**DARAZ SCRAPER**

1. Introduction

* **Project Overview:**
  + 1. This project involves the development of a web scraping and API application using Flask and Selenium.
    2. The application aims to scrape product data from the Daraz.pk website and store it in a MySQL database.
    3. Users can then access the scraped data through API endpoints to retrieve information about products.
* **Objective:**
  + 1. The main objective of the project is to automate the process of scraping product data from Daraz.pk.
    2. It aims to provide a convenient and efficient way to gather information about products available on the website.
    3. Additionally, the project intends to showcase the integration of web scraping, database management, and API development.
* **Technologies Used:**
  + 1. Flask: It is a lightweight Python web framework used for developing the API endpoints.
    2. Selenium: It is a web automation tool used for navigating web pages and extracting data from dynamic websites.
    3. MySQL: It is a popular open-source relational database management system used to store the scraped product data.
    4. Python: The project is implemented using the Python programming language.

1. Methodology
   * **Overview of Web Scraping:**
     1. Web scraping is the process of extracting data from websites automatically.
     2. It involves sending HTTP requests to web pages, parsing the HTML content, and extracting specific information.
     3. Web scraping enables automated data collection, analysis, and integration into other applications.
   * **Selenium for Web Scraping:**
     1. Selenium is a popular web automation tool used for web scraping.
     2. It provides a programming interface to control web browsers programmatically.
     3. With Selenium, you can navigate web pages, interact with elements, and extract data from dynamic websites that require JavaScript execution.
     4. In the provided code, Selenium is used to navigate to the Daraz.pk website, locate specific elements (product names and prices), and extract their text values.
   * **MySQL Database Integration:**
     1. MySQL is an open-source relational database management system.
     2. It is commonly used for storing structured data in a tabular format.
     3. In the provided code, a MySQL database is utilized to store the scraped product data.
     4. Two tables, "all\_data" and "latest\_data," are created with specified headers to store the scraped data.
     5. The scraped product information, including the search type, name, price, and timestamp, is inserted into these tables.
   * **Flask Framework for API Development:**
     1. Flask is a lightweight Python web framework used for developing web applications and APIs.
     2. It provides tools and libraries for routing HTTP requests, handling responses, and building RESTful APIs.
     3. In the provided code, Flask is used to create API endpoints for the application.
     4. The '/scrape' endpoint triggers the web scraping process, where the search type is extracted from the query parameter.
     5. The scraped data can be accessed through the '/data' endpoint, returning the latest product data stored in the "latest\_data" table.
     6. The '/id\_data' endpoint allows retrieving specific product data based on the 'search\_id' query parameter, querying the "all\_data" table.
2. Implementation Details
   * **Setting up the Development Environment:**
     1. To set up the development environment, you need to ensure you have Python installed on your system.
     2. Install a code editor or IDE of your choice, such as Visual Studio Code or PyCharm, to write and manage your code.
     3. Set up a virtual environment to keep your project dependencies isolated.
     4. Clone or download the project code from a version control repository or create a new project directory.
   * **Installing Dependencies:**
     1. Before running the code, you need to install the required dependencies.
     2. The main dependencies for this project include Flask, Selenium, and mysql-connector-python.
     3. You can use pip, the Python package installer, to install these dependencies by running the command pip install flask selenium mysql-connector-python.
   * **Configuration and Database Setup:**
     1. In the provided code, the MySQL database is used for storing the scraped data.
     2. Update the database connection details in the code to match your MySQL server configuration.
     3. Set the host, user, password, and database name according to your MySQL setup.
     4. Ensure that your MySQL server is running before executing the code.
   * **Database Schema Design:**
     1. In the code, two database tables are created: "all\_data" and "latest\_data."
     2. The "all\_data" table stores all the scraped product data, while the "latest\_data" table stores the latest scraped data.
   * **The schema design includes the following columns:**
     1. id: An auto-incrementing primary key for each record.
     2. search\_type: Represents the type of product being searched (e.g., "mobile", "laptop").
     3. name: Stores the name of the product.
     4. price: Stores the price of the product.
     5. timestamp: Represents the timestamp when the data was inserted, with a default value of the current timestamp.
     6. The schema design allows storing and retrieving product data based on search types and individual record IDs.
3. Web Scraping Process
   * **Interaction with daraz.pk Website:**
     1. The code interacts with the daraz.pk website to scrape product data.
     2. It uses the Selenium library to automate the web browser and navigate to the desired page.
     3. The URL of the page to scrape is constructed dynamically by appending the search\_type parameter to the daraz.pk catalog URL.
     4. The code sends a GET request to the constructed URL and retrieves the HTML content of the page.
   * **Extracting Product Names and Prices:**
     1. After loading the page, the code uses Selenium's find\_elements method to locate specific elements on the page.
     2. CSS selectors are used to identify the product names and prices on the daraz.pk website.
     3. The code finds all the elements with the specified CSS selectors and stores them in the names and prices variables, respectively.
     4. The extracted product names and prices are stored in the database for further processing and analysis.
   * **Handling Dynamic Content:**
     1. Since the daraz.pk website may have dynamic content that loads asynchronously, the code uses the implicitly\_wait method provided by Selenium.
     2. This method instructs the web driver to wait for a specified amount of time (in seconds) to allow dynamic content to load before attempting to extract the data.
     3. Adjusting the wait time may be necessary depending on the specific website and its loading speed.
   * **Storing Data in the Database:**
     1. The code utilizes a MySQL database to store the scraped product data.
     2. Two tables are created: "all\_data" and "latest\_data," as mentioned earlier.
     3. The product names and prices extracted from the daraz.pk website are inserted into both tables using the INSERT INTO SQL statements.
     4. Before inserting the latest data, the "latest\_data" table is truncated to ensure it only contains the most recent data.
     5. After inserting the data, the changes are committed to the database using the commit method.API Development
   * **Routes and Endpoints:**
     1. The Flask application defines several routes and endpoints to handle different functionalities.
     2. The /scrape endpoint is responsible for triggering the web scraping process. It accepts a GET request and extracts the search\_type parameter from the query string. The code then uses Selenium to scrape the data from the daraz.pk website based on the provided search type.
     3. The /data endpoint is used to retrieve the latest scraped data from the "latest\_data" table in the database. It accepts a GET request and fetches all the rows from the table. The data is then converted to a list of dictionaries and serialized to JSON before being returned as the response.
     4. The /id\_data endpoint allows retrieving data from the "all\_data" table based on the provided search\_id. It accepts a GET request and uses the search\_id parameter to query the database for the corresponding row. Similar to the /data endpoint, the retrieved data is serialized to JSON and returned as the response.
   * **Request Handling and Parameter Extraction:**
     1. Flask's request object is used to handle incoming requests and extract parameters from the query string.
     2. In the /scrape endpoint, the search\_type parameter is extracted using request.args.get('search\_type').
     3. In the /id\_data endpoint, the search\_id parameter is extracted using request.args.get('search\_id').
     4. These parameters are then used in the respective functionalities of the code, such as constructing the URL for web scraping or querying the database.

