COURSEWORK 1

FINAL REPORT: BOOK MATCH

Name: DANIEL OBIEFULE CHIDERA

Student Number: M00976304

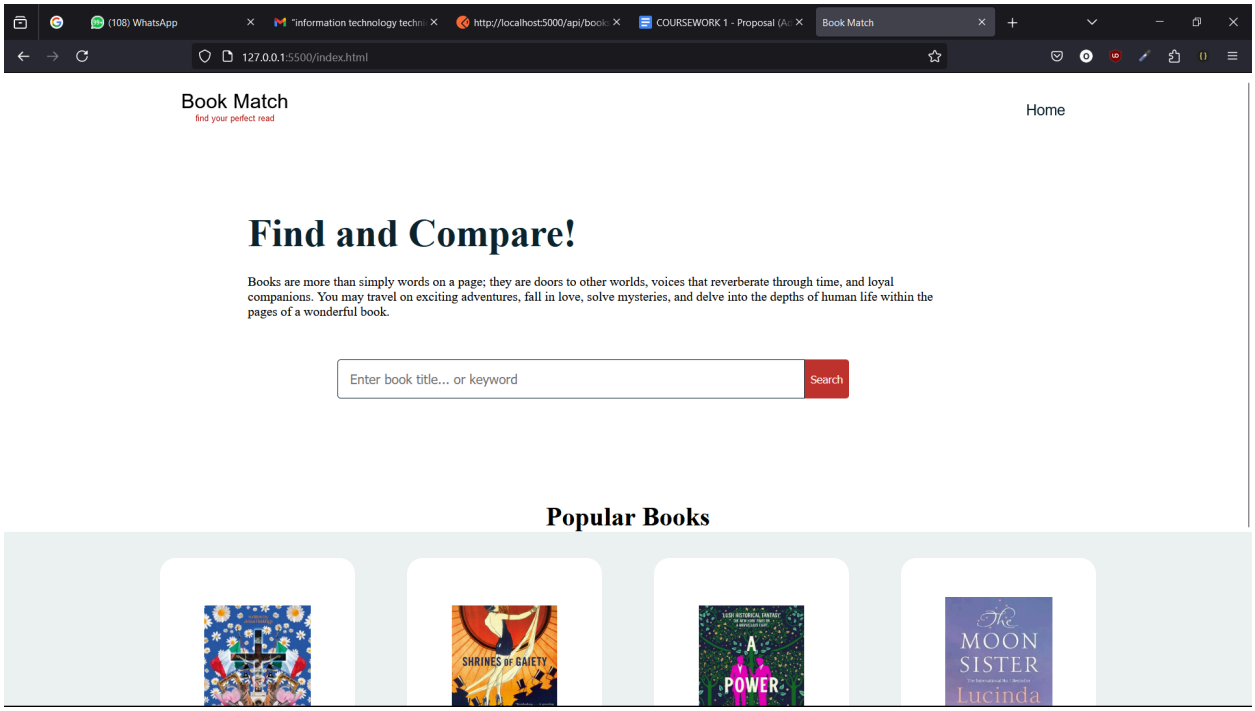
Module: 2023-24 CST3130 Advanced Web Development with Big Data

Table of Contents

Website Description and Overview

Book Match is a user-friendly website that offers a wide range of reading options for readers who value convenience and diversity. Through this platform, you can easily explore a vast array of literature from the comfort of your home. We acquire data through a systematic web scraping method. Our web scrapers extract pricing information and additional data related to books from various designated websites. Webscraping was streamlined to Romance books to be able to properly demonstrate the comparison feature.

Landing Page:



The landing page of the Book Match website has an attractive and user-friendly design. It features a clean and inviting user interface serving as a welcome page to BookMatch. At the top,

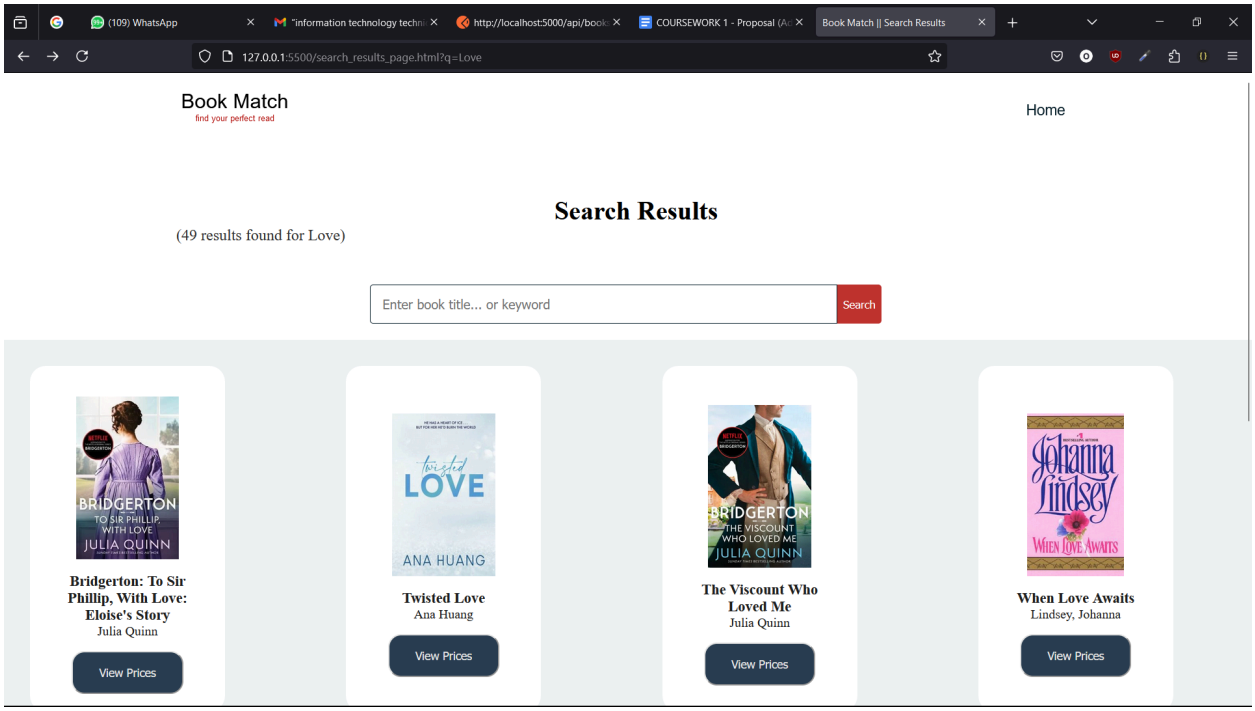
there is a sleek navigation bar showcasing the Book Match logo and a "Home" link for easy access to the main page from any section of the site.

