Price Comparison Website Report

This website is a price-comparison website that is a specialized platform for earbud's price comparison, designed to assist users in making informed purchase decisions. It empowers users to choose earbuds that match their budget and preferences. Data acquisition is accomplished through a systematic web scraping process, focusing on extracting pricing information and additional data related to earbuds from five designated websites**.**

The websites subject to data scraping include:

1. https://www.currys.co.uk

2. https://www.Argos.co.uk

3. https://www.Ebay.co.uk

4. https://www.johnlewis.com

5. https://www.backmarket.co.uk

In developing this price comparison website, many technologies were used to ensure efficiency, reliability, and seamless user experience. Leveraging a combination of Java, multithreading, Selenium, Spring, JUnit, Express.js, MySQL, Tailwind CSS and Next.js, the platform seamlessly integrates web scraping, testing, database management, REST API and frontend development. Here is a brief description of these technologies:

<u>Java and Multithreading</u>: The foundation of our data acquisition lies in Java, utilizing multithreading for concurrent processing. This ensures efficient and parallelized web scraping, enhancing the speed and responsiveness of data retrieval.

<u>Selenium for Web Scraping</u>: Selenium is employed for automated web interactions, enabling dynamic content extraction during the web scraping process. This ensures accurate and real-time data collection from various online retailers.

<u>Spring for Integration</u>: To facilitate seamless integration of web scraping processes, the Spring framework is utilized. This robust framework enhances the scalability and maintainability of the system, ensuring smooth coordination between different components.

MySQL Database: Our MySQL database serves 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 product information.

JUnit for Testing Web Scraper: To validate the reliability and accuracy of the Java web scraper, JUnit is employed for unit testing. This ensures the stability of the codebase and the dependability of the web scraping mechanisms.

REST API with Express.js: The backend of the website is powered by a Nodejs framework called Express.js, facilitating the creation of a RESTful API with three essential routes. This API seamlessly interacts with the MySQL database, enabling efficient handling of HTTP requests and responses.

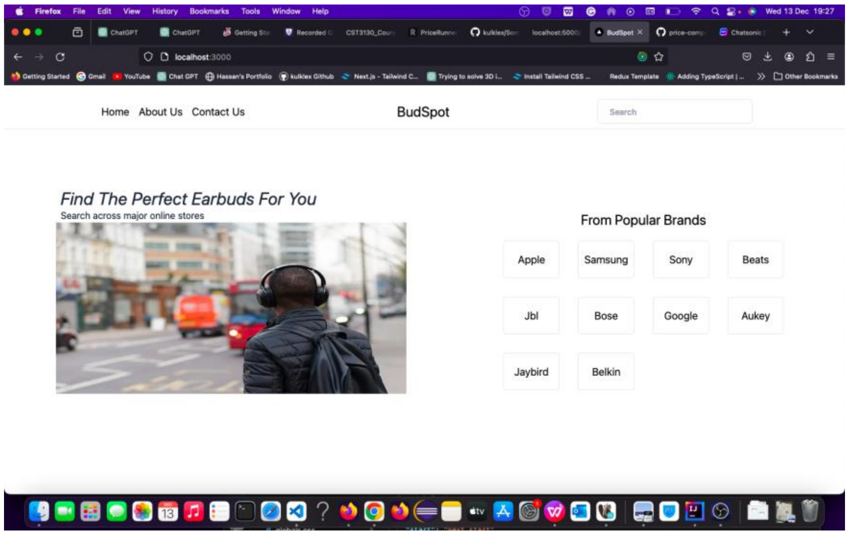
Next.js for Frontend: The frontend, developed with Next.js, leverages the power of React for efficient server-side rendering and streamlined data fetching. This ensures a responsive and dynamic user interface, with data retrieval from Express.js routes for real-time updates.

Jest for Testing API Routes and Utilities: Jest is used in this project to ensure the robustness of the Express.js API and for comprehensive testing of routes and utilities.

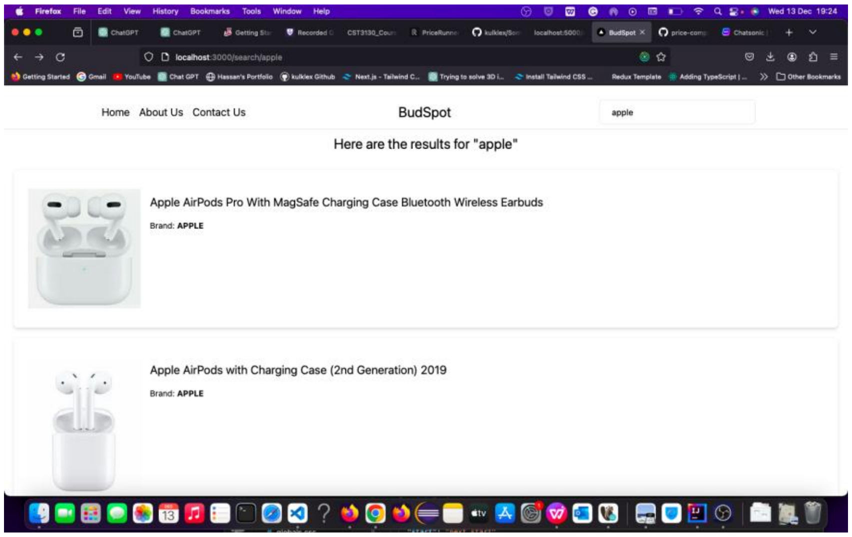
Tailwind CSS: Enhances the price-comparison website's design with its utility-first approach, providing efficient styling through pre-built classes. The framework's simplicity and consistency contribute to a visually appealing and responsive interface

