок дозволяє оновлювати дані про книги, або розширювати наявну про них інформацію в будь-який час для зручності користувача.

Розроблений застосунок сприяє покращенню досвіду читання як в плані хобі, так і в дослідницьких цілях, оскільки робить великий внесок у формування остаточних думок про прочитані праці та їх систематизацію.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Taycher L. Books of the world, stand up and be counted! All 129,864,880 of you. Inside Google Books. [Electronic resource] / L. Taycher. – Access mode: <https://booksearch.blogspot.com/2010/08/books-of-world-stand-up-and-be-counted.html>.
2. Goodreads. Вікіпедія. [Електрон. Ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Goodreads>.
3. Pinchuk M. New Goodreads Homepage Now Rolling Out to Everyone! Goodreads. [Electronic resource] / M. Pinchuk. – Access mode: <https://www.goodreads.com/blog/show/670-new-goodreads-homepage-now-rolling-out-to-everyone>.
4. The StoryGraph. Wikipedia, the free encyclopedia. [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/The_StoryGraph.
5. The StoryGraph. [Electronic resource]. – Access mode: <https://www.thestorygraph.com/>.
6. LibraryThing. Wikipedia, the free encyclopedia. [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/LibraryThing>.
7. Vailshery L. S. Most used languages among software developers globally 2023. Statista. [Electronic resource] / L. S. Vailshery. – Access mode: <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/>.
8. Raval N. TypeScript vs JavaScript: Which One Is Better to Choose? Radixweb. [Electronic resource] / N. Raval. – Access mode: <https://radixweb.com/blog/typescript-vs-javascript>.
9. Що таке Реакт? web-developer.in.ua. [Електрон. Ресурс]. – Режим доступу: <https://web-developer.in.ua/assets/articles/react/react-introduction/react-introduction.html>.
10. Angular для інтерактивних вебсайтів. Web-developer.in.ua. [Електрон. Ресурс]. – Режим доступу: <https://web-developer.in.ua/angular/>.

developer.in.ua/assets/articles/angular/%D0%B0ngular-interactive/%D0%B0ngular-interactive.html.

11. State of JavaScript 2022: Front-end Frameworks. stateofjs. [Electronic resource]. – Access mode: <https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/>.

12. Angular vs React: як обрати відповідний фреймворк під проєкт. FoxmindEd. [Electronic resource]. – Access mode: <https://foxminded.ua/angular-vs-react/>.

13. Все що потрібно знати про Node.js: особливості, характеристики, область застосування. Wezom. [Електрон. Ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/vse-chto-nuzhno-znat-o-nodejs>.

14. MySQL. Вікіпедія. [Електрон. Ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/MySQL>.

15. Ostrowska K. Top 5 Most Popular Databases in 2023. LinkedIn. [Electronic resource] / K. Ostrowska. – Access mode: <https://www.linkedin.com/pulse/top-5-most-popular-databases-2023-learnsql-com>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

Проект – Розробка вебзастосунку для відстеження користувацького списку читання. Проект призначений для покращення та структуризації хобі читання, створення списку прочитаного, відслідковування поточної зайнятості та планування читання майбутньої літератури.

A.2 Підстава для розробки

Дипломна робота на здобуття освітнього ступеня бакалавра має тему "Розробка вебзастосунку для відстеження користувацького списку читання", затверджену наказом Національного університету «Запорізька політехніка» №168 від 24 квітня 2024 року.

A.3 Призначення розробки

Основною ціллю роботи є спрощення процесу відслідковування списку читання користувача, допомога в його структуризації та підбитті підсумків прочитаного матеріалу.

Користувач зможе переглядати книги, з якими він взаємодівав, розподіляти їх за станами читання, додавати свої думки та оцінки до книг, які він вже прочитав.

A.4 Вимоги до розробки проекту

A.4.1 Вимоги до функціоналу

Функціонал застосунку має бути реалізований у скрипт-файлах мовою JavaScript з використанням бібліотеки React і її інструментів.

Основними функціональними вимогами до проекту є:

- додавання нових книг до читацької бази;

- додавання та редагування будь-якої інформації про створені книги;
- розподілення книг за списками, які позначають поточний стан її читання;
- пошук книг у списках за назвою або автором;
- додавання рецензії на книгу, можливість редагування рецензії;
- можливість оцінити книгу при перенесенні її до стану прочитаних.

A.4.2 Вимоги до інтерфейсу

Інтерфейс повинен бути не тільки простим і зрозумілим, але й ефективним. Він повинен дозволяти користувачам легко виконувати необхідні дії, не заважаючи їм зосередитися на основній меті використання програми або системи.

Проект у вигляді вебзастосунку має бути представлений у вигляді HTML-сторінки, оформлений за допомогою CSS. Компоненти на HTML-сторінках будуть створюватися мовою JSX, що поєднує в собі об'єкти, які відображуються та логічні або функціональні елементи.

A.4.3 Вимоги до зберігання даних

Інформація про книги, які користувач додає до вебзастосунку повинна зберігатися в базі даних, що керується системою MySQL. Безпека даних забезпечується тим, що база розгортається локально і зовнішні користувачі не мають можливості підключитися та змінювати її.

A.4.4 Технічні вимоги

Технічними вимогами для підтримання проекту є:

- операційна система Windows 10;
- процесор Intel Core I7-3770;

- ОЗП ємністю 16 ГБ;
- монітор роздільною здатністю 1920x1080;
- будь-який вебпереглядач на основі Chromium версії 124;
- npm версії 10.5.0;
- node версії 20.12.2;
- редактор початкового коду Visual Studio Code.

Працездатність проєкту, можливість його покращення гарантується на заявлених технічних вимогах або близьких до них, але цілком можлива з іншими складовими, но новіших або старіших версіях забезпечення та систем.

ДОДАТОК Б
Текст програми

Б.1 Текст файлу App.js

```

import React from "react"
import Header from "../components/Header";
import Items from "../components/Items";
import ItemPage from "../components/ItemPage";
import SearchBar from "../components/SearchBar";
import AuthForm from "../components/AuthForm";

import axios from "axios";

const dbItemsReload = (updatedItems) => {
  axios.delete('http://localhost:3001/items')
    .then(response => {
      console.log("Старі дані успішно видалено");
      axios.post('http://localhost:3001/items', updatedItems)
        .then(response => {
          console.log("Нові дані успішно вставлено");
        })
        .catch(error => {
          console.log("Помилка вставлення нових даних:", error);
        });
    })
    .catch(error => {
      console.log("Помилка видалення старих даних:", error);
    });
}

class App extends React.Component {
  constructor(props){
    super(props);
    this.state = {
      items: [],
      activeStatus: 'Читатиму',
      selectedItem: null,
      searchText: "",
      currentUser: null,
      userId: null
    }
  }

  componentDidMount() {
    axios.get('http://localhost:3001/items')
      .then(response => {
        this.setState({ items: response.data });
      })
      .catch(error => {
        console.log(error);
      });
  }

  handleStatusChange = (status) => {

```

```

    this.setState({ activeStatus: status });
  };

  handleItemsChange = (updatedItems) => {
    axios.delete('http://localhost:3001/items')
      .then(response => {
        console.log("Старі дані успішно видалено");
        axios.post('http://localhost:3001/items', updatedItems)
          .then(response => {
            console.log("Нові дані успішно вставлено");
            this.setState({ items: updatedItems });
          })
          .catch(error => {
            console.log("Помилка вставлення нових даних:", error);
          });
      })
      .catch(error => {
        console.log("Помилка видалення старих даних:", error);
      });
  };

  handleGoBackToLists = () => {
    this.setState({ selectedItem: null });
    const updatedItems = this.state.items;
    dbItemsReload(updatedItems);
  };

  handleItemClick = (clickedItem) => {
    this.setState({ selectedItem: clickedItem });
  };

  handleRatingChange = (updatedItem) => {
    const updatedItems = this.state.items.map(item =>
      item.id === updatedItem.id ? updatedItem : item
    );
    dbItemsReload(updatedItems);
    this.setState({ items: updatedItems });
  };

  handleSaveReviewText = (updatedItem) => {
    const updatedItems = this.state.items.map(item => {
      if (item.id === updatedItem.id) {
        return updatedItem;
      }
      return item;
    });
    dbItemsReload(updatedItems);
    this.setState({ items: updatedItems });
  };

  handleSavePage = (updatedItem) => {
    const updatedItems = this.state.items.map(item => {
      if (item.id === updatedItem.id) {