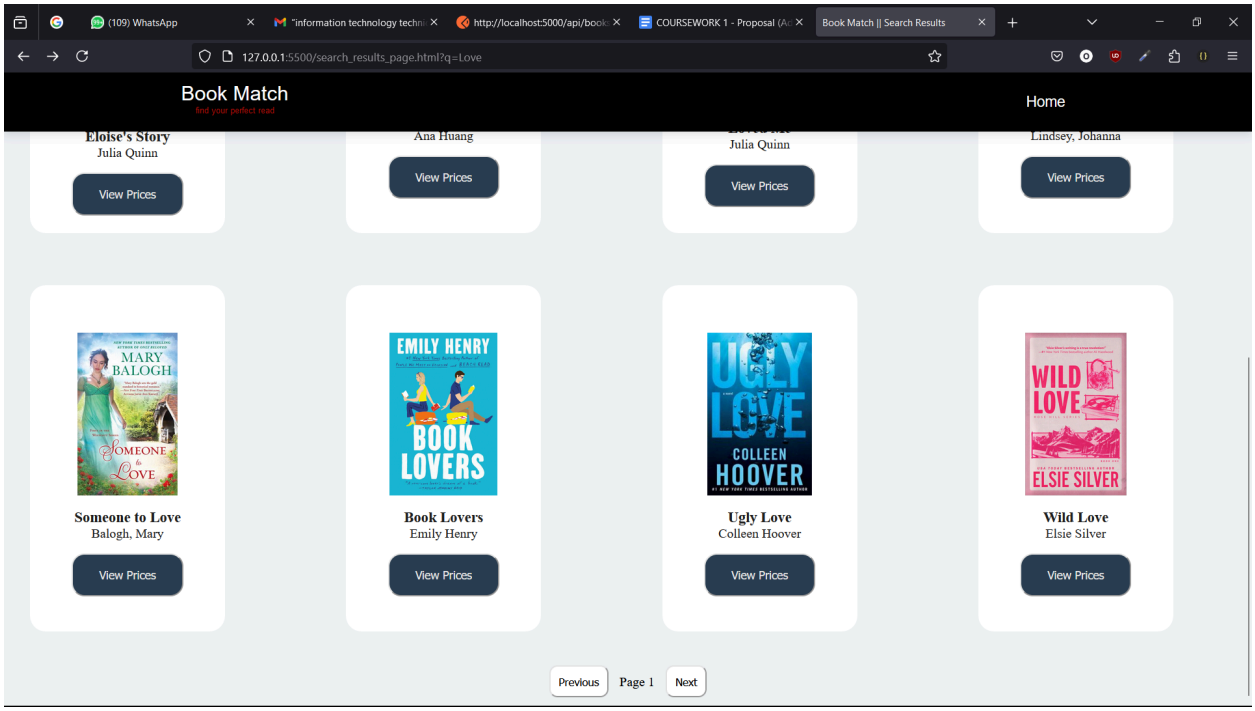
Below the navigation bar, the hero section immediately draws attention with a bold headline, "Find and Compare!", encouraging users to explore the application's core functionality. This section is complemented by descriptive text that eloquently articulates the transformative power of books, positioning them as gateways to diverse experiences and emotions.

A prominent search form below the hero section invites users to enter book titles or keywords. The "Search" button provides a call-to-action for initiating book searches.

The "Popular Books" section is a dynamic gateway to featured literary works. Displayed in a grid layout, each book card includes captivating book cover images, accompanied by concise titles and author names. This arrangement enhances visual appeal and facilitates quick access to popular book selections, enticing users to explore further.

Search Results Page:



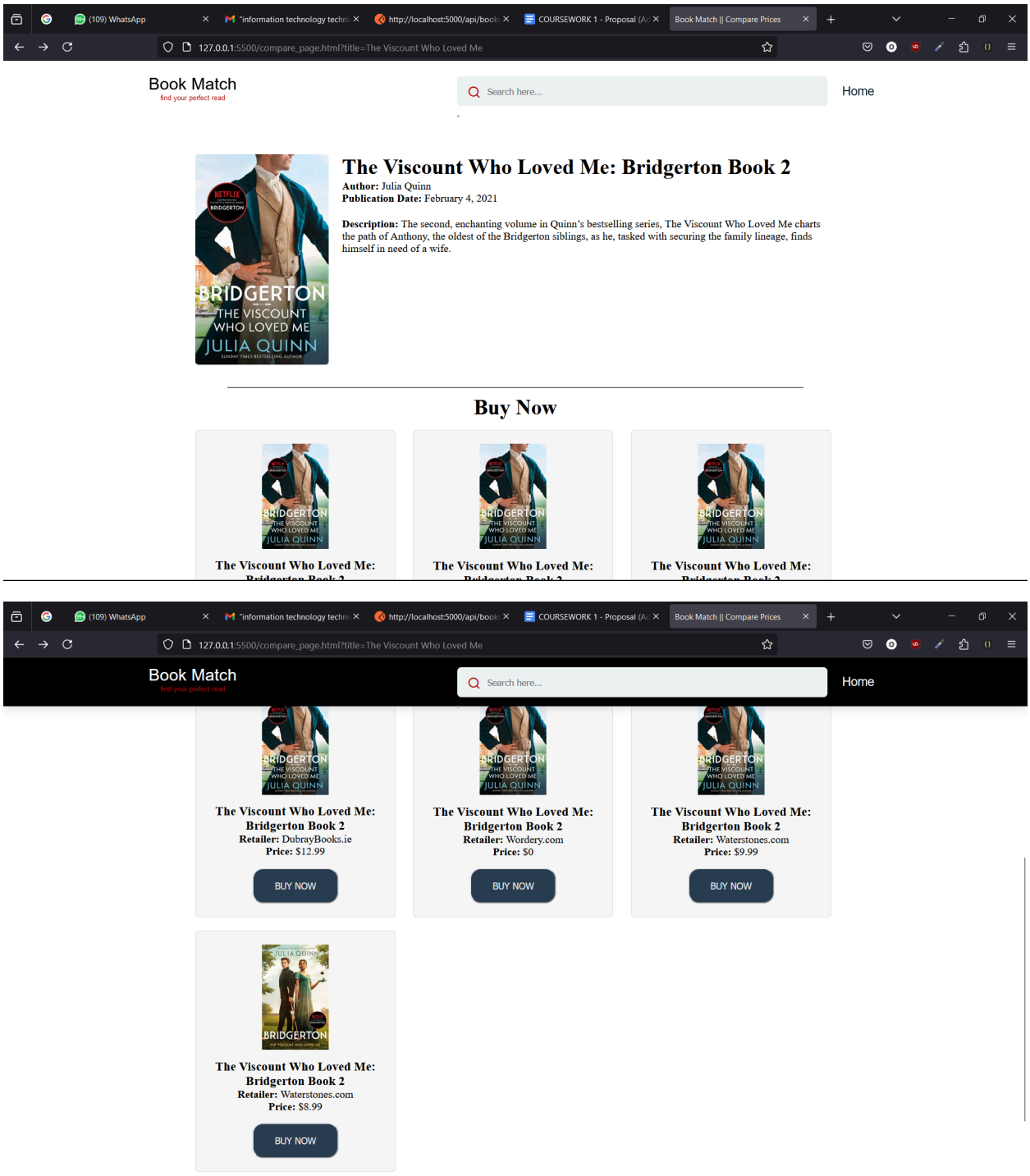


The layout of the Search Results Page begins with a navigation bar at the top, which includes the Book Match logo and a link back to the Home page for easy navigation. Directly below the navigation bar, the page prominently displays the title "Search Results," indicating the current context of the page. Next to it, the "results-count" dynamically updates to show the number of books found matching the user's search criteria, providing instant feedback to enhance usability.

A prominent search form below the "Search Results" heading allows users to enter book titles or keywords. Clicking the "Search" button initiates the book search. Below the search bar, the results with book information based on the user's search query are displayed. Each book entry includes essential details such as title, author, cover image, and a "View Prices" button, which takes the user to the compare_page when clicked.

To navigate through multiple search results, a pagination control interface is incorporated at the bottom of the page. It includes "Previous" and "Next" buttons flanking the current page of the search results "page 1" indicator, displaying the current page number.

Comparison Page:



The Comparison Page is designed to provide users with comprehensive details and options for purchasing their chosen book. The page layout begins with a consistent navigation bar featuring the Book Match logo and a link back to the Home page, maintaining continuity with the rest of the site's design and navigation structure.

Once users enter the Comparison Page, they encounter a central section that dynamically populates with essential information about the selected book, including its title, author, publication date, and a brief description. This setup ensures that users have immediate access to key details that inform their purchasing decisions.

Below the book details of the selected book, the page features a distinct "Buy Now" section. This section is a standout feature of the Comparison Page, presenting users with a grid layout that lists various online retailers offering the book for sale. Each retailer entry includes details such as the price of the book and links directly to the retailer's product page. This setup allows users to compare prices across different platforms effortlessly.

A "Buy Now" button is prominently displayed for each retailer listed. Clicking this button directs users to the specific retailer's website, where they can purchase the book. This functionality provides a seamless transition from browsing to purchasing, enhancing user convenience and facilitating informed decision-making.

Technologies Used:

The website Book Match was created using a range of technologies to guarantee effectiveness, dependability, and a smooth user experience. The platform incorporates web scraping, testing, database management, REST API, and frontend development using Java, multithreading, Selenium, Spring, Hibernate, JUnit, Express.js, MySQL, HTML, CSS, and JavaScript.

Frontend Development:

- HTML and CSS: Establish the layout and design of the website, delivering a neat and user-friendly interface for users to browse and compare books.
- JavaScript: Improves interaction on the website, enabling dynamic content updates and user participation.

Backend Development:

- REST API with Express.js: Supported by Express.js, a Node.js framework that creates a RESTful API, the backend communicates with the MySQL database, effectively managing HTTP requests and responses.

Data Acquisition:

- Java and Multithreading: Java is utilized for data acquisition, utilizing multithreading for simultaneous processing, ensuring efficient and parallel web scraping for rapid data retrieval.
- Selenium for Web Scraping: This tool is used for automated web interactions, allowing dynamic content extraction during the web scraping process, and ensuring precise and real-time data collection from various online retailers.

Integration and Database Management:

- Spring for Integration: The Spring framework enables seamless integration of web scraping processes, enhancing scalability and maintainability, and ensuring smooth coordination between different components.
- Hibernate for Database Connectivity: Hibernate manages connectivity with the MySQL database, simplifying data handling by mapping Java objects to database tables and streamlining database operations.
- MySQL Database: Acts as a centralized repository for scraped data, providing a consistent and structured source of information. The Java web scraper populates this database, maintaining an up-to-date inventory of book information.

Testing:

- JUnit for Testing Web Scraper: JUnit is used for unit testing to validate the reliability and accuracy of the Java web scraper, ensuring the stability of the codebase and the dependability of the web scraping mechanisms.

- ❖ Jest for Testing API Routes and Utilities: Used to ensure the robustness of the Express.js API and comprehensively test routes and utilities, enhancing the overall reliability of the backend.

Data Scraping

Data for the Book Match website is scraped from:

- Dubray Books - https://www.dubraybooks.ie
- Half Price Books - https://www.hpb.com
- Owl's Nest Books - https://owlsnestbooks.com
- Qbd Books- https://www.qbd.com.au
- Wordery - https://wordery.com
- Water Stones - https://www.waterstones.com