WEBSITE PAGES

1. Home Page: The home page serves as an inviting entry point, offering users a warm welcome to the website. Search bar icon is placed on the Navbar, which is available to all the pages ensuring an intuitive and user-friendly starting experience.



2. Search Page: Upon interaction with the search bar icon, users are seamlessly redirected to the search page. Here, an aggregation of search results from all scraped websites awaits, providing users with a consolidated view of available options.



Here are the results for "apple"

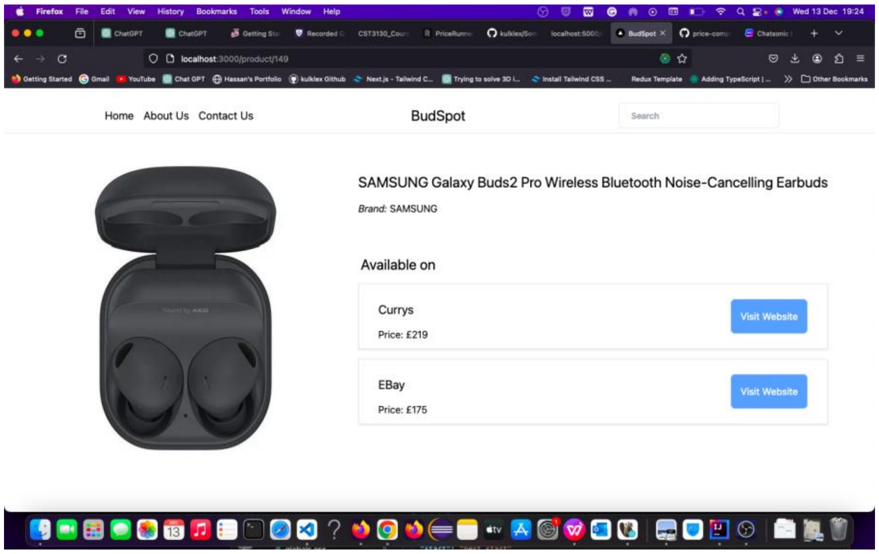
Apple AirPods Pro With MagSafe Charging Case Bluetooth Wireless Earbuds

Brand: **APPLE**

Apple AirPods with Charging Case (2nd Generation) 2019

Brand: **APPLE**

3. Comparison Page: The comparison page stands as the focal point for decisive insights. Displaying the prices of the searched results, this page empowers users to conduct a comprehensive and informed price analysis. It serves as a key resource for users seeking clarity in their purchasing decisions.



SAMSUNG Galaxy Buds2 Pro Wireless Bluetooth Noise-Cancelling Earbuds

Brand: SAMSUNG

Available on

Currys
Price: £219

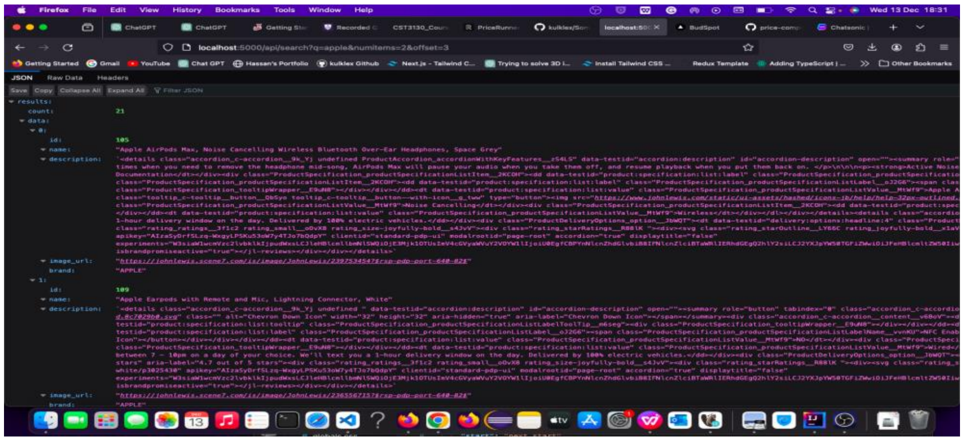
EBay
Price: £175

REST API ROUTES

These routes collectively provide a structured and reliable API for the price comparison website, ensuring accurate data retrieval and thoughtful error handling. There are as follows:

1. Search Route ("/api/search?q={searchTerm}&numitems=10&offset=20"): Using the GET method, this route facilitates a product search based on query parameters, including the search term, offset, and the number of items. It ensures data validity and sends a response with a JSON object containing the search item count and corresponding searched data.