```

```

        return updatedItem;
    }
    return item;
  });
  dbItemsReload(updatedItems);
  this.setState({ items: updatedItems });
};

handleAddItem = (newItem) => {
  const maxId = this.state.items.length > 0 ? Math.max(...this.state.items.map(item => item.id)) : 0;
  const updatedItems = [...this.state.items, {
    id: maxId + 1,
    ...newItem,
    cover: "",
    pagenum: "",
    publicationdate: "",
    genres: "",
    description: "",
    rating: 0,
    reviewText: "",
    readingStatus: this.state.activeStatus,
    userId: this.state.userId
  }];
  dbItemsReload(updatedItems);
  this.setState({ items: updatedItems });
};

handleDeleteItem = (itemId) => {
  const updatedItems = this.state.items.filter(item => item.id !== itemId);
  dbItemsReload(updatedItems);
  this.setState({ items: updatedItems });
};

handleDeleteCover = () => {
  const { selectedItem } = this.state;
  const filename = selectedItem.cover.split('/').pop();

  axios.delete(`http://localhost:3001/delete-cover/${filename}`)
    .then(response => {
      console.log('File deleted successfully:', response.data);
      selectedItem.cover = "";
      const updatedItems = this.state.items.map(item => {
        if (item === selectedItem) {
          item.cover = selectedItem.cover;
        }
        return item;
      });
      dbItemsReload(updatedItems);
      this.setState({ selectedItem });
    })
    .catch(error => {
      console.error('There was a problem with your Axios request:', error);
    });
};

```

```

};

handleAddCover = (coverURL) => {
  const { selectedItem, items } = this.state;
  const updatedItems = items.map(item => {
    if (item === selectedItem) {
      item.cover = coverURL;
    }
    return item;
  });
  dbItemsReload(updatedItems);
  this.setState({ items: updatedItems });
};

handleSearchTextChange = (text) => {
  this.setState({ searchText: text });
};

handleAuthClick = (login, password) => {
  axios.post('http://localhost:3001/auth', { login, password })
    .then(response => {
      if (response.data.success) {
        this.setState({ userId: response.data.userId, currentUser: response.data.login });
      } else {
        this.setState({ userId: null, currentUser: null });
        alert("Введені дані є некоректними");
      }
    })
    .catch(error => {
      console.error('Error authenticating:', error);
    });
}

handleRegisterClick = (login, password) => {
  if (login !== "" && password !== ""){
    axios.post('http://localhost:3001/register', { login, password })
      .then(response => {
        if (response.data.success) {
          alert("Створено нового користувача " + login);
          axios.post('http://localhost:3001/auth', { login, password })
            .then(response => {
              if (response.data.success) {
                this.setState({ userId: response.data.userId, currentUser: response.data.login });
              } else {
                this.setState({ userId: null, currentUser: null });
                alert("Введені дані є некоректними");
              }
            })
            .catch(error => {
              console.error('Error authenticating:', error);
            });
        } else {
          alert("Помилка реєстрації");
        }
      })
  }
}

```

```

    }
  })
  .catch(error => {
    alert("Користувач вже існує");
    console.error('Error registering user:', error);
  });
} else {alert("Дані введено некоректно");}
}

handleLogoutClick = () => {
  this.setState({
    activeStatus: 'Читатиму',
    selectedItem: null,
    searchText: "",
    userId: null,
    currentUser: null
  });
  window.location.reload();
}

render(){
  const { selectedItem, searchText, currentUser, userId } = this.state;

  if (currentUser){
    if (selectedItem === null){
      return (
        <div className="wrapper">
          <div className="logoutdiv">Вітаємо, {currentUser}! <span className="logoutbutton"
onClick={this.handleLogoutClick}>Вийти</span></div>
          <Header
            onChange={this.handleStatusChange}
            selectedItem={selectedItem}
            onGoBackToLists={this.handleGoBackToLists}
          />
          <SearchBar
            onSearchTextChange={this.handleSearchTextChange}
          />
          <Items
            items={this.state.items}
            onItemsChange={this.handleItemsChange}
            activeStatus={this.state.activeStatus}
            onItemClick={this.handleClick}
            onAddItem={this.handleAddItem}
            onDeleteItem={this.handleDeleteItem}
            searchText={searchText}
            userId={userId}
          />
        </div>
      );
    } else {
      return (
        <div className="wrapper">

```

```

        <div className="logoutdiv">Вийти, {currentUser}! <span className="logoutbutton"
onClick={this.handleLogoutClick}>Вийти</span></div>

        <Header
          onChange={this.handleStatusChange}
          selected={selectedItem}
          onGoBackToLists={this.handleGoBackToLists}
        />

        <ItemPage
          selected={selectedItem}
          onRatingChange={this.handleRatingChange}
          onSaveReviewText={this.handleSaveReviewText}
          onSavePage={this.handleSavePage}
          onDeleteItem={this.handleDeleteItem}
          onGoBackToLists={this.handleGoBackToLists}
          onAddCover={this.handleAddCover}
          onDeleteCover={this.handleDeleteCover}
          onReadingStatusChange={this.handleStatusChange}
        />
      </div>
    );
  }
} else {
  return(
    <div className="wrapper">
      <AuthForm
        onAuthClick={this.handleAuthClick}
        onRegisterClick={this.handleRegisterClick}
      />
    </div>
  );
}
}
}
}

```

```
export default App;
```

Б.2 Текст файла index.js

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

```

Б.3 Текст файлу server.js

```

const express = require('express');
const app = express();
const mysql = require('mysql2');
const cors = require('cors');
const fs = require('fs');
const multer = require('multer');
const path = require('path');

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use('/covers', express.static(path.join(__dirname, '../public/covers')));

const db = mysql.createConnection({
  user: 'root',
  host: 'localhost',
  password: '',
  database: 'booksschema',
  port: 3307
});

app.post('/items', (req, res) => {
  const items = req.body;

  db.query(
    'INSERT INTO items (id, bookname, author, cover, pagenum, publicationdate, genre, description, rating, reviewText, readingStatus) VALUES ?',
    [items.map(item => [item.id, item.bookname, item.author, item.cover, item.pagenum, item.publicationdate, item.genre, item.description, item.rating, item.reviewText, item.readingStatus])],
    (err, result) => {
      if (err) {
        console.error('Помилка вставки даних:', err);
        res.status(500).send('Помилка сервера');
      } else {
        res.status(200).send('Дані успішно вставлено');
      }
    }
  );
});

app.delete('/items', (req, res) => {
  db.query('DELETE FROM items', (err, result) => {
    if (err) {
      console.error('Помилка виконання запиту на видалення:', err);
      res.status(500).send('Помилка сервера');
    } else {
      res.status(200).send('Всі записи успішно видалено');
    }
  });
});

```

```

app.get('/items', (req, res) => {
  db.query('SELECT * FROM items', (err, results) => {
    if (err) {
      console.error('Помилка виконання запиту:', err);
      res.status(500).send('Помилка сервера');
    } else {
      res.status(200).json(results);
    }
  });
});