**Scrape Data Examples:**

The code utilizes Selenium to scrape data from the daraz.pk website based on the provided search type.

The /scrape endpoint is responsible for initiating the scraping process.

It constructs the URL using the search type parameter and navigates to the corresponding page on the daraz.pk website.

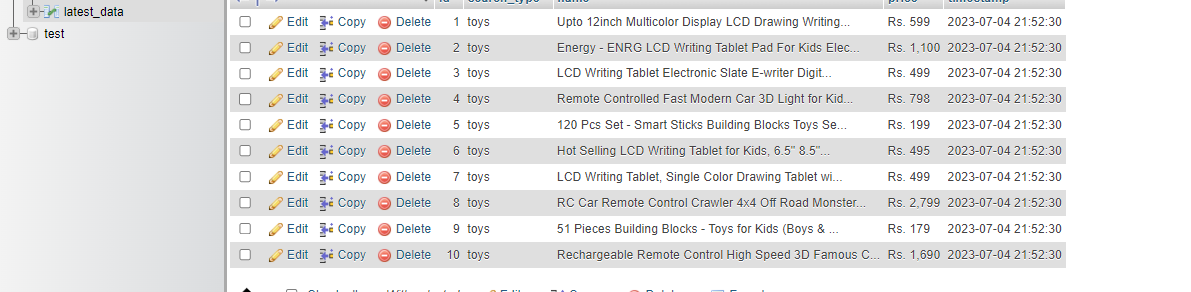
The code then uses Selenium's find\_elements method to locate the product names and prices on the page.