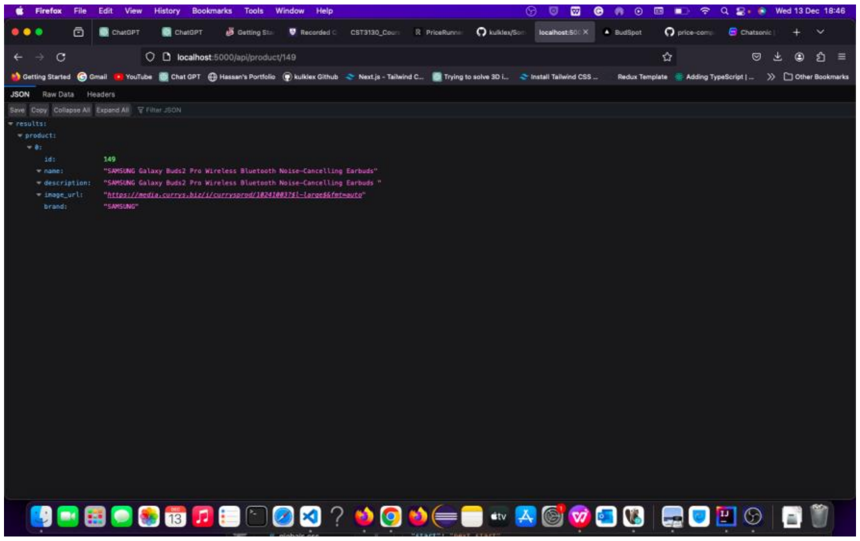
This screenshot below shows the result after using this route (where searchTerm= ‘apple,’ numitems=2 and offset=3).