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, path.join(__dirname, '../public/covers/'));
  },
  filename: function (req, file, cb) {
    cb(null, `${Date.now()}cover.jpg`);
  }
});

const upload = multer({ storage: storage });

app.post('/upload', upload.single('cover'), (req, res) => {
  const coverURL = `http://localhost:3001/covers/${req.file.filename}`;
  res.json({ coverURL: coverURL });
});

app.delete('/delete-cover/:filename', (req, res) => {
  const filename = req.params.filename;
  const filePath = path.join(__dirname, '../public/covers/', filename);

  fs.unlink(filePath, (err) => {
    if (err) {
      res.status(500).send('Internal Server Error');
    } else {
      res.status(200).send('File deleted successfully');
    }
  });
});

app.listen(3001, () => {
  console.log("Server is running. Port: 3001")
})

```

Б.4 Текст файлу AuthForm.js

```

import React, { Component } from 'react'

export class AuthForm extends Component {
  constructor(props) {

```

```

    super(props);
    this.state = {
      login: "",
      password: ""
    };
  }

  handleInputChange = (e) => {
    this.setState({
      [e.target.name]: e.target.value
    });
  };

  handleAuthClick = () => {
    this.props.onAuthClick(this.state.login, this.state.password);
    this.setState({
      login: "",
      password: ""
    });
  }

  handleRegisterClick = () => {
    this.props.onRegisterClick(this.state.login, this.state.password);
    this.setState({
      login: "",
      password: ""
    });
  }

  render() {
    return (
      <div className='formdiv'>
        <form className='authform'>
          <div className='formgroup'>
            <label htmlFor='login'>Логин</label>
            <input
              type='text'
              id='logininput'
              name='login'
              value={this.state.login}
              onChange={this.handleInputChange}
              required
            />
          </div>
          <div className='formgroup'>
            <label htmlFor='password'>Пароль </label>
            <input
              type='password'
              id='passwordinput'
              name='password'
              value={this.state.password}
              onChange={this.handleInputChange}
              required
            />
          </div>
        </form>
      </div>
    );
  }
}

```

```

    />
  </div>
  <div className='formbuttons'>
    <button type='button' onClick={this.handleAuthClick}>Авторизація</button>
    <button type='button' onClick={this.handleRegisterClick}>Реєстрація</button>
  </div>
</form>
</div>
)
}
}
export default AuthForm

```

Б.5 Текст файлу AddItemCard.js

```

import React, { Component } from 'react'

export class Item extends Component {
  constructor(props) {
    super(props);
    this.state = {
      bookname: "",
      author: ""
    };
  }

  handleInputChange = (e) => {
    this.setState({
      [e.target.name]: e.target.value
    });
  };

  handleAddItem = () => {
    const newItem = {
      bookname: this.state.bookname,
      author: this.state.author
    };
    this.props.onAddItem(newItem);
    this.setState({
      bookname: "",
      author: ""
    });
  };

  render() {
    return (
      <div className='item'>
        <img
          src={`../covers/addNewItem.png`}
          alt='Add new book cover'
          onClick={this.handleAddItem}
        />

```

```

      <input
        type="text"
        name="bookname"
        placeholder="Назва книги"
        value={this.state.bookname}
        onChange={this.handleInputChange}
        className='addItemCardInput'
      />
      <input
        type="text"
        name="author"
        placeholder="Ім'я автора"
        value={this.state.author}
        onChange={this.handleInputChange}
        className='addItemCardInput'
      />
      <div className='addItemButton' onClick={this.handleAddItem}>Додати</div>
    </div>
  )
}
}

export default Item

```

Б.6 Текст файлу Dropdown.js

```

import React, { Component } from 'react';

class Dropdown extends Component {
  render() {
    const { value, onChange } = this.props;

    const handleChange = (e) => {
      onChange(e.target.value);
    };

    return (
      <select value={value} onChange={handleChange} className="dropdown">
        <option value="Читатиму">Читатиму</option>
        <option value="Читаю">Читаю</option>
        <option value="Прочитав">Прочитав</option>
      </select>
    );
  }
}

export default Dropdown;

```

Б.7 Текст файлу Header.js

```
import React, {useState} from 'react'

export default function Header({ onChange, selectedItem, onGoBackToLists }) {
  const [active, setActive] = useState("Читатиму");

  const handleStatusChange = (status) => {
    setActive(status);
    onChange(status);
  };

  const handleGoBackClick = () => {
    onGoBackToLists();
  };

  if (selectedItem === null){
    return (
      <header>
        <div>
          <ul className='menu'>
            <li className={active === "Читатиму" ? "activeItem" : ""} onClick={() =>
handleStatusChange("Читатиму")}>Читатиму</li>
            <li className={active === "Читаю" ? "activeItem" : ""} onClick={() => handleStatusChange("Читаю")}>Читаю</li>
            <li className={active === "Прочитав" ? "activeItem" : ""} onClick={() =>
handleStatusChange("Прочитав")}>Прочитав</li>
          </ul>
        </div>
      </header>
    )
  } else {
    return (
      <header>
        <div>
          <ul className='menu'>
            <li onClick={handleGoBackClick} >До списків</li>
          </ul>
        </div>
      </header>
    )
  }
}
```

Б.8 Текст файлу Item.js

```
import React, { Component } from 'react'
import Dropdown from './Dropdown'

export class Item extends Component {
  constructor(props) {
```

```

    super(props);
    this.state = {
      readingStatus: this.props.item.readingStatus || "",
    };
  }

  handleReadingStatusChange = (status) => {
    this.setState({ readingStatus: status });
    this.props.onReadingStatusChange(this.props.item.id, status);
  };

  handleItemClick = () => {
    this.props.onItemClick(this.props.item);
  };

  handleDeleteClick = () => {
    const { onDeleteItem } = this.props;
    onDeleteItem(this.props.item.id);
  };

  render() {
    return (
      <div className='item'>
        <div className='xmark' onClick={this.handleDeleteClick}>x</div>
        <img
          src={this.props.item.cover ? this.props.item.cover : "/covers/noImage.png"}
          alt={'Book cover with ID ' + this.props.item.id}
          onClick={this.handleClick}
        />
        <h2>{this.props.item.bookname}</h2>
        <h3>{this.props.item.author}</h3>
        <Dropdown value={this.state.readingStatus} onChange={this.handleReadingStatusChange} />
      </div>
    )
  }
}

export default Item

```

Б.9 Текст файла Items.js

```

import React, { Component } from 'react'
import Item from './Item'
import AddItemCard from './AddItemCard'

export class Items extends Component {
  state = {
    filterOption: 'all',
    reviewFilterOption: 'all',
    genreFilterOption: 'all',
    minPageFilter: "",
    maxPageFilter: ""
  }

```

```

};

handleFilterChange = (filter) => {
  this.setState({ filterOption: filter });
};

handleReviewFilterChange = (filter) => {
  this.setState({ reviewFilterOption: filter });
};

handleGenreFilterChange = (filter) => {
  this.setState({ genreFilterOption: filter });
};

handleMinPageFilterChange = (event) => {
  this.setState({ minPageFilter: event.target.value });
};

handleMaxPageFilterChange = (event) => {
  this.setState({ maxPageFilter: event.target.value });
};

handleReadingStatusChange = (itemId, newStatus) => {
  const updatedItems = this.props.items.map(item => {
    if (item.id === itemId) {
      return { ...item, readingStatus: newStatus };
    }
    return item;
  });
  this.props.onItemsChange(updatedItems);
};

handleItemClick = (clickedItem) => {
  this.props.onItemClick(clickedItem);
};

handleDeleteItem = (clickedItem) => {
  this.props.onDeleteItem(clickedItem);
};

render() {
  const { items, activeStatus, searchText } = this.props;
  const { filterOption, reviewFilterOption, genreFilterOption, minPageFilter, maxPageFilter } = this.state;

  let filteredItemstemp = items.filter(item =>
    (item.readingStatus === activeStatus)

  );

  filteredItemstemp = filteredItemstemp.filter(item =>
    (item.bookname.toLowerCase().includes(searchText.toLowerCase()) ||
    item.author.toLowerCase().includes(searchText.toLowerCase()))
  );

```