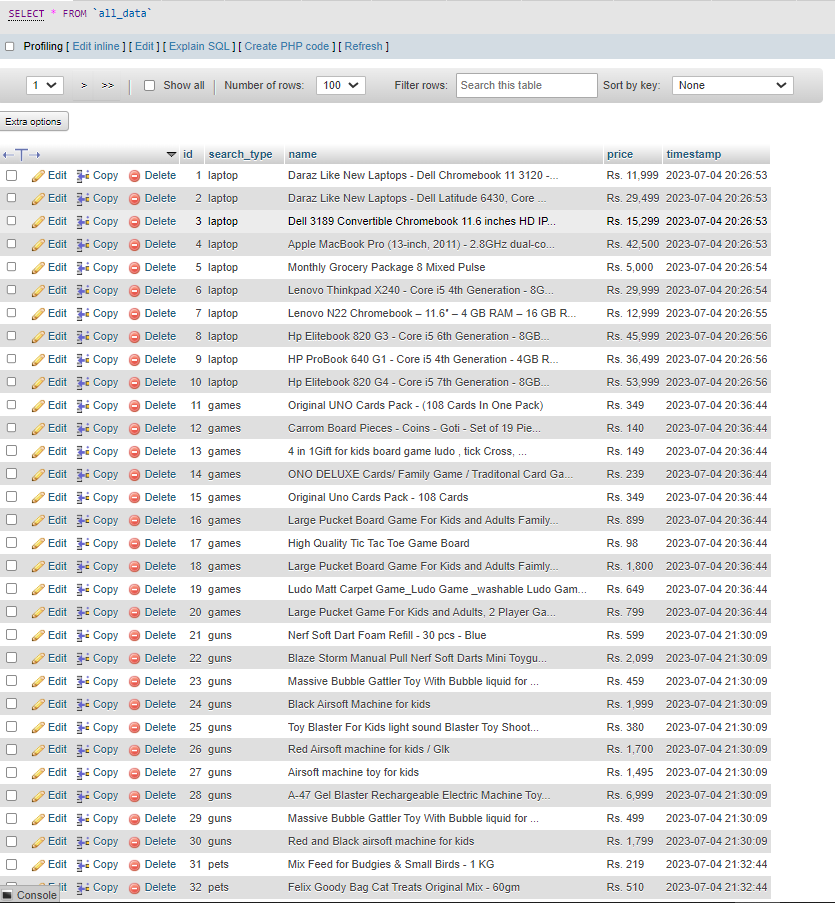
It iterates over the elements and extracts the text values of each product name and price.

The extracted data is then stored in the MySQL database, both in the all\_data table and the latest\_data table.

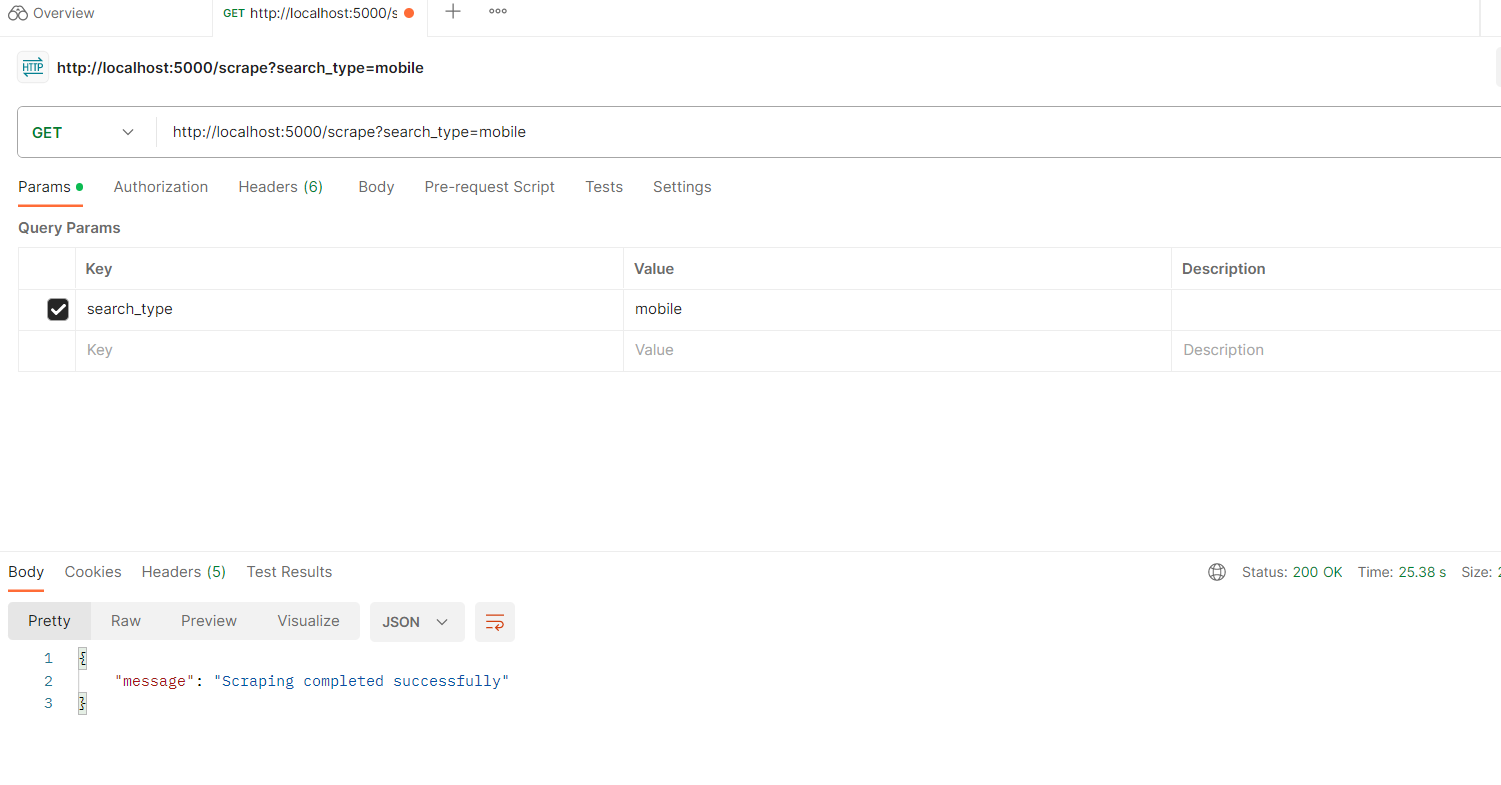
The latest\_data table holds the most recent scraped data, which can be accessed through the /data endpoint.