Firefox File Edit View History Bookmarks Tools Window Help
Wed 13 Dec 18:31
localhost:5000/api/search?q=apple&numitems=2&offset=3
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON

```
▾ results:
    count:          21
  ▾ data:
    ▾ 0:
        id:          105
      ▾ name:        "Apple AirPods Max, Noise Cancelling Wireless Bluetooth Over-Ear Headphones, Space Grey"
      ▾ description: '<details class="accordion_c-accordion__9k_Yj undefined ProductAccordion_accordionWithKeyFeatures__zS4LS" data-testid="accordion:description" id="accordion-description" open=""><summary role="
                     times when you need to remove the headphone mid-song, AirPods Max will pause your audio when you take them off, and resume playback when you put them back on. </p>\n\n\n<p><strong>Active Noise
                     Documentation</dt></div><div class="ProductSpecification_productSpecificationListItem__2KCOH"><dd data-testid="product:specification:list:label" class="ProductSpecification_productSpecificatio
                     class="ProductSpecification_productSpecificationListItem__2KCOH"><dd data-testid="product:specification:list:label" class="ProductSpecification_productSpecificationListLabel__oJ2G6"><span clas
                     class="ProductSpecification_tooltipWrapper__E9uN8"></div></div></dd><dt data-testid="product:specification:list:value" class="ProductSpecification_productSpecificationListValue__MtWf9">Apple A
                     class="tooltip_c-tooltip__button__QbSyo tooltip_c-tooltip__button--with-icon__g_tww" type="button"><img src="https://www.johnlewis.com/static/ui-assets/hashed/icons-jb/help/help-32px-outlined.
                     class="ProductSpecification_productSpecificationListValue__MtWf9">Noise Cancelling</dt></div><div class="ProductSpecification_productSpecificationListItem__2KCOH"><dd data-testid="product:spec
                     </div></dd><dt data-testid="product:specification:list:value" class="ProductSpecification_productSpecificationListValue__MtWf9">Wireless</dt></div></dl></div></details><details class="accordio
                     1-hour delivery window on the day. Delivered by 100% electric vehicles.</dd></div><div class="ProductDeliveryOptions_option__JbWQT"><dt data-testid="delivery:options:headline:4" class="Product
                     class="rating_ratings__3f1c2 rating_small__oOvX8 rating_size-joyfully-bold__s4JvV"><div class="rating_starRatings__R8BlK "><div><svg class="rating_starOutline__LY66C rating_joyfully-bold__x1aV
                     apikey="AIzaSyDrfSLzq-WxgyLPSKu53oW7y4TJo7bQdpY" clientid="standard-pdp-ui" modalrootid="page-root" accordion="true" displaytitle="false"
                     experiments="W3siaW1wcmVzc2lvbklkIjpudWxsLCJleHBlcmltbmNlSWQiOjE3Mjk1OTUsImV4cGVyaW1uY2VOYW1lIjoiU0EgfCBPYnNlcnZhdGlvbiB0IFNlcnZlciBTaWRlIERhdGEgQ2hlY2siLCJ2YXJpYW50TGFiZWwiOiJFeHBlcmltZW50Iiw
                     isbrandpromiseactive="true"></jl-reviews></div></div></details>'
      ▾ image_url:   "https://johnlewis.scene7.com/is/image/JohnLewis/239753454?$rsp-pdp-port-640-82$"
        brand:       "APPLE"
    ▾ 1:
        id:          109
      ▾ name:        "Apple Earpods with Remote and Mic, Lightning Connector, White"
      ▾ description: '<details class="accordion_c-accordion__9k_Yj undefined " data-testid="accordion:description" id="accordion-description" open=""><summary role="button" tabindex="0" class="accordion_c-accordio
                     d.0c70290d.svg" class="" alt="Chevron Down Icon" width="32" height="32" aria-hidden="true" aria-label="Chevron Down Icon"></span></summary><div class="accordion_c-accordion__content__v68oV"><d
                     testid="product:specification:list:tooltip" class="ProductSpecification_productSpecificationListLabelToolTip__m6seg"><div class="ProductSpecification_tooltipWrapper__E9uN8"></div></div></dd><d
                     testid="product:specification:list:label" class="ProductSpecification_productSpecificationListLabel__oJ2G6"><span class="ProductSpecification_productSpecificationListLabelName__vvnKU">NFC Enab
                     Icon"></button></div></div></div></dd><dt data-testid="product:specification:list:value" class="ProductSpecification_productSpecificationListValue__MtWf9">NO</dt></div><div class="ProductSpeci
                     class="ProductSpecification_tooltipWrapper__E9uN8"></div></div></dd><dt data-testid="product:specification:list:value" class="ProductSpecification_productSpecificationListValue__MtWf9">Wired</
                     between 7 - 10pm on a day of your choice. We'll text you a 1-hour delivery window on the day. Delivered by 100% electric vehicles.</dd></div><div class="ProductDeliveryOptions_option__JbWQT"><
                     stars" aria-label="4.7 out of 5 stars"><div class="rating_ratings__3f1c2 rating_small__oOvX8 rating_size-joyfully-bold__s4JvV"><div class="rating_starRatings__R8BlK "><div><svg class="rating_s
                     white/p3025430" apikey="AIzaSyDrfSLzq-WxgyLPSKu53oW7y4TJo7bQdpY" clientid="standard-pdp-ui" modalrootid="page-root" accordion="true" displaytitle="false"
                     experiments="W3siaW1wcmVzc2lvbklkIjpudWxsLCJleHBlcmltbmNlSWQiOjE3Mjk1OTUsImV4cGVyaW1uY2VOYW1lIjoiU0EgfCBPYnNlcnZhdGlvbiB0IFNlcnZlciBTaWRlIERhdGEgQ2hlY2siLCJ2YXJpYW50TGFiZWwiOiJFeHBlcmltZW50Iiw
                     isbrandpromiseactive="true"></jl-reviews></div></div></details>'
      ▾ image_url:   "https://johnlewis.scene7.com/is/image/JohnLewis/236556715?$rsp-pdp-port-640-82$"
        brand:       "APPLE"
```

2. Product Details Route ("/api/product/:productID"): Using the GET method, it is specifically designed to fetch detailed information about a product identified by its unique ID. It queries the database using the provided product ID and responds with a JSON object containing product details.

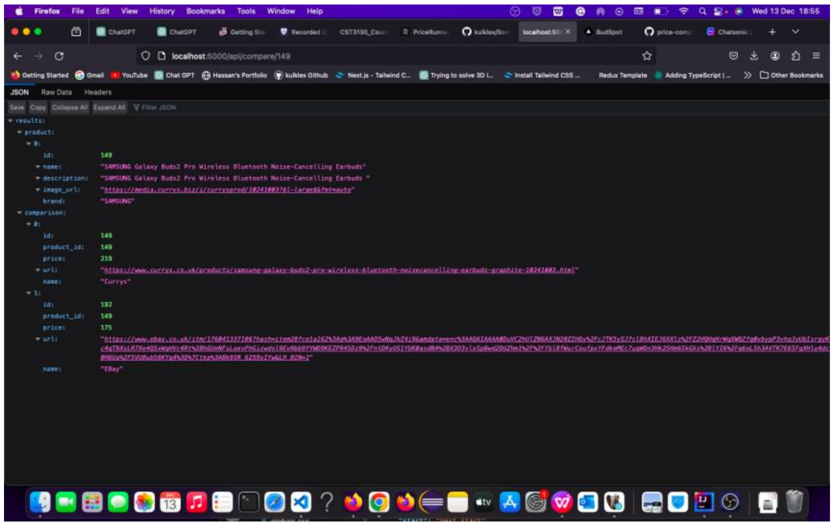
The screenshot below shows the result after using this route (where productID is ‘149’).



```
localhost:5000/api/product/149

▾ results:
  ▾ product:
    ▾ 0:
        id:            149
      ▾ name:          "SAMSUNG Galaxy Buds2 Pro Wireless Bluetooth Noise-Cancelling Earbuds"
      ▾ description:   "SAMSUNG Galaxy Buds2 Pro Wireless Bluetooth Noise-Cancelling Earbuds "
      ▾ image_url:     "https://media.currys.biz/i/currysprod/10241003?$l-large$&fmt=auto"
        brand:         "SAMSUNG"
```