```

if (filterOption === 'rated') {
  filteredItemstemp = filteredItemstemp.filter(item => item.rating && item.rating >= 1 && item.rating <= 5);
} else if (filterOption === 'unrated') {
  filteredItemstemp = filteredItemstemp.filter(item => !item.rating || item.rating === 0);
}

if (reviewFilterOption === 'reviewed') {
  filteredItemstemp = filteredItemstemp.filter(item => item.reviewText !== "" && item.reviewText !== null);
} else if (reviewFilterOption === 'unreviewed') {
  filteredItemstemp = filteredItemstemp.filter(item => item.reviewText === "" || item.reviewText === null);
}

if (genreFilterOption !== 'all') {
  filteredItemstemp = filteredItemstemp.filter(item => item.genre === genreFilterOption);
}

if (minPageFilter !== "") {
  const minPages = parseInt(minPageFilter);
  filteredItemstemp = filteredItemstemp.filter(item => item.pagenum >= minPages);
}

if (maxPageFilter !== "") {
  const maxPages = parseInt(maxPageFilter);
  filteredItemstemp = filteredItemstemp.filter(item => item.pagenum <= maxPages);
}

const filteredItems = filteredItemstemp;

return (
  <div className='maindiv'>

    <div className='sidediv'>
      <h1>Фільтри</h1>
      <hr></hr>
      <h2>Жанри</h2>
      <div>
        <label>
          <input
            type="radio"
            name="genreFilter"
            checked={genreFilterOption === 'all'}
            onChange={() => this.handleGenreFilterChange('all')}
          />
          Всі
        </label>
        <br />
        <label>
          <input
            type="radio"
            name="genreFilter"
            checked={genreFilterOption === 'науковий'}
            onChange={() => this.handleGenreFilterChange('науковий')}
          />
        </label>
      </div>
    </div>
  </div>
)

```

```

    />
    Науковий
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="genreFilter"
      checked={genreFilterOption === 'фікшн'}
      onChange={() => this.handleGenreFilterChange('фікшн')}
    />
    Фікшн
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="genreFilter"
      checked={genreFilterOption === 'нонфікшн'}
      onChange={() => this.handleGenreFilterChange('нонфікшн')}
    />
    Нонфікшн
  </label>
</div>
<hr></hr>
<h2>Оцінка</h2>
<div>
  <label>
    <input
      type="radio"
      name="filter"
      checked={filterOption === 'all'}
      onChange={() => this.handleFilterChange('all')}
    />
    Всі
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="filter"
      checked={filterOption === 'rated'}
      onChange={() => this.handleFilterChange('rated')}
    />
    Оцінені
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="filter"
      checked={filterOption === 'unrated'}
      onChange={() => this.handleFilterChange('unrated')}
    />
    Не оцінені
  </label>

```

```

    />
    Не оцінені
  </label>
</div>
<hr></hr>
<h2>Рецензія</h2>
<div>
  <label>
    <input
      type="radio"
      name="reviewFilter"
      checked={reviewFilterOption === 'all'}
      onChange={() => this.handleReviewFilterChange('all')}
    />
    Всі
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="reviewFilter"
      checked={reviewFilterOption === 'reviewed'}
      onChange={() => this.handleReviewFilterChange('reviewed')}
    />
    З рецензією
  </label>
  <br />
  <label>
    <input
      type="radio"
      name="reviewFilter"
      checked={reviewFilterOption === 'unreviewed'}
      onChange={() => this.handleReviewFilterChange('unreviewed')}
    />
    Без рецензії
  </label>
</div>
<hr></hr>
<h2>Кількість сторінок</h2>
<div className='pageNumForm'>
  <input
    type="number"
    value={minPageFilter}
    onChange={this.handleMinPageFilterChange}
    placeholder='Від'
  />
  <input
    type="number"
    value={maxPageFilter}
    onChange={this.handleMaxPageFilterChange}
    placeholder='До'
  />
</div>

```

```

    </div>

    <main>
      <AddItemCard onAddItem={this.props.onAddItem} />
      {filteredItems.map(el => (
        <Item
          key={el.id}
          item={el}
          onReadingStatusChange={this.handleReadingStatusChange}
          onItemClick={this.handleItemClick}
          onDeleteItem={this.handleDeleteItem}
        />
      ))}
    </main>
    <div className='sidediv'></div>
  </div>
)
}
}

export default Items

```

Б.10 Текст файла ItemPage.js

```

import React, { Component } from 'react'
import Dropdown from './Dropdown'
import StarRating from './StarRating';
import axios from 'axios';

export class ItemPage extends Component {
  constructor(props) {
    super(props);
    this.state = {
      isEditable: false,
      isPageEditable: false,
      bookname: "",
      author: "",
      pagenum: "",
      publicationdate: "",
      genre: "",
      description: "",
      reviewText: ""
    };
  }

  componentDidMount() {
    const { selectedItem } = this.props;
    if (selectedItem) {
      this.setState({
        bookname: selectedItem.bookname,
        author: selectedItem.author,
        pagenum: selectedItem.pagenum,

```

```

        publicationdate: selectedItem.publicationdate,
        genre: selectedItem.genre,
        description: selectedItem.description,
        reviewText: selectedItem.reviewText
    });
}
}

handleRatingChange = (newRating) => {
    const { selectedItem, onRatingChange } = this.props;
    selectedItem.rating = newRating;
    onRatingChange(selectedItem);
};

handleEditClick = () => {
    this.setState({ isEditable: true });
};

handleSaveClick = () => {
    const { selectedItem, onSaveReviewText } = this.props;
    const { reviewText } = this.state;
    selectedItem.reviewText = reviewText;
    onSaveReviewText(selectedItem);
    this.setState({ isEditable: false });
};

handleEditPageClick = () => {
    this.setState({ isPageEditable: true });
};

handleSavePageClick = () => {
    const { selectedItem, onSavePage } = this.props;
    const { bookname, author, pagenum, publicationdate, genre, description } = this.state;

    selectedItem.description = description;
    selectedItem.bookname = bookname;
    selectedItem.author = author;
    selectedItem.pagenum = pagenum;
    selectedItem.publicationdate = publicationdate;
    selectedItem.genre = genre;

    onSavePage(selectedItem);
    this.setState({ isPageEditable: false });
};

handleReviewChange = (event) => {
    this.setState({ reviewText: event.target.value });
};

handleDeleteClick = () => {
    const { selectedItem, onDeleteItem } = this.props;
    onDeleteItem(selectedItem.id);
    this.props.onGoBackToLists();
};

```

```

};

handleDeleteCoverClick = () => {
  const { onDeleteCover } = this.props;
  onDeleteCover();
};

handleAddCoverClick = () => {
  const fileInput = document.createElement('input');
  fileInput.type = 'file';
  fileInput.accept = '.jpg';
  fileInput.onChange = this.handleCoverChange;
  fileInput.click();
};

handleCoverChange = (event) => {
  const { onDeleteCover } = this.props;
  onDeleteCover();
  const file = event.target.files[0];
  const formData = new FormData();
  formData.append('cover', file);

  axios.post('http://localhost:3001/upload', formData, {
    headers: {
      'Content-Type': 'multipart/form-data'
    }
  })
  .then(response => {
    console.log('File uploaded successfully:', response.data);
    this.props.onAddCover(response.data.coverURL);
  })
  .catch(error => {
    console.error('There was a problem with your Axios request:', error);
  });
};

handleReadingStatusChange = (status) => {
  const { selectedItem, onReadingStatusChange } = this.props;
  selectedItem.readingStatus = status;
  onReadingStatusChange(selectedItem);
};

handleDescriptionChange = (event) => {
  this.setState({ description: event.target.value });
};

handleBooknameChange = (event) => {
  this.setState({ bookname: event.target.value });
};

handleAuthorChange = (event) => {
  this.setState({ author: event.target.value });
};

```