**Latest data Table**



**All Data Table**

**Inserting Data through Flask and Postman**



**Retrieving Data through API Endpoints:**

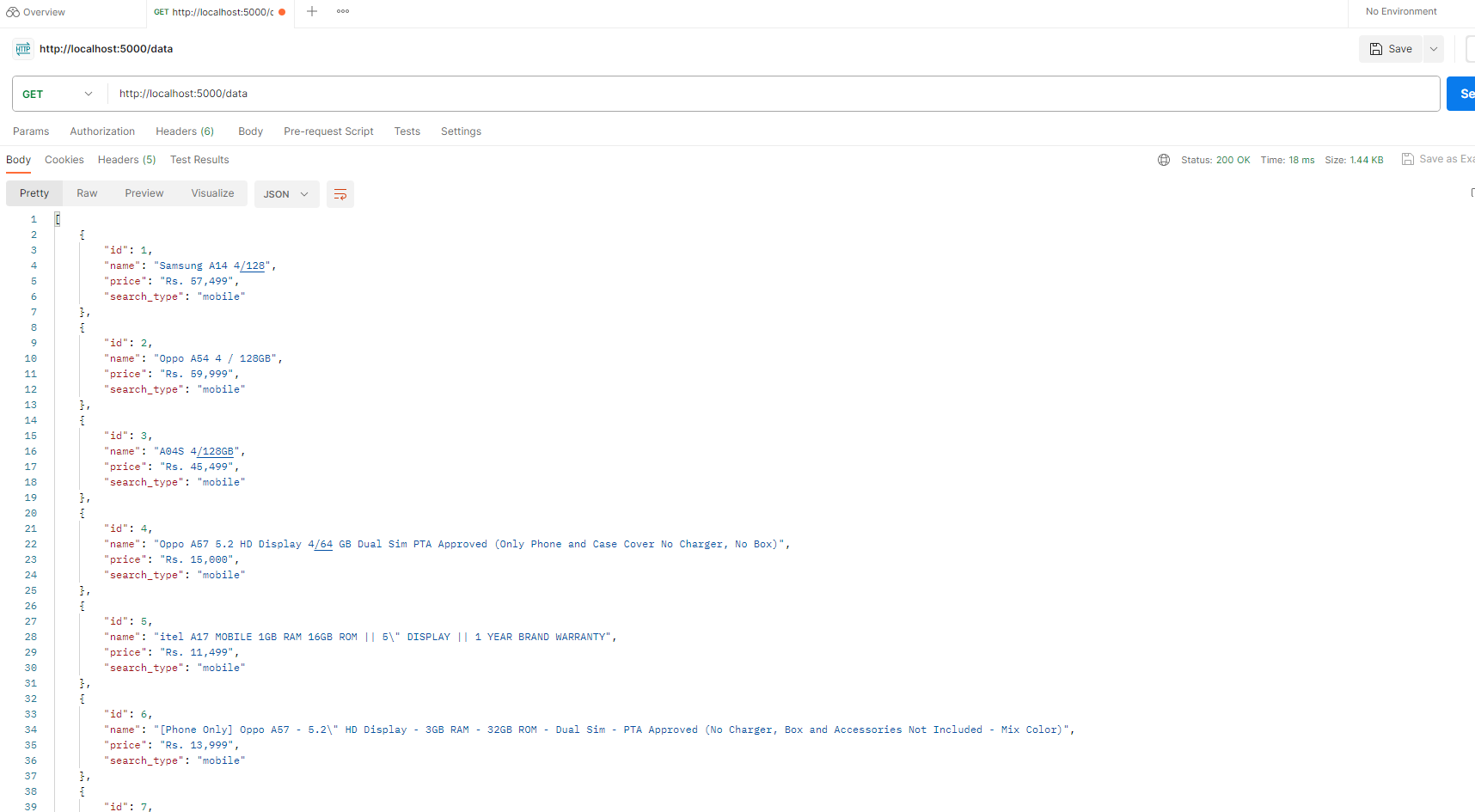
The /data endpoint allows retrieving the latest scraped data from the latest\_data table in the MySQL database.