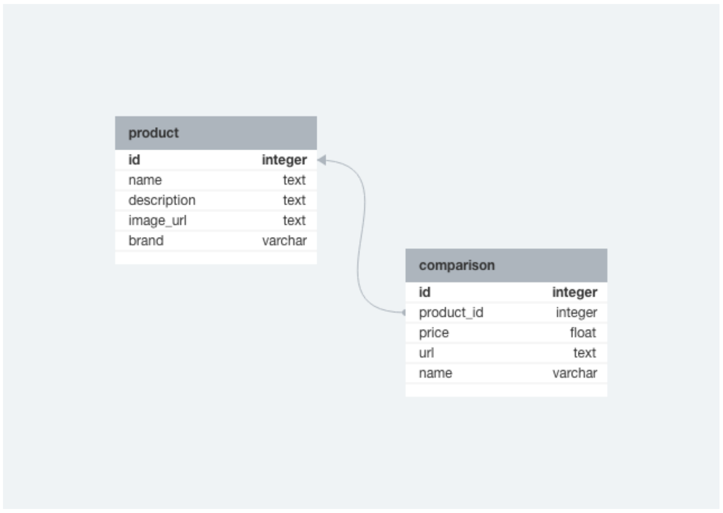
3. Product Comparison Route ("/api/compare/:productID"): Using the GET method, it is tailored for comparing details of a specific product, including additional comparison data. It utilizes the product ID from the route parameter, fetches both product details and comparison data from the database, and responds with a comprehensive JSON object.

The screenshot below shows the result after using this route (where productID is ‘149’).



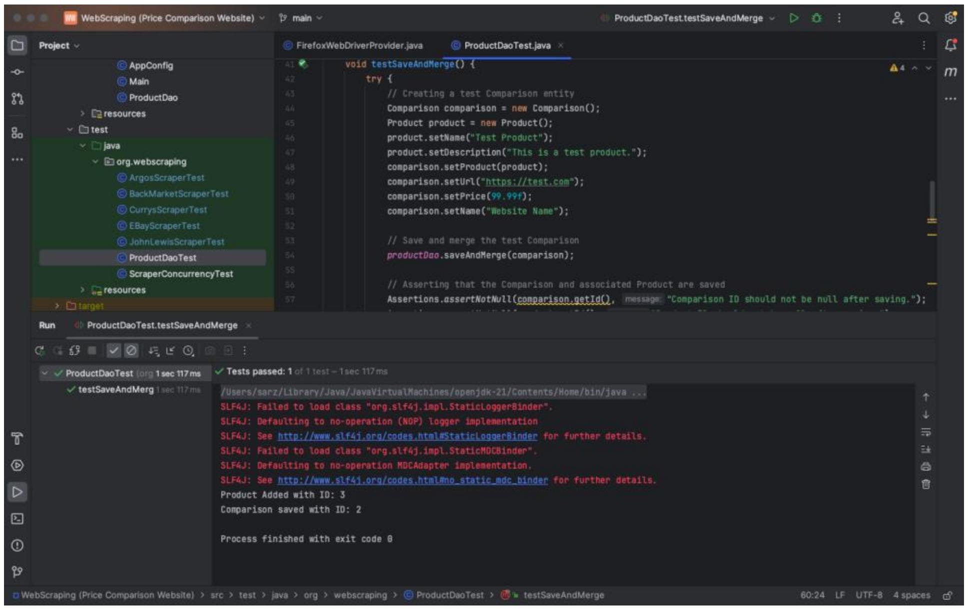
```
localhost:5000/api/compare/149

▾ results:
  ▾ product:
    ▾ 0:
        id:            149
      ▾ name:          "SAMSUNG Galaxy Buds2 Pro Wireless Bluetooth Noise-Cancelling Earbuds"
      ▾ description:   "SAMSUNG Galaxy Buds2 Pro Wireless Bluetooth Noise-Cancelling Earbuds "
      ▾ image_url:     "https://media.currys.biz/i/currysprod/10241003?$l-large$&fmt=auto"
        brand:         "SAMSUNG"
  ▾ comparison:
    ▾ 0:
        id:            149
        product_id:    149
        price:         219
      ▾ url:           "https://www.currys.co.uk/products/samsung-galaxy-buds2-pro-wireless-bluetooth-noisecancelling-earbuds-graphite-10241003.html"
        name:          "Currys"
    ▾ 1:
        id:            192
        product_id:    149
        price:         175
      ▾ url:           "https://www.ebay.co.uk/itm/176041337186?hash=item28fce1a162%3Ag%3A9EoAAOSwNqJkZ4i96amdata=enc%3AAQAIAAAA0DuVC2hU1ZN6AXJN28ZIHDy%2FcJTK5y5J7cl8hXIEJ6XXls%2FZ2HQHgHrWqXW0Zfg0vbypP3yhqJyUbIsrgyK[illegible]
                       c4qT9XsLR7Xe4Q5xWgHVc4Rt%2BhGUeNFsLoevPhGizwdsl9EvRbb9YYWD9KEZP945Dz9%2FntDKyO5IYUK0asdBA%2BX3O3ylxSpBwd20UZhm1%2F%2Ffbl8fWurCoufpxYFdkeMEc7ugW0n3Hk25Hm6SkGks%2BTYI6%2Fg6oL5h3AVTK7E65FgXHlu4dc[illegible]
                       BH6Ua%2F5VUBub56KYg4%3D%7Ctkp%3ABk9SR_6259vIYw6LH_BIN=1"
        name:          "EBay"
```