```

handlePagenumChange = (event) => {
  this.setState({ pagenum: event.target.value });
};

handlePublicationdateChange = (event) => {
  this.setState({ publicationdate: event.target.value });
};

handleGenreChange = (event) => {
  this.setState({ genre: event.target.value });
};

render() {
  const { selectedItem } = this.props;
  const { isEditable, isPageEditable, bookname, author, pagenum, publicationdate, description, reviewText, genre } = this.state;
  const genres = ['науковий', 'фікшн', 'нонфікшн', ''];
  return (
    <div className='itemPageMainDiv'>

      <div className='sidediv leftdiv'>
        <div className='imageForm'>
          <span>ID: {selectedItem.id}</span>
          <img
            src={selectedItem.cover ? selectedItem.cover : './covers/noImage.png'}
            alt={'Book cover with ID ' + selectedItem.id}
          />
          <span>Відношення 3 : 5</span>
          <br></br>
          <span>Формат .jpg</span>
          <div className='itemPageButton' onClick={this.handleAddCoverClick}>Додати</div>
          <div className='itemPageButton deleteButton' onClick={this.handleDeleteCoverClick}>Видалити</div>
        </div>
      </div>
      <main>
        <ul className='itemPageUI'>
          <div className='partUI'>
            <li><input
              className='i1'
              type='text'
              value={bookname}
              readOnly={!isPageEditable}
              style={{ borderWidth: isPageEditable ? '1px' : '0px' }}
              onChange={this.handleBooknameChange}
            /></li>
            <li><input
              className='i2'
              type='text'
              value={author}
              readOnly={!isPageEditable}
              style={{ borderWidth: isPageEditable ? '1px' : '0px' }}
              onChange={this.handleAuthorChange}
            /></li>

```

```

</li>Сторінка: <input
  type="number"
  value={pagenum}
  readOnly={!isPageEditable}
  style={{ borderWidth: isPageEditable ? '1px' : '0px' }}
  onChange={this.handlePagenumChange}
/></li>
<li>Рік: <input
  type="number"
  value={publicationdate}
  readOnly={!isPageEditable}
  style={{ borderWidth: isPageEditable ? '1px' : '0px' }}
  onChange={this.handlePublicationdateChange}
/></li>
<li>Жанр: {isPageEditable ? (
  <select
    value={genre}
    onChange={this.handleGenreChange}
    className='genreSelector'
  >
    {genres.map((genreOption, index) => (
      <option key={index} value={genreOption}>{genreOption}</option>
    ))}
  </select>
) : (
  <span>{genre}</span>
)}
</li>

</div>
<hr></hr>
<div className='partUI'>
  <li><h2>Опис</h2></li>
  <li><textarea
    readOnly={!isPageEditable}
    style={{ borderWidth: isPageEditable ? '1px' : '0px' }}
    value={description}
    onChange={this.handleDescriptionChange}
    rows={7}
  />
</li>
</div>
<hr></hr>
<div className='partUI'>
  <li><h2>Рецензія</h2></li>
  <li>
    {selectedItem.rating}/5
    <StarRating
      rating={selectedItem.rating}
      onRatingChange={this.handleRatingChange}
    />
  </li>
</div>

```

```

        <textarea
          readOnly={!isEditable}
          style={{ borderWidth: isEditable ? '1px' : '0px' }}
          value={reviewText}
          onChange={this.handleReviewChange}
          rows={10}
        />
      </li>
    </div>
    <li className='reviewForm'>
      {isEditable ? (
        <div className='itemPageButton' onClick={this.handleSaveClick}>Зберегти</div>
      ) : (
        <div className='itemPageButton' onClick={this.handleEditClick}>Редагувати</div>
      )}
    </li>
  </ul>
</main>
<div className='sidediv rightdiv'>
  <Dropdown value={selectedItem.readingStatus} onChange={this.handleReadingStatusChange} />
  {isPageEditable ? (
    <div className='itemPageButton' onClick={this.handleSavePageClick}>Зберегти</div>
  ) : (
    <div className='itemPageButton' onClick={this.handleEditPageClick}>Редагувати</div>
  )}
  <div className='itemPageButton deleteButton' onClick={this.handleDeleteClick}>Видалити</div>
</div>
</div>
)
}
}

export default ItemPage

```

Б.12 Текст файлу SearchBar.js

```

import React, { Component } from 'react'

export class SearchBar extends Component {
  handleInputChange = (event) => {
    const searchText = event.target.value;
    this.props.onSearchTextChange(searchText);
  };
  render() {
    return (
      <div className='searchdiv'>
        <input
          type="text"
          placeholder="Пошук"
          onChange={this.handleInputChange}
        />
      </div>
    )
  }
}

```

```

    </div>
  )
}
}

export default SearchBar

```

Б.11 Текст файла StarRating.js

```

import React from 'react';

const StarRating = ({ rating, onRatingChange }) => {
  const handleStarClick = (clickedRating) => {
    onRatingChange(clickedRating);
  };

  const renderStars = () => {
    const stars = [];
    for (let i = 1; i <= 5; i++) {
      stars.push(
        <span
          key={i}
          style={{ color: i <= rating ? 'gold' : 'grey', cursor: 'pointer' }}
          onClick={() => handleStarClick(i)}
        >
          ★
        </span>
      );
    }
    return stars;
  };

  return (
    <div className='starRating'>
      {renderStars()}
    </div>
  );
};

export default StarRating;

```

Б.13 Текст файла index.css

```

@import
url('https://fonts.googleapis.com/css2?family=Comfortaa:wght@300..700&family=Kurale&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap');
body {
  font-family: Kurale;
  background-color: #FFF2D7;
  color: #862B0D;

```

```

    cursor: default;
}
* {
    margin: 0;
    padding: 0;
}
.wrapper {
    width: 95vw;
    margin: 2.56rem auto;
}
header{
    text-align: center;
    margin-bottom: 5rem;
}
.menu li {
    display: inline-flex;
    margin-inline: 5vw;
    font-size: 2.5rem;
    max-width: 100%;
    transition: all 1s ease;
}
.menu li:hover{
    font-size: 2.525rem;
    cursor: pointer;
    opacity: 70%;
}
.activeItem{
    font-weight: 900;
    color: #db4615;
}
.maindiv{
    display: flex;
}
main {
    display: flex;
    width: 50%;
    flex-wrap: wrap;
    justify-content: space-between;
    margin: 0 auto;
}
.item {
    width: 15rem;
    min-height: 31rem;
    max-height: 31rem;
    margin-bottom: 50px;
    border: solid #862B0D 1px;
    border-radius: 15px;
    padding: 0.5rem;
    background-color: bisque;
    padding-bottom: 0.5rem;
    transition: all 1s ease;
}
.item:hover{

```

```

    scale: 1.005;
    border-color: rgb(202, 0, 0);
}
.item img {
    margin-top: -1.5rem;
    height: 23rem;
    width: 100%;
    border-radius: 15px 15px 0 0;
}
.item img:hover, .item .dropdown:hover, .item .addItemButton:hover {
    cursor: pointer;
}
.item *, .item {
    text-align: center;
}
.dropdown {
    font-family: inherit;
    font-size: 1.25rem;
    cursor: inherit;
    line-height: inherit;
    background-color: inherit;
    border-color: #862B0D;
    border-radius: 5px;
    width: 100%;
    color: inherit;
    margin: 0.5rem 0;
    transition: all 1s ease;
}
.dropdown:focus, .dropdown::selection, select:focus, select::selection {
    outline: none;
}
.addItemButton {
    font-size: 1.25rem;
    border-style: solid;
    border-width: 1px;
    border-color: #862B0D;
    border-radius: 5px;
    margin: 0.5rem auto;
    padding: 0.1rem;
    width: 90%;
    transition: all 1s ease;
}
.addItemButton:hover, .dropdown:hover {
    background-color: rgba(255, 128, 0, 0.14);
    border-color: rgb(202, 0, 0);
    color: rgb(202, 0, 0);
}
.itemPageMainDiv {
    width: 100%;
    display: flex;
}
.itemPageMainDiv main {
    padding-left: 5rem;

```