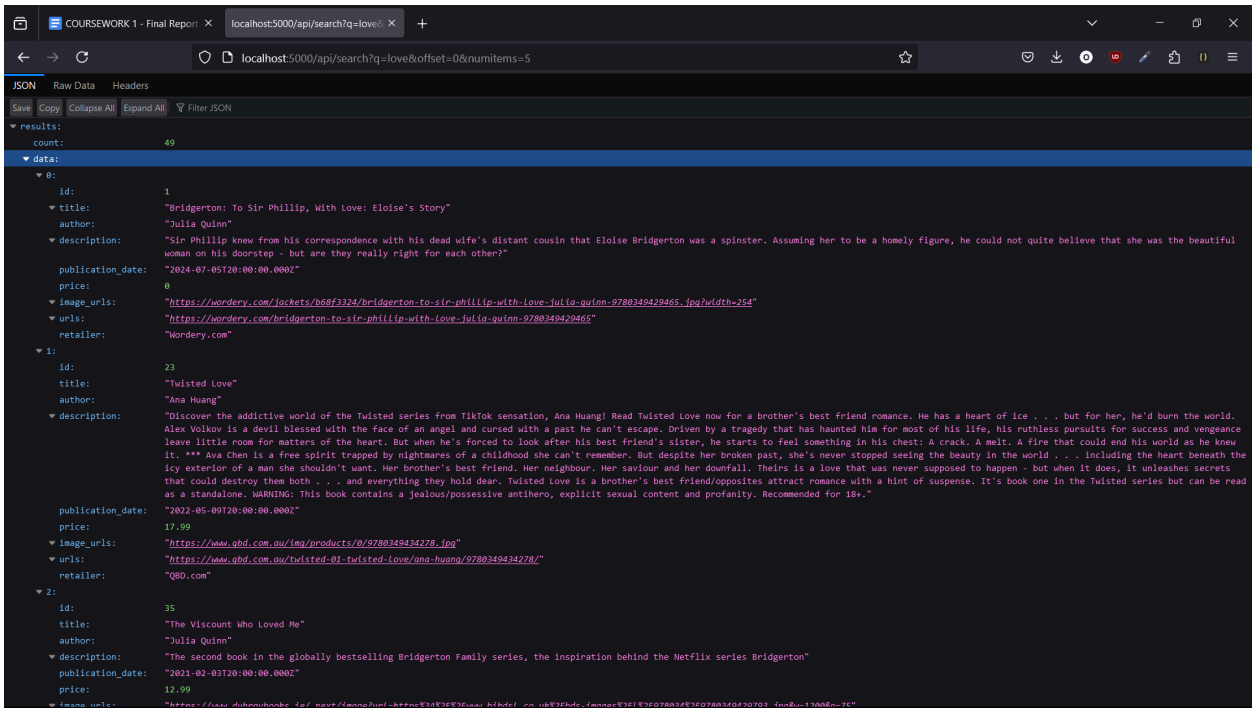
Rest API Routes Description

1. Search Route (/search)
Description: This route allows users to search for books based on a query parameter (q). It also supports pagination through offset and numitems query parameters.
Functionality:

- It first validates if the search term is provided.
- Fetches the total count of search results using the getSearchCount utility.

- Fetches the search results using the search utility.
- Combines the count and results into a single object and returns it as a JSON response.

After using this specific route with the parameters searchTerm= love, offset=0, and numitems=5, the resulting output is shown in the screenshot below.



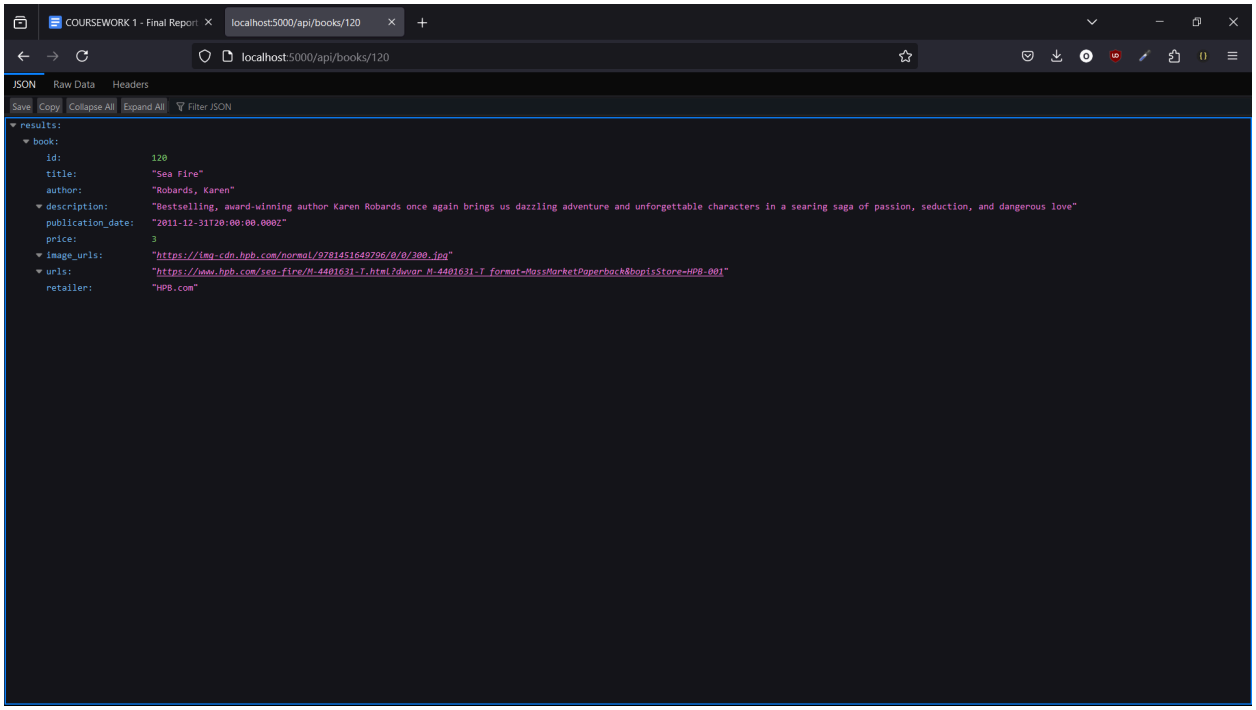
2. Book Details Route (/books/:id)

Description: This route fetches the detailed information of a specific book by its ID.

Functionality:

- Retrieves the book ID from the route parameter.
- Executes a SQL query to fetch details of the book along with its price, image URLs, and retailer information from the compare_prices table.
- If the book is not found, returns a 404 error with an appropriate message.
- Returns the book details as a JSON response.

The result below shows the book details retrieved using the route where book ID is 120