DATABASE DESIGN



TESTS SCREENSHOTS

ProductDao Test: A JUnit test for the 'saveAndMerge' method in the ProductDao class which saves, or merges data from the scraped sites to the database.



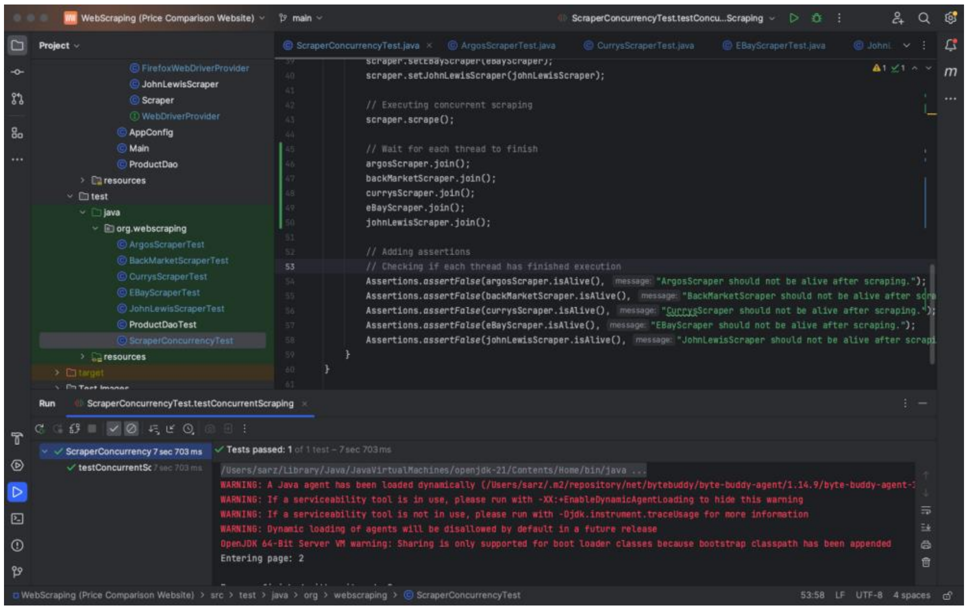
```java
    void testSaveAndMerge() {
        try {
            // Creating a test Comparison entity
            Comparison comparison = new Comparison();
            Product product = new Product();
            product.setName("Test Product");
            product.setDescription("This is a test product.");
            comparison.setProduct(product);
            comparison.setUrl("https://test.com");
            comparison.setPrice(99.99f);
            comparison.setName("Website Name");

            // Save and merge the test Comparison
            productDao.saveAndMerge(comparison);

            // Asserting that the Comparison and associated Product are saved
            Assertions.assertNotNull(comparison.getId(), message: "Comparison ID should not be null after saving.");
```

Tests passed: 1 of 1 test – 1 sec 117 ms

```
/Users/sarz/Library/Java/JavaVirtualMachines/openjdk-21/Contents/Home/bin/java ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBinder".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no_static_mdc_binder for further details.
Product Added with ID: 3
Comparison saved with ID: 2

Process finished with exit code 0
```

WebScraping (Price Comparison Website)
main
ProductDaoTest.testSaveAndMerge
FirefoxWebDriverProvider.java
ProductDaoTest.java
Project
AppConfig
Main
ProductDao
resources
test
java
org.webscraping
ArgosScraperTest
BackMarketScraperTest
CurrysScraperTest
EBayScraperTest
JohnLewisScraperTest
ProductDaoTest
ScraperConcurrencyTest
resources
target
Run
ProductDaoTest.testSaveAndMerge
ProductDaoTest (org 1 sec 117 ms
testSaveAndMerg 1 sec 117 ms
WebScraping (Price Comparison Website) › src › test › java › org › webscraping › ProductDaoTest › testSaveAndMerge
60:24 LF UTF-8 4 spaces