```

    display: inline-flex;
    max-width: 100%;
}
.itemPageMainDiv .sidediv {
    width: 30%;
}
.itemPageMainDiv .sidediv img {
    width: 15vw;
    border-color: #862B0D;
    border-width: 1px;
    border-style: solid;
}
.rightdiv {
    text-align: left;
}
.imageForm {
    float: right;
    text-align: center;
    width: 15vw;
}
.itemPageMainDiv .sidediv .dropdown {
    width: 10rem;
}
.itemPageUI {
    list-style-type: none;
    width: 100%;
    margin: 0;
    padding: 0 5rem 0 0;
    font-size: 1.1rem;
}
.itemPageUI li {
    margin-bottom: 0.25rem;
}
.itemPageUI hr {
    border-style: solid;
    border-color: #862B0D;
    border-width: 1px;
}
.partUI, .itemPageUI hr {
    margin-bottom: 2.5rem;
}
.starRating span {
    font-size: 2.5rem;
    transition: all 1s ease;
}
.starRating span:hover {
    cursor: pointer;
    font-size: 2.52rem;
    opacity: 60%;
}
.itemPageButton {
    text-align: center;

```

```

    font-size: 1.25rem;
    border-style: solid;
    border-width: 1px;
    border-color: #862B0D;
    border-radius: 5px;
    margin: 0.5rem auto;
    padding: 0.1rem;
    width: 10rem;
    transition: all 1s ease;
}

.reviewForm .itemPageButton {

    display: inline-block;
    margin: 0;
    margin-inline: 2.5rem;
}

.reviewForm {
    text-align: center;
}

.rightdiv .itemPageButton{
    margin-left: 0;
}

.itemPageButton:hover, .dropdown:hover, select:hover{
    background-color: rgba(255, 128, 0, 0.14);
    border-color: rgb(202, 0, 0);
    color: rgb(202, 0, 0);
    cursor: pointer;
}

.deleteButton:hover {
    background-color: rgba(255, 0, 0, 0.2);
}

textarea {
    font-family: inherit;
    font-size: inherit;
    color: inherit;
    background-color: inherit;
    border-color: #862B0D;
    border-radius: 20px;
    border-width: 0px;
    padding: 1rem;
    width: 95%;
    height: auto;
}

textarea:focus{
    outline: none;
}

li input {
    font-family: inherit;
    font-size: 1rem;
    color: inherit;
    background: transparent;
    border-width: 0px;
    border-color: #862B0D;

```

```

width: auto;
border-radius: 10px;
font-weight: 500;
padding: 0.25rem;
}
input:focus {
  outline: none;
}
.i1 {
  font-size: 2rem;
  font-weight: 600;
}
.i2 {
  font-size: 1.5rem;
  font-weight: 600;
}
.addItemCardInput {
  font-family: inherit;
  font-size: 1.25rem;
  font-weight: 100;
  color: inherit;
  background: transparent;
  border: inherit;
  border-color: #862B0D;
  border-radius: 5px;
  width: 92%;
  margin: 0.05rem auto;
}
::placeholder {
  color: inherit;
}
.addItemCardInput, .addItemButton {
  margin-block: 0.2rem
}
.addItemButton {
  background-color: rgb(255, 217, 170);
  position: relative;
}
.xmark {
  color: rgb(202, 0, 0);
  margin: 0;
  padding: 0;
  background: bisque;
  font-size: 1rem;
  font-weight: 600;
  border-radius: 15px;
  width: 1.5rem;
  height: 1.5rem;
  position: relative;
  left: 13.5rem;

  text-align: center;
  cursor: pointer;

```

```

        transition: all 1s ease;
    }
    .xmark:hover{
        background: rgb(202, 0, 0);
        color:bisque
    }
    .searchdiv {
        width: 100%;
        margin: 0 auto;
        margin-bottom: 2rem;
        text-align: center;
    }
    .searchdiv input {
        border-width: 1px;
        border-radius: 30px;
        border-color: #862B0D;
        background: bisque;
        font: inherit;
        font-size: 1.5rem;
        color: inherit;
        width: 50%;
        text-align: center;
        padding: 0.75rem;
        transition: all 1s ease;
    }
    .searchdiv input:hover{
        border-color: rgb(202, 0, 0);
        color: rgb(202, 0, 0);
    }
    .genreSelector{
        font-family: inherit;
        font-size: 1rem;
        cursor: inherit;
        line-height: inherit;
        background-color: inherit;
        border-color: #862B0D;
        border-radius: 5px;
        width: auto;
        color: inherit;
        margin: 0.5rem 0;
        transition: all 1s ease;
    }
    .sidediv{
        width: 20%;
        text-align: center;
    }
    .rightdiv {
        text-align: left;
    }
    .sidediv hr {
        margin-block: 1rem;
        border:#862B0D solid 1px;
    }

```

```

.sidediv h2 {
    margin-bottom: 0.5rem;
}
.sidediv label input {
    margin-right: 0.5rem;
}
.sidediv label {
    font-size: 1.25rem;
}
.sidediv label:hover{
    cursor: pointer;
}
.pageNumForm input {
    font-family: inherit;
    font-size: 1rem;
    color: inherit;
    background: transparent;
    border-width: 1px;
    border-color: #862B0D;
    width: 30%;
    border-radius: 10px;
    font-weight: 500;
    padding: 0.25rem;
    margin: 0.25rem;
}
input[type='radio'] {
    accent-color: #862B0D;
}
.authform {
    display: flex;
    flex-direction: column;
    padding: 1.5rem;
    border: 1px solid #862B0D;
    border-radius: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
.formdiv, #logininput, #passwordinput{
    text-align: center;
    font-size: 2rem;
}
.formbuttons {
    display: flex;
    justify-content: space-between;
}
.formbuttons button{
    font-size: 1.25rem;
    color: inherit;
    margin-top: 1rem;
    padding: 0.75rem;
    border-style: solid;
    border-width: 1px;
    border-color: #862B0D;

```

```

    border-radius: 15px;
    background: inherit;
    margin: 1rem;
    transition: all 1s ease;
  }
  .formbuttons button:hover{
    cursor: pointer;
    background-color: rgba(255, 128, 0, 0.043);
    border-color: rgb(202, 0, 0);
    color: rgb(202, 0, 0);
    scale: 1.025;
  }
  #logininput, #passwordinput{
    background: inherit;
    color: inherit;
    border-style: solid;
    border-width: 1px;
    border-color: #862B0D;
    border-radius: 15px;
    font-size: 1.25rem;
    padding: 0.5rem;
  }
  .formdiv {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 80vh;
  }
  .formgroup {
    margin-bottom: 1rem;
  }
  .formgroup label {
    display: block;
    margin-bottom: 1rem;
  }
  .logoutdiv{
    text-align: right;
  }

  .logoutbutton{
    color: blue;
    transition: all 1s ease;
  }

  .logoutbutton:hover{
    color:red;
    cursor: pointer;
  }

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»

Дипломна кваліфікаційна робота бакалавра

РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ ВІДСТЕЖЕННЯ КОРИСТУВАЦЬКОГО СПИСКУ ЧИТАННЯ

DEVELOPMENT OF A WEB APPLICATION FOR USER'S READING LIST TRACKING

Розробив: ст. гр. КНТ-210

ІНДИК Андрій

Керівник: доцент, к.т.н.

ФЕДОРОНЧАК Тетяна

Рисунок В.1 – Слайд 1

Об'єкт дослідження – засоби для управління списком літератури.

Предмет дослідження – вебзастосунки для створення та відслідковування власного списку читання.

Мета роботи – створення вебзастосунка для покращення слідкування за власним списком читання, зберігання прочитаної літератури, запланованої та літератури, що читається в поточний момент, зручний пошук та сортування цієї літератури.