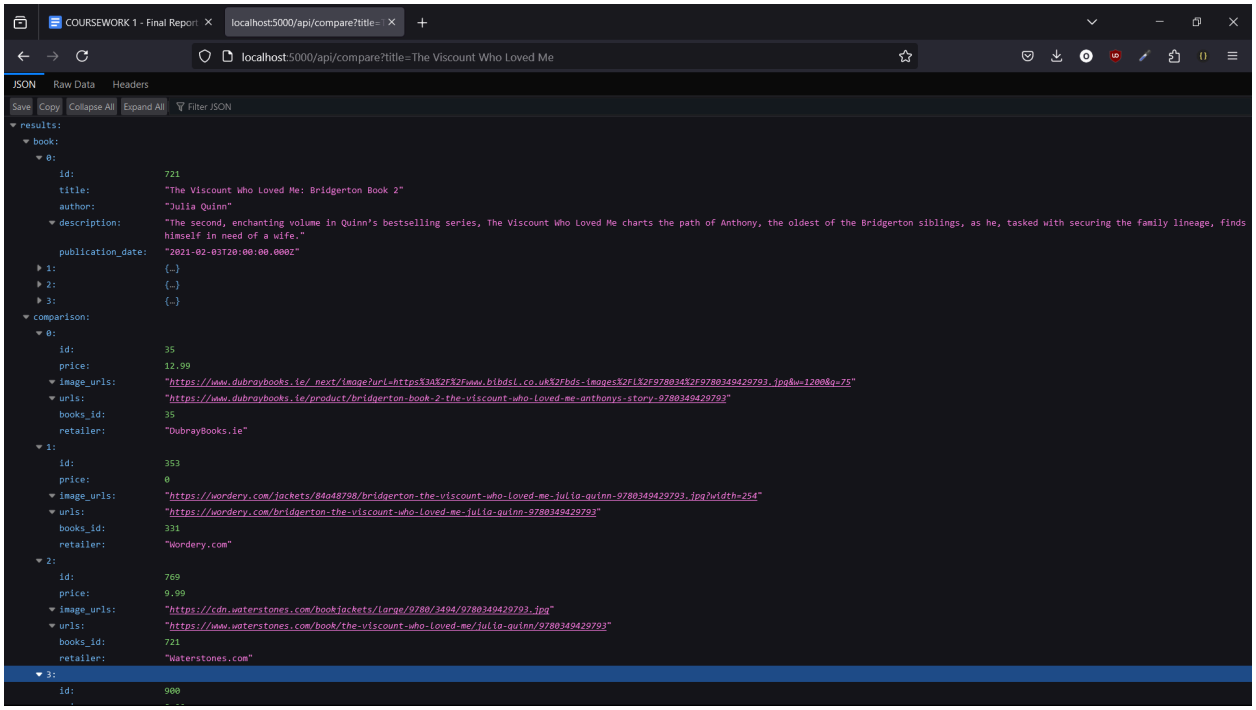
3. Compare Route (/compare)

Description: This route allows users to compare book details and prices based on the book title.

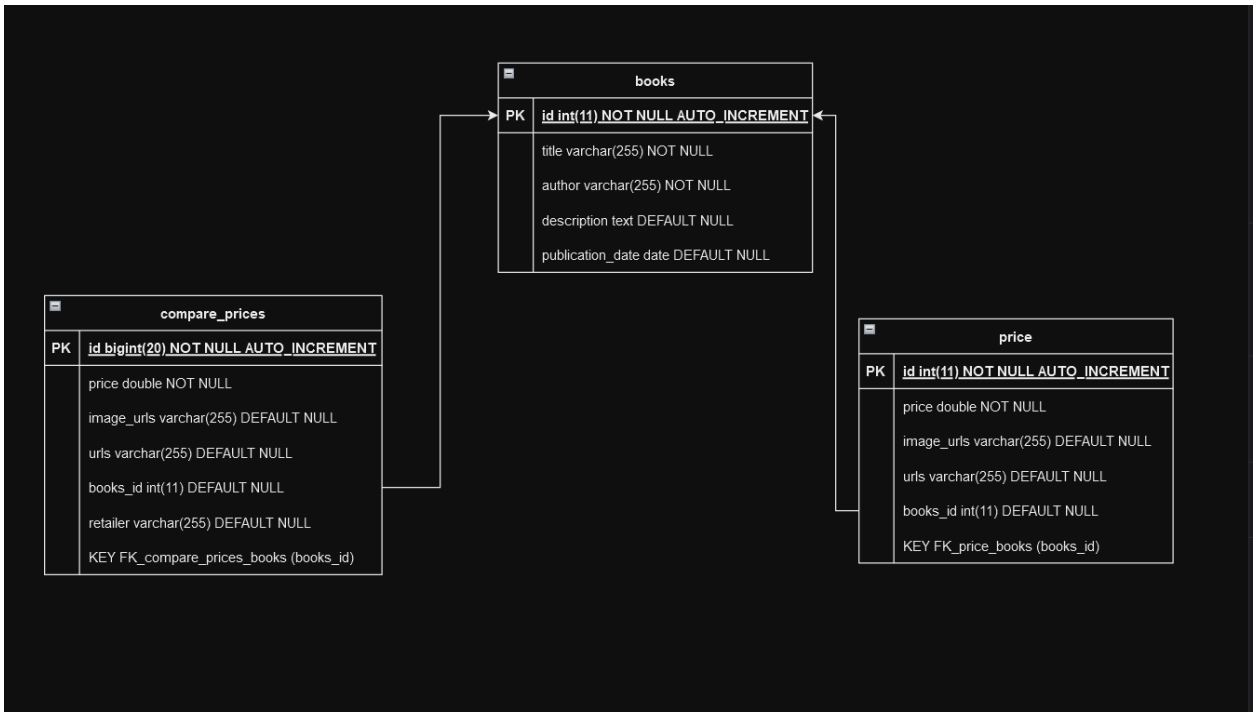
Functionality:

- Validates if the title query parameter is provided.
- Executes SQL queries to fetch book details and comparison information based on the title.
- Returns the combined results of book details and comparison information as a JSON response.

The result below shows the book details retrieved using the route where the book title matches “The Viscount Who Loved Me”


COURSEWORK 1 - Final Report
localhost:5000/api/compare?title=
localhost:5000/api/compare?title=The Viscount Who Loved Me
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
results:
book:
0:
id: 721
title: "The Viscount Who Loved Me: Bridgerton Book 2"
author: "Julia Quinn"
description: "The second, enchanting volume in Quinn's bestselling series, The Viscount Who Loved Me charts the path of Anthony, the oldest of the Bridgerton siblings, as he, tasked with securing the family lineage, finds himself in need of a wife."
publication_date: "2021-02-03T20:00:00.000Z"
1: {…}
2: {…}
3: {…}
comparison:
0:
id: 35
price: 12.99
image_urls: "https://www.dubraybooks.ie/_next/image?url=https%3A%2F%2Fwww.btbdsl.co.uk%2Fbds-images%2Fl%2F9780349429793.jpg&w=1200&q=75"
urls: "https://www.dubraybooks.ie/product/bridgerton-book-2-the-viscount-who-loved-me-anthonys-story-9780349429793"
books_id: 35
retailer: "DubrayBooks.ie"
1:
id: 353
price: 0
image_urls: "https://wordery.com/jackets/84a48798/bridgerton-the-viscount-who-loved-me-julia-quinn-9780349429793.jpg?width=254"
urls: "https://wordery.com/bridgerton-the-viscount-who-loved-me-julia-quinn-9780349429793"
books_id: 331
retailer: "Wordery.com"
2:
id: 769
price: 9.99
image_urls: "https://cdn.waterstones.com/bookjackets/large/0780/3494/9780349429793.jpg"
urls: "https://www.waterstones.com/book/the-viscount-who-loved-me/julia-quinn/9780349429793"
books_id: 721
retailer: "Waterstones.com"
3:
id: 900

Database Design



The database above consists of the structure of three tables: books, compare_prices, and price. These tables are designed to store information about books and their price comparisons across different retailers. Below is a detailed description of each table and its purpose.