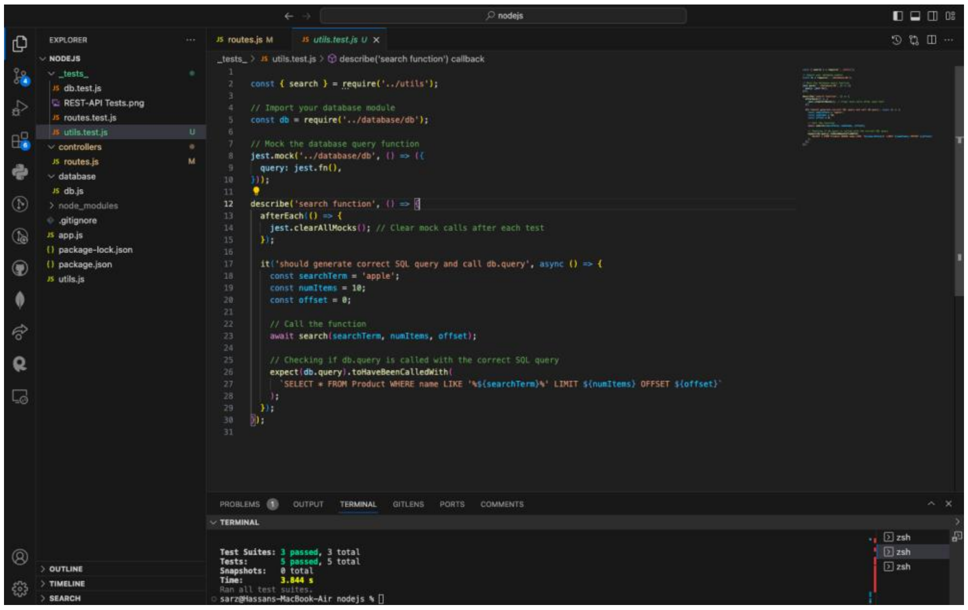
Scrapper Concurrency Test: A JUnit test that checks if the scrapers are running concurrently.



```java
        scraper.setJohnLewisScraper(johnLewisScraper);

        // Executing concurrent scraping
        scraper.scrape();

        // Wait for each thread to finish
        argosScraper.join();
        backMarketScraper.join();
        currysScraper.join();
        eBayScraper.join();
        johnLewisScraper.join();

        // Adding assertions
        // Checking if each thread has finished execution
        Assertions.assertFalse(argosScraper.isAlive(), message: "ArgosScraper should not be alive after scraping.");
        Assertions.assertFalse(backMarketScraper.isAlive(), message: "BackMarketScraper should not be alive after scra
        Assertions.assertFalse(currysScraper.isAlive(), message: "CurrysScraper should not be alive after scraping.");
        Assertions.assertFalse(eBayScraper.isAlive(), message: "EBayScraper should not be alive after scraping.");
        Assertions.assertFalse(johnLewisScraper.isAlive(), message: "JohnLewisScraper should not be alive after scrapi
    }
}
```

```
Tests passed: 1 of 1 test – 7 sec 703 ms
/Users/sarz/Library/Java/JavaVirtualMachines/openjdk-21/Contents/Home/bin/java ...
WARNING: A Java agent has been loaded dynamically (/Users/sarz/.m2/repository/net/bytebuddy/byte-buddy-agent/1.14.9/byte-buddy-agent-1
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
Entering page: 2
```

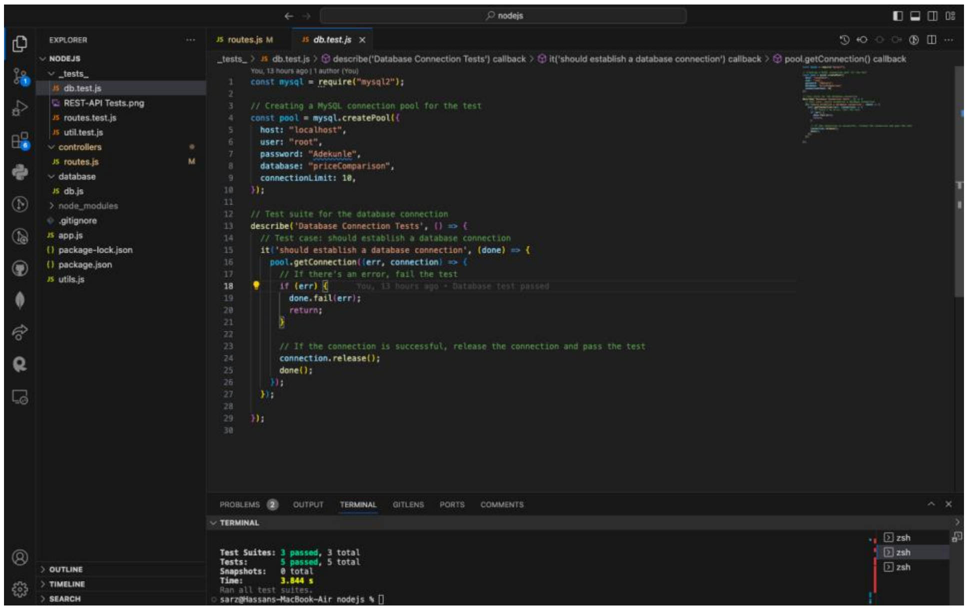
Utility Test: A Jest test that checks a utility used for the search route



```javascript

const { search } = require('../utils');

// Import your database module
const db = require('../database/db');

// Mock the database query function
jest.mock('../database/db', () => ({
  query: jest.fn(),
}));

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 = 'apple';
    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 * FROM Product WHERE name LIKE '%${searchTerm}%' LIMIT ${numItems} OFFSET ${offset}`
    );
  });
});