Етапи роботи:

- постановка завдання роботи;
- аналіз предметної області;
- розробка архітектури програми;
- розробка програми;
- тестування та експериментальне дослідження програми.

Рисунок В.2 – Слайд 2

Порівняння альтернатив

Параметр	Goodreads	The StoryGraph	LibraryThing
Інтерфейс	Класичне стримане оформлення, але є простір для модернізації	Сучасний, не навантажений	Застарілий, місцями не має оформлення
Оцінювання	П'ятизірковий варіант оцінювання, що може скоувати читачів.	П'ятизірковий, є можливість ставити половину та чверть зірки.	П'ятизіркове оцінювання з можливістю ставити половину зірки.
Фільтрування	Пошук не має фільтрів, але після введення запиту можна обрати критерій.	Пропонуються фільтри за настроєм книги, її темпом, переліком жанрів, трьома типами розмірів та датою публікації.	Пошук не має початкових фільтрів, але результати запиту можна відфільтрувати за типом речі, що шукають.
Додаткові можливості пошуку	Після введення запиту, пропонуються певні результати на нього, а також, на основі цих результатів, користувачу пропонуються полиці жанрів, де він може знайти схожі роботи.	Додаткових можливостей немає, але це компенсується багатими фільтрами і великою системою настрою книг.	Сервіс має унікальну систему тегів, які кожен користувач сам додає до кожної книги. Є можливість шукати книги за цими тегами.
Статистика	Надається статистика про середній рейтинг прочитаних книг, кількість книг та сторінок на рік.	Надається велика кількість статистики у вигляді діаграм	Надається широка статистика з діаграмами
Рецензія	Надається можливість поставити оцінку, написати ємну рецензію та призначити книгу на певні власні полиці за категоріями.	Пропонує коротку вижимку думок читача, його оцінку. Також читач проходить коротке опитування для того, щоб сформувати категорії під які підпадає книга, її настрої.	Користувачі не обмежені в обсязі рецензії, призначається оцінка та користувач обирає теги до книги. Рецензії можна сортувати та фільтрувати за мовами.
Додавання нових книг	Потрібно створювати запит на додавання книги, якщо її неможливо зайти на сервісі.	Є можливість ручного додавання книги, яку потім можуть затвердити.	Є можливість додати книгу за номером або вручну, якщо він відсутній. Потребує підтвердження

3

Рисунок В.3 – Слайд 3

Вибір мови програмування

Параметр	TypeScript	JavaScript
Типізація	Сильна типізація. TypeScript підтримує як статичну, так і динамічну типізацію.	Слабка типізація. JavaScript підтримує лише динамічну типізацію.
Компіляція	TypeScript потребує компіляції.	JavaScript не потребує компіляції.
Прив'язка даних	TypeScript використовує такі поняття, як інтерфейси та типи для визначення даних, що використовуються.	У JavaScript таких понять немає.
Крива навчання	TypeScript має жорстку криву навчання. Крім того, він потребує знань зі створення сценаріїв.	Це проста у вивченні і гнучка мова для написання вебскриптів.
Спільнота	TypeScript має меншу спільноту розробників програмного забезпечення.	JavaScript має велику спільноту розробників програмного забезпечення.
Створення прототипів	Функція прототипування доступна в TypeScript.	JS не підтримує прототипування.

4

Рисунок В.4 – Слайд 4

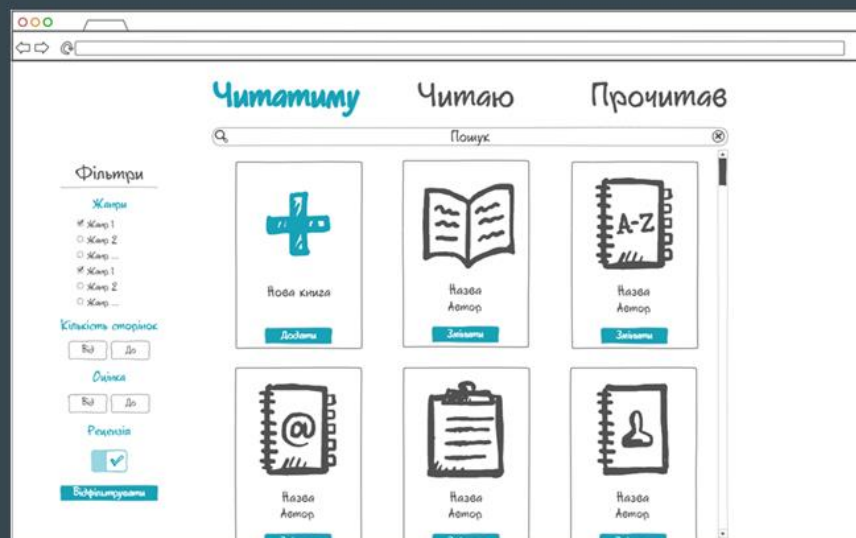
Вибір фреймворку

Параметр	Angular	React
Тип	Фреймворк	Бібліотека
Двостороннє зв'язування даних	Автоматична синхронізація даних інтерфейсу та коду	Необхідно використовувати додаткові інструменти
Компонентний підхід	Використовуються компоненти для створення користувацьких елементів	Компоненти – основа створення користувацького інтерфейсу
Складність початку роботи	Потрібен певний час для освоєння і налаштування	Простий старт і швидке створення компонентів з JSX
Розширена екосистема	Великий набір інструментів і бібліотек для різних завдань	Деякі функціональності вимагають додаткових бібліотек
Роутинг і управління станом	Вбудовані інструменти для маршрутизації та управління станом	Потрібно використовувати сторонні бібліотеки для роутингу та управління станом
Розмір і продуктивність	Більший розмір через включені інструменти, хороша продуктивність у великих додатках	Менший розмір, швидка і плавна робота, особливо на невеликих і середніх проектах
Суспільство та підтримка	Велика й активна спільнота розробників і хороша підтримка	Величезна й активна спільнота, безліч рішень і допомоги