1. books: The books table is used to store the core information about each book, including its title, author, description, and publication date.

Structure:

- **id** (int, Primary Key, Auto Increment): A unique identifier for each book.
- **title** (varchar): The title of the book.
- **author** (varchar): The author of the book.
- **description** (text): Description of the book.
- **publication_date** (date): The date when the book was published.

2. compare_prices: The compare_prices table stores the price information for books from different retailers, allowing users to compare prices.

Structure:

- **id** (bigint, Primary Key, Auto Increment): A unique identifier for each price entry.
- **price** (double): The price of the book at a specific retailer.
- **image_urls** (varchar): The URL of the book's image on the retailer's site.
- **urls** (varchar): The URL to the book's product page on the retailer's site.
- **books_id** (int): A foreign key linking to the id in the books table.
- **retailer** (varchar): The name of the retailer offering the book at this price.

Indexes:

- **FK_compare_prices_books**: An index on the books_id field to optimize join operations with the books table.

3. price: The price table appears to be similar to the compare_prices table, storing price information for books. This could be used for storing historical price data or prices from additional sources.

Structure:

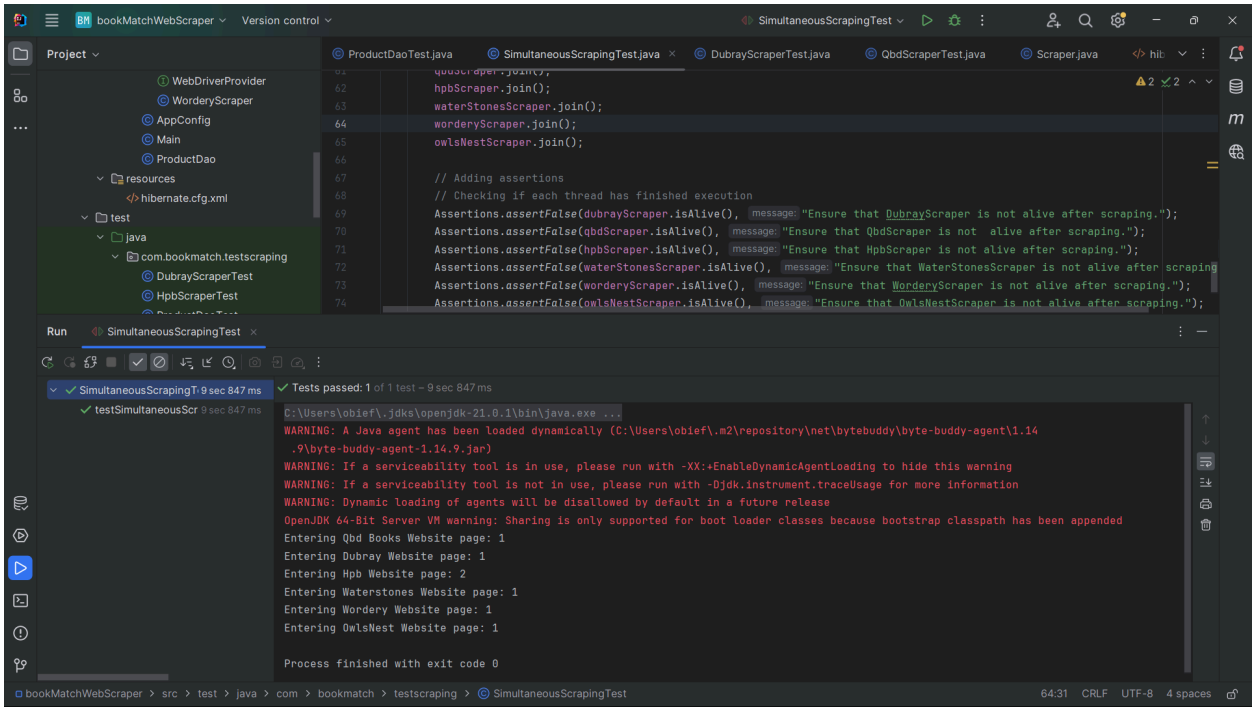
- **id** (int, Primary Key, Auto Increment): A unique identifier for each price entry.
- **price** (double): The price of the book.
- **image_urls** (varchar): The URL of the book's image.
- **urls** (varchar): The URL to the book's product page.
- **books_id** (int): A foreign key linking to the id in the books table.

Indexes:

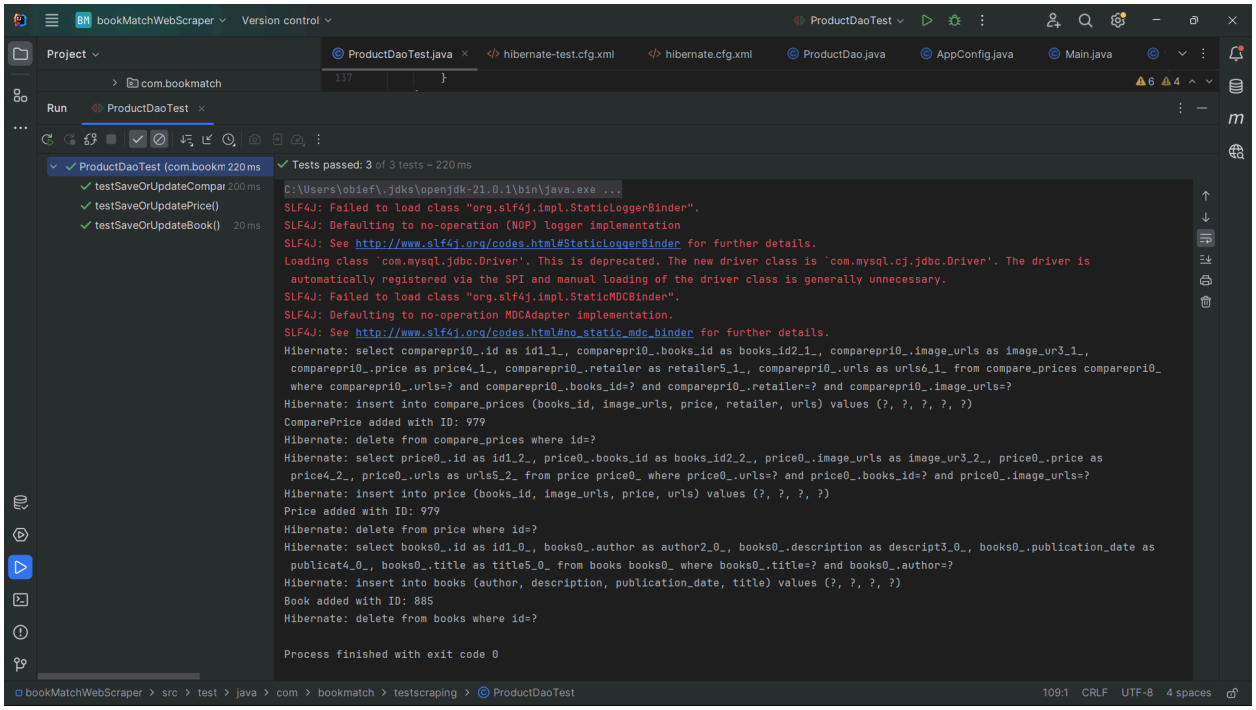
- **FK_price_books**: An index on the books_id field to optimize join operations with the books table.

