**First backend/frontend code development**

**1. 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. Additionally, frontend components have been added to enhance the user experience, including an index page with stylish styling and authentication features.

# 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. * `/index`: Handles searching for products based on title in the database and renders the results. * `/login`: Renders a login page. * /signup: Renders a signup page. * `/scrapeResults`: Handles the scraping process, saves scraped data to the database, and renders the results. * `/scrape`: Renders a page for initiating the scraping process. |
| 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. |
| Model (User) | * User: Defines the schema for the user model to be stored in the MongoDB database, including fields such as username, email, and hashed password. |
| 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, as well as login and signup pages with stylish styling. |

# Main Functionality:

* Users can register for an account, log in, and log out using the authentication functionality provided, with user data securely stored in the database. Passwords are hashed before being stored in the database, ensuring that even if the database is compromised, the passwords cannot be easily decrypted.
* Users can search for products based on their titles using the `/searchProduct` route, and the application retrieves matching products from the database.
* Users can initiate the scraping process by visiting the `/scrape` route, providing a URL to scrape product information.
* Scraped data is saved to the MongoDB database, avoiding duplication by checking if a product with the same title already exists.

Conclusion

The enhanced web application now offers a seamless integration of frontend and backend components, providing users with an aesthetically pleasing interface, efficient product scraping and search capabilities, and secure authentication features. It showcases the versatility and robustness of technologies such as Node.js, Express, Puppeteer, MongoDB, and Passport.js in building modern web applications. The addition of user authentication functionality enhances the overall user experience and ensures secure access to the application's features. The implementation of password hashing further enhances security by protecting user passwords from unauthorized access.