5

Рисунок В.5 – Слайд 5

Вайрфрейм



6

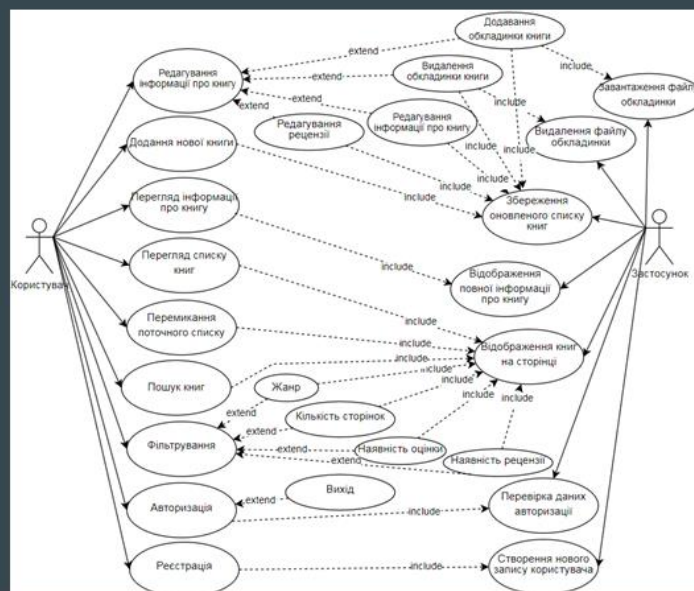
Рисунок В.6 – Слайд 6



7

Рисунок В.7 – Слайд 7

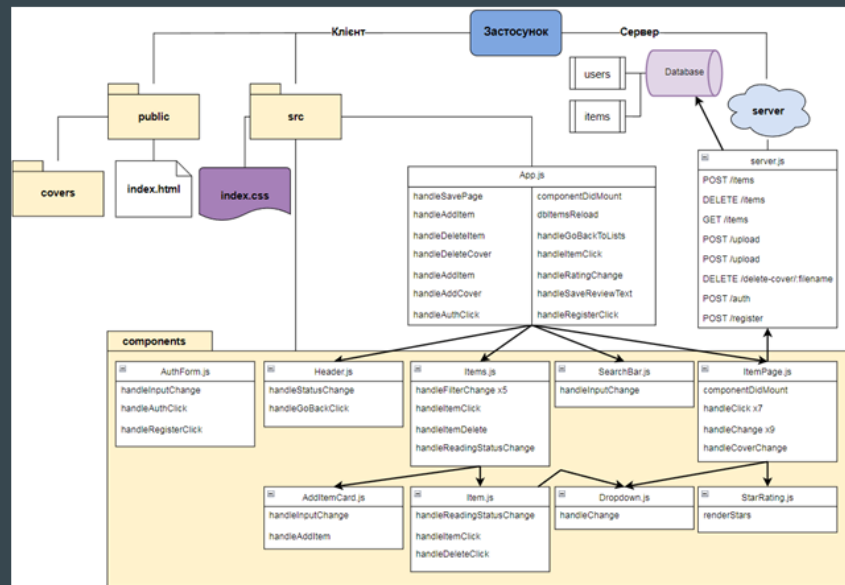
Діаграма прецедентів



8

Рисунок В.8 – Слайд 8

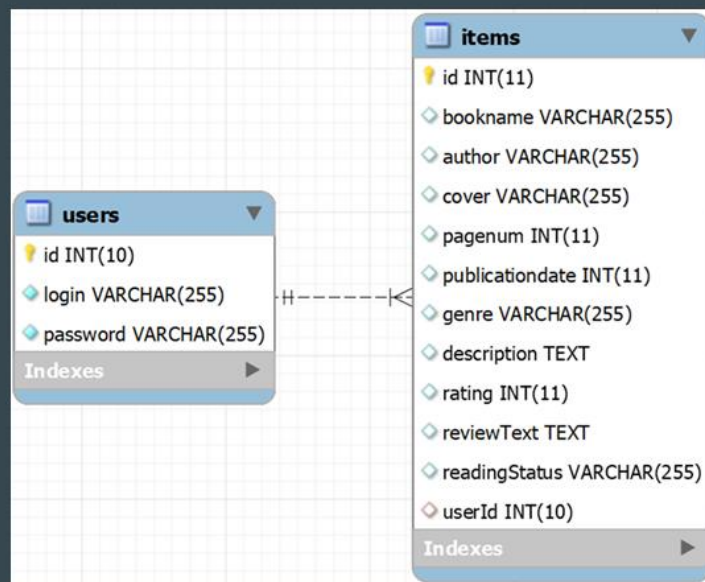
Діаграма структури проєкту



9

Рисунок В.9 – Слайд 9

Структура бази даних



10

Рисунок В.10 – Слайд 10

Сторінка авторизації та реєстрації



Логін

Пароль

Авторизація Реєстрація

11

Рисунок В.11 – Слайд 11

Головна сторінка



Читатиму Читаю Прочитав

Пошук

Фільтри

Жанри

- ☒ Всі
- ☐ Науковий
- ☐ Фікшн
- ☐ Нонфікшн

Оцінка

- ☒ Всі
- ☐ Оцінені
- ☐ Не оцінені

Рецензії

- ☒ Всі
- ☐ З рецензією
- ☐ Без рецензії

Кількість сторінок

Від До

+

МІЯ КНИГА

Перша назва
Автор Один

Читатиму

П'ята назва
Автор Три

Читатиму

МІЯ

МІЯ

12

Рисунок В.12 – Слайд 12

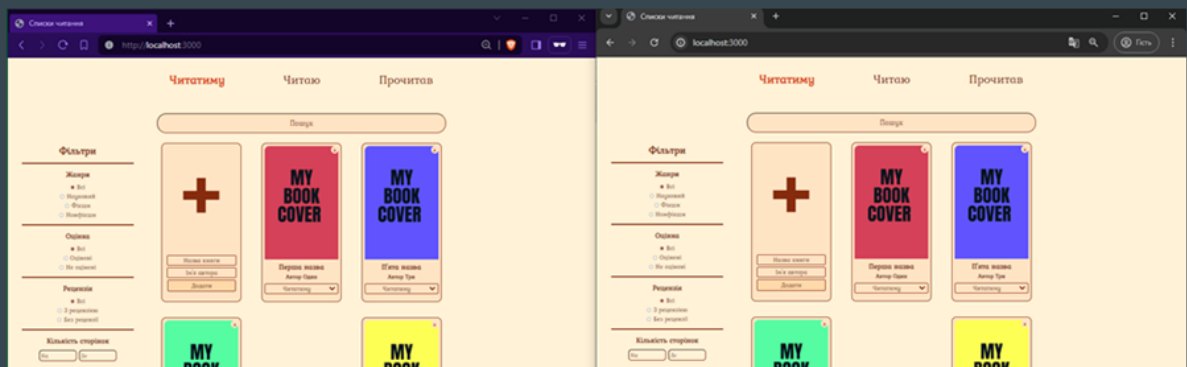
Сторінка інформації про обрану книгу



13

Рисунок В.13 – Слайд 13

Тестування кросбраузерності



14

Рисунок В.14 – Слайд 14

Чекліст тестування

Параметр, що перевіряється	Google Chrome 124.0.6367.208	Brave 1.65.133 124.0.6367.208
Перегляд списку книг	Пройдено	Пройдено
Перемикання між доступними списками	Пройдено	Пройдено
Пошук за автором або назвою	Пройдено	Пройдено
Фільтрування	Пройдено	Пройдено
Створення нової книги	Пройдено	Пройдено
Видалення картки книги	Пройдено	Пройдено
Зміна стану читання на картці книги	Пройдено	Пройдено
Перегляд сторінки детальної інформації	Пройдено	Пройдено
Редагування детальної інформації	Пройдено	Пройдено
Встановлення оцінки	Пройдено	Пройдено
Редагування рецензії	Пройдено	Пройдено
Зміна стану читання на сторінці інформації	Пройдено	Пройдено
Видалення книги на сторінці інформації	Пройдено	Пройдено
Додавання нової обкладинки до книги	Пройдено	Пройдено
Видалення обкладинки книги	Пройдено	Пройдено

15

Рисунок В.15 – Слайд 15

Висновки

Під час виконання кваліфікаційної дипломної роботи бакалавра було проведено огляд альтернатив у вигляді успішних проєктів на цю тематику, серед них Goodreads, TheStoryGraph та LibraryThing. Визначено слабкі та сильні сторони цих альтернатив, описано їх основний функціонал. Відмінністю розробленого застосунку від них можна назвати відкритий доступ до створення власних об'єктів книг, їх редагування, додавання власних обкладинок з боку користувача, що, або неможливо, або є дуже ускладненим процесом в розглянутих аналогах, який займає багато часу і несе за собою багато кроків, підтверджень та перевірок.

Функціональну частину розробки проєкту для відстеження користувацького списку читання було виконано мовою JavaScript з використанням бібліотеки React для побудови інтерфейсу та середовищем виконання Node.js для розгортання клієнтської та серверної частин проєкту. Зберігання даних реалізовано шляхом впровадження до проєкту бази даних, що підключена до сервера. В свою чергу, сервер має методи отримання і маніпулювання цими даними, які може використовувати клієнт, що до нього звертається. Для реалізації бази даних було використано СКБД MySQL.

В результаті отримано вебзастосунок, що надає користувачу можливості переглядати книги, з якими він взаємодіє, розподіляти їх за станами читання, додавати свої думки та оцінки до книг, які він вже прочитав. Окрім цього, застосунок дозволяє оновлювати дані про книги, або розширювати наявну про них інформацію в будь-який час для зручності користувача.

16

Рисунок В.16 – Слайд 16