It executes a SELECT query to fetch all rows from the latest\_data table.

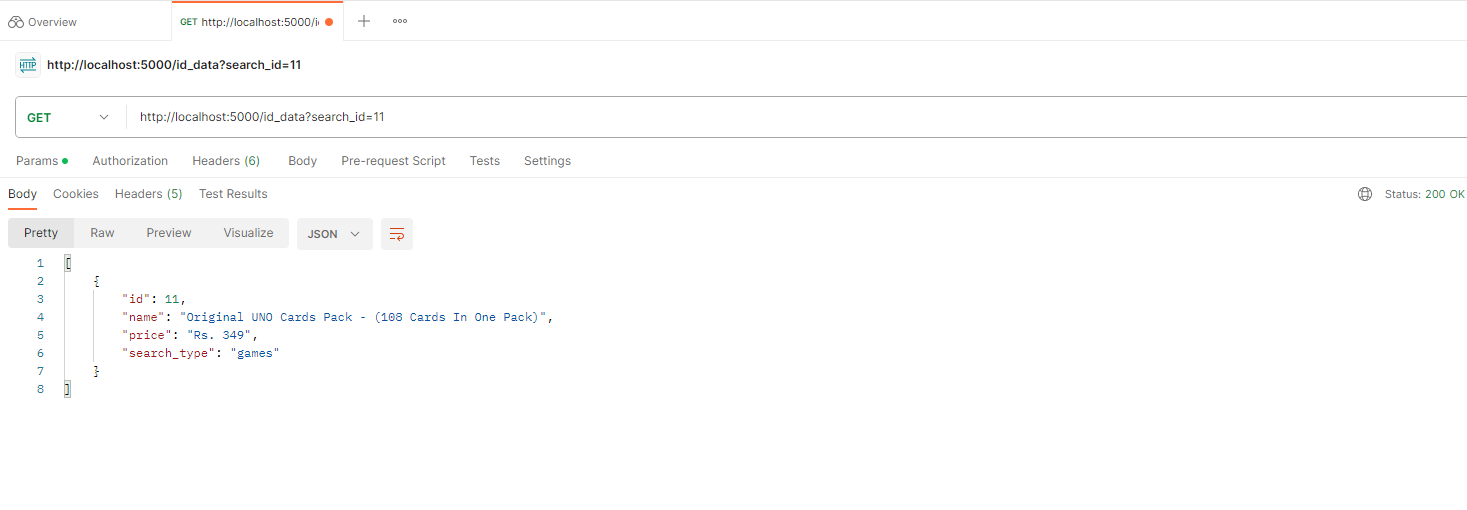
The fetched rows are converted into a list of dictionaries, where each dictionary represents a product with its attributes (id, search\_type, name, price).

The data is then serialized to JSON format using jsonify and returned as the API response.

**Fetches the Latest top 10 Searches**



**Search by given Id**



1. Discussion
   * **Challenges Faced:**
     1. Handling dynamic content: One of the challenges in web scraping is dealing with websites that load content dynamically using JavaScript. In this project, Selenium was used to handle dynamic content, but it can sometimes be complex to identify and extract the desired data.
     2. Performance considerations: Web scraping can be resource-intensive, especially when scraping a large number of pages or dealing with complex websites. Balancing the performance requirements with the limitations of the target website is essential.

Thank You