Tests Results

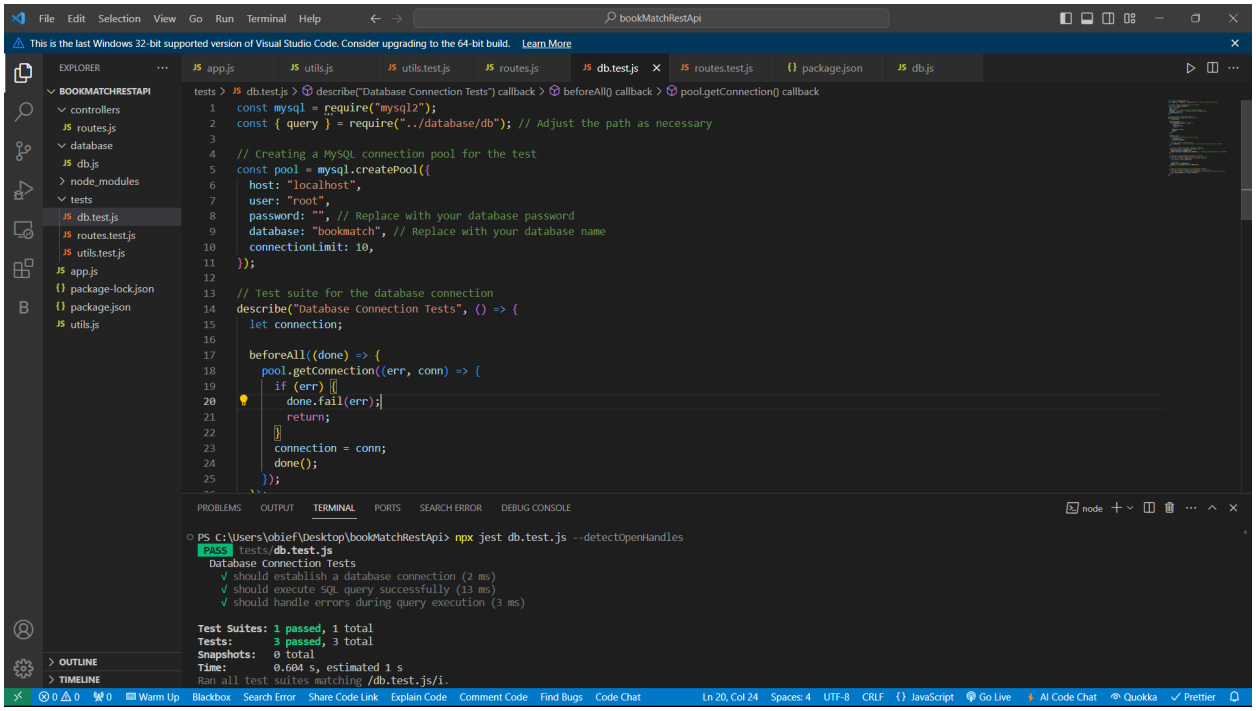
SimultanoeusScrapingTest: The screenshot below shows the results of a JUnit test carried out to confirm that the data from websites is being scraped simultaneously.



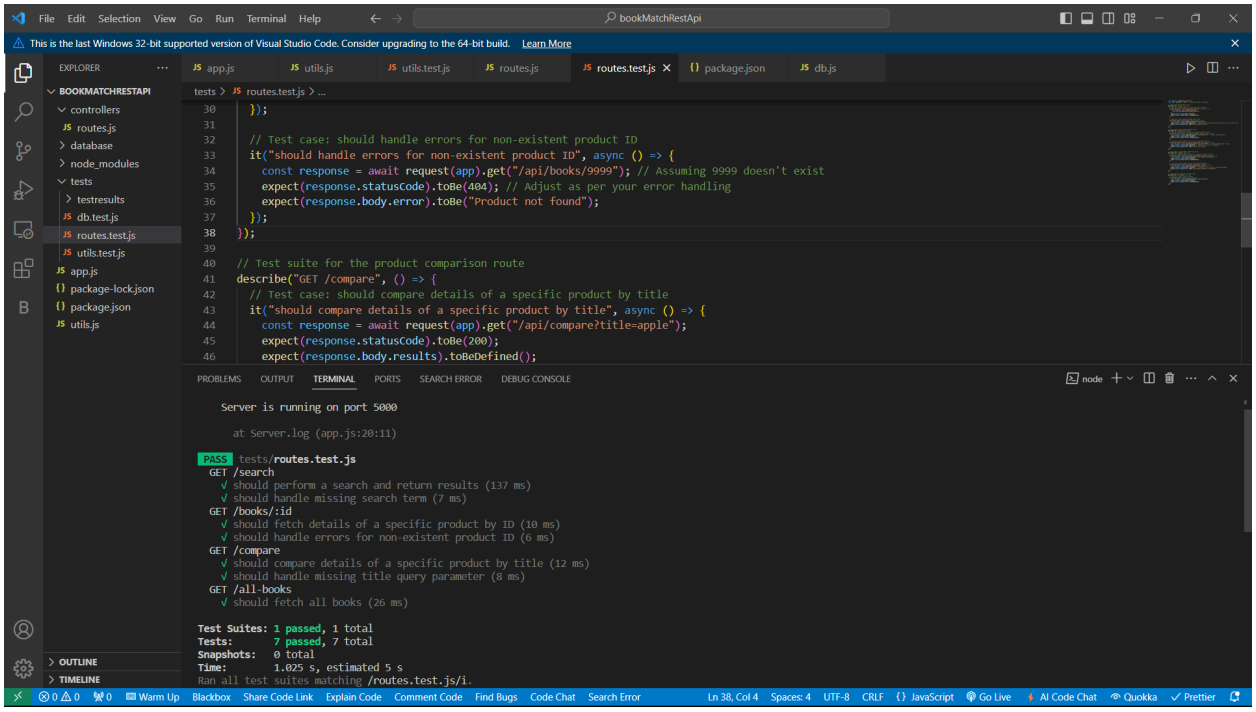
ProductDaoTest: The ProductDaoTest class contains JUnit tests designed to verify the database connection and ensure that the data scraped from websites is correctly added or updated in the database. The results of this test are displayed in the screenshot below:



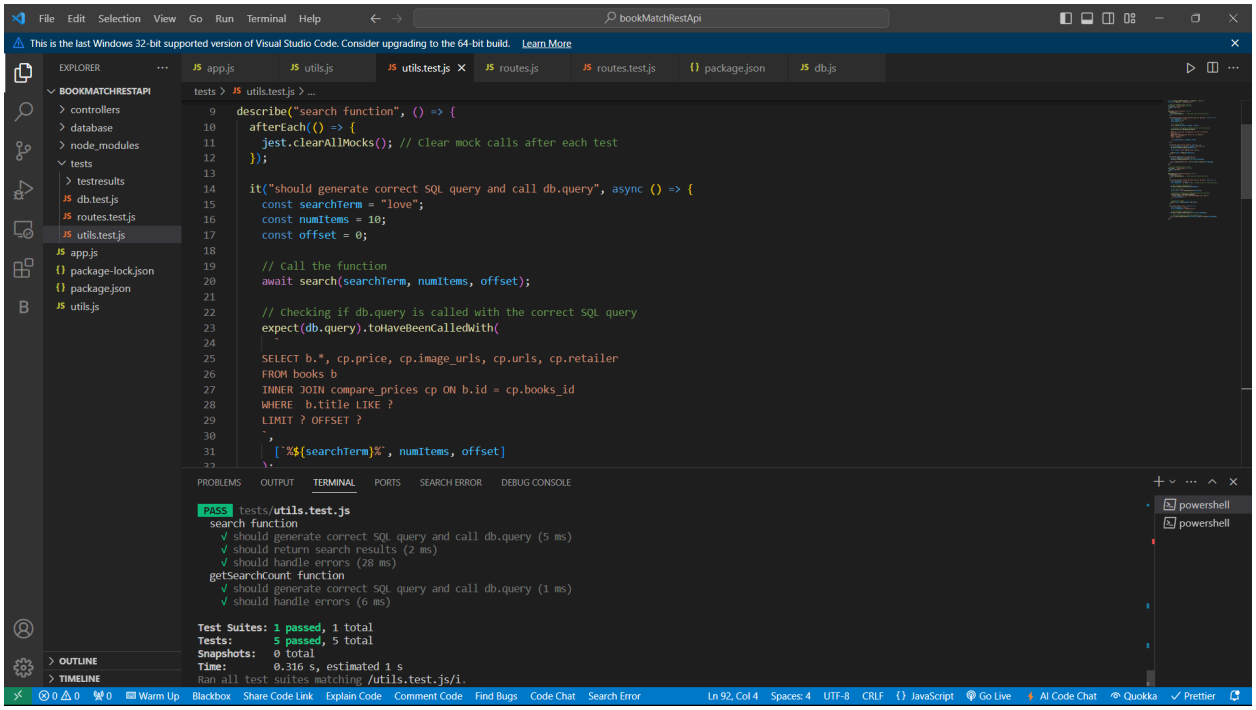
Database Connectivity: In the db.test.js file, we use Jest to write tests that ensure our Node.js application can correctly connect to the database. This file will test the connectivity and basic query operations to verify the setup.



API Routes Test: This test was carried out to ensure the correctness of the API routes in the BookMatch application. The tests verify that the routes return the expected responses and handle errors appropriately. The screenshot below shows the results of the test.



API Utilis Test: The API Utils Test in “utils.test.js” is designed to verify the correctness and robustness of utility functions used in the REST API. These utility functions typically handle various tasks such as database queries, data processing, and other helper tasks essential for the API's functionality.


File Edit Selection View Go Run Terminal Help
bookMatchRestApi
This is the last Windows 32-bit supported version of Visual Studio Code. Consider upgrading to the 64-bit build. Learn More
EXPLORER
BOOKMATCHRESTAPI
controllers
database
node_modules
tests
testresults
db.test.js
routes.test.js
utils.test.js
app.js
package-lock.json
package.json
utils.js
app.js
utils.js
utils.test.js
routes.js
routes.test.js
package.json
db.js
tests > utils.test.js > ...
describe("search function", () => {
afterEach(() => {
jest.clearAllMocks(); // Clear mock calls after each test
});
it("should generate correct SQL query and call db.query", async () => {
const searchTerm = "love";
const numItems = 10;
const offset = 0;
// Call the function
await search(searchTerm, numItems, offset);
// Checking if db.query is called with the correct SQL query
expect(db.query).toHaveBeenCalledWith(
`
SELECT b.*, cp.price, cp.image_urls, cp.urls, cp.retailer
FROM books b
INNER JOIN compare_prices cp ON b.id = cp.books_id
WHERE b.title LIKE ?
LIMIT ? OFFSET ?
`,
[`%${searchTerm}%`, numItems, offset]
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR DEBUG CONSOLE
powershell
powershell
PASS tests/utils.test.js
search function
√ should generate correct SQL query and call db.query (5 ms)
√ should return search results (2 ms)
√ should handle errors (28 ms)
getSearchCount function
√ should generate correct SQL query and call db.query (1 ms)
√ should handle errors (6 ms)
Test Suites: 1 passed, 1 total
Tests: 5 passed, 5 total
Snapshots: 0 total
Time: 0.316 s, estimated 1 s
Ran all test suites matching /utils.test.js/i.
OUTLINE
TIMELINE
Warm Up Blackbox Share Code Link Explain Code Comment Code Find Bugs Code Chat Search Error
Ln 92, Col 4 Spaces: 4 UTF-8 CRLF JavaScript Go Live AI Code Chat Quokka Prettier

Conclusion

Book Match is poised to revolutionize the way readers discover and purchase books. With its user-friendly interface, powerful search capabilities, and comprehensive price comparisons, it offers an unparalleled experience for book lovers. Leveraging advanced technologies such as web scraping, RESTful APIs, and a structured database, Book Match provides a diverse and current selection of books from various retailers.

Our rigorous testing ensures the reliability and accuracy of the data. Every feature, from seamless navigation to detailed book comparisons, is designed for user convenience. As we continue to expand our database, Book Match will remain a trusted resource for finding and comparing books, enriching the overall reading experience. We are committed to continually improving Book Match to meet the evolving needs of our users.