**First backend/frontend code development**

# Overview:

This code aims to create a web application for scraping product information from Coles, then storing it in a MongoDB database, and providing search functionality for users to find products based on their titles.

# Key Components:

|  |  |
| --- | --- |
| Components | Aims |
| Express Server Setup (`app.js`) | * Sets up an Express server to handle HTTP requests. * Configures the view engine and static file serving. * Connects to MongoDB using Mongoose. |
| Routes (`router.js`) | * Defines routes for different functionalities of the web application. * `/scrape`: Renders a page for initiating the scraping process. * `/searchProduct`: Renders a page for users to search for products. * `/scrapeResults`: Handles the scraping process, saves scraped data to the database, and renders the results. * `/searchResults`: Handles searching for products based on title in the database and renders the results. |
| Scraping Functionality (`scrapeData.js`) | * Contains the `scrapeData` function responsible for scraping product information from a given URL. * Uses Puppeteer to launch a headless browser, navigate to the URL, and extract relevant data using Cheerio. |
| Model (`Product`) | * Defines the schema for the product model to be stored in the MongoDB database. * Includes fields such as title, price, discount, product details, and image URL. |
| View Templates (`.ejs` files) | * Contains EJS templates for rendering HTML pages dynamically with data. * Templates include pages for initiating scraping, displaying scraping results, searching for products, and displaying search results. |

# Main Functionality:

* 1. Users can initiate the scraping process by visiting the `/scrape` route, providing a URL to scrape product information.
  2. Scraped data is saved to the MongoDB database, avoiding duplication by checking if a product with the same title already exists.
  3. Users can search for products based on their titles using the `/searchProduct` route, and the application retrieves matching products from the database.

Conclusion

The developed web application provides a convenient way to scrape and store product information, enabling users to search for products efficiently. It demonstrates the integration of web scraping, database storage, and server-side rendering using Node.js, Express, Puppeteer, and MongoDB.