```

```
Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        3.844 s
Ran all test suites.
sarz@Hassans-MacBook-Air nodejs % 
```

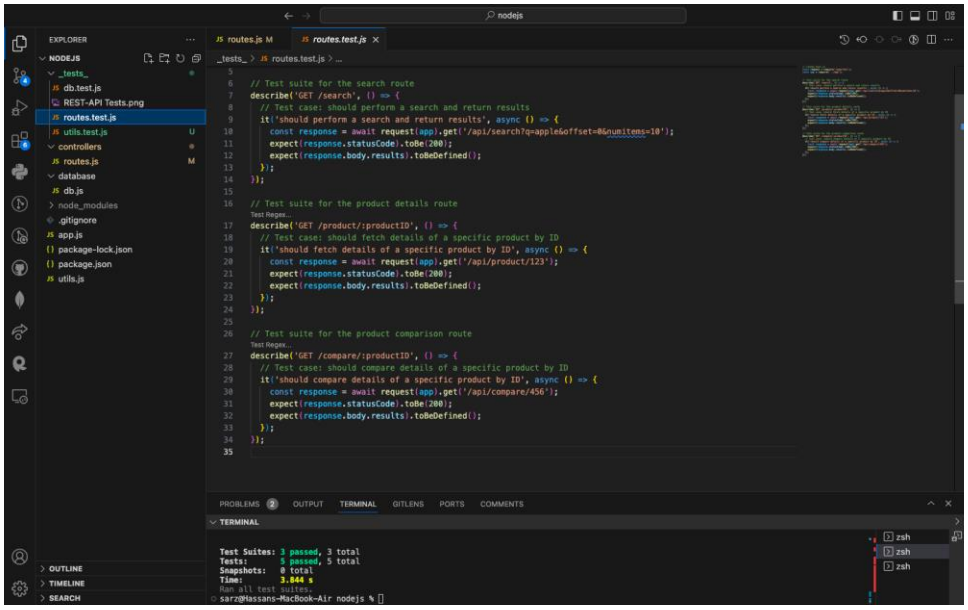
Database Connection Test: A Jest test that checks the database connection to the REST API.



```javascript
const mysql = require("mysql2");

// Creating a MySQL connection pool for the test
const pool = mysql.createPool({
  host: "localhost",
  user: "root",
  password: "Adekunle",
  database: "priceComparison",
  connectionLimit: 10,
});

// Test suite for the database connection
describe('Database Connection Tests', () => {
  // Test case: should establish a database connection
  it('should establish a database connection', (done) => {
    pool.getConnection((err, connection) => {
      // If there's an error, fail the test
      if (err) {      You, 13 hours ago • Database test passed
        done.fail(err);
        return;
      }

      // If the connection is successful, release the connection and pass the test
      connection.release();
      done();
    });
  });

});
```

```
Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        3.844 s
Ran all test suites.
sarz@Hassans-MacBook-Air nodejs % 
```

Routes Tests: Three Jest tests checking three routes.



```javascript

// Test suite for the search route
describe('GET /search', () => {
  // Test case: should perform a search and return results
  it('should perform a search and return results', async () => {
    const response = await request(app).get('/api/search?q=apple&offset=0&numitems=10');
    expect(response.statusCode).toBe(200);
    expect(response.body.results).toBeDefined();
  });
});

// Test suite for the product details route
Test Regex...
describe('GET /product/:productID', () => {
  // Test case: should fetch details of a specific product by ID
  it('should fetch details of a specific product by ID', async () => {
    const response = await request(app).get('/api/product/123');
    expect(response.statusCode).toBe(200);
    expect(response.body.results).toBeDefined();
  });
});

// Test suite for the product comparison route
Test Regex...
describe('GET /compare/:productID', () => {
  // Test case: should compare details of a specific product by ID
  it('should compare details of a specific product by ID', async () => {
    const response = await request(app).get('/api/compare/456');
    expect(response.statusCode).toBe(200);
    expect(response.body.results).toBeDefined();
  });
});

```

```
Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        3.844 s
Ran all test suites.
sarz@Hassans-MacBook-Air nodejs % 
```