Project 1: Web Scraping E-Commerce Flask App

📌 1. Project Title

Web Scraping E-Commerce Data with Flask Display

📝 2. Objective

To build a web application that scrapes data from an e-commerce website (BooksToScrape), stores the data in an Excel file, and presents it in a user-friendly interface using Flask with added search, pagination, and image display features.

🔍 3. Problem Statement

Manually browsing product data from e-commerce websites is time-consuming and repetitive. This project automates the data extraction process and displays it in an organized, searchable, and visual way, ideal for analysis or quick insights.

🧰 4. Tools and Technologies Used

Tool/Library	Purpose
Python	Core programming language
Flask	Web framework for UI
BeautifulSoup (bs4)	Web scraping
Requests	Sending HTTP requests
Pandas	Storing and exporting data
Matplotlib / Seaborn	(Optional) Data visualization
HTML + CSS	Frontend (Jinja templating)

🧠 5. Project Description

The app scrapes all available books from `https://books.toscrape.com/`, extracting the following fields:

- **Title** 📖
- **Price** 💷
- **Rating** ⭐
- **Product Description** 📝
- **Cover Image** 🖼️

After scraping, it:

- Saves data into `books_data.xlsx`
- Displays top N books with **pagination**
- Enables **searching** by book title
- Shows book cover images (if available)
- Uses emojis to improve user experience

🔄 6. Working Mechanism

1. **Scraper Logic (`scraper.py`)**:
    - Uses BeautifulSoup to parse the website's HTML
    - Navigates multiple pages (pagination)
    - Extracts required fields for each book
    - Saves results in a Pandas DataFrame and Excel file

2. **Flask App (`app.py`)**:
    - Loads data from Excel
    - Serves the homepage with search and pagination
    - Uses `index.html` to display styled results with emojis and images

---

📂 7. Folder Structure

```
project_1_WebScraping_Flask/
├── app.py                  # Flask backend
├── scraper.py              # Scraper logic
├── books_data.xlsx         # Output data file
├── templates/
│   └── index.html          # UI template
├── requirements.txt        # Python libraries
└── README.md               # Project description
```

⚙️ 8. How to Run the Project

1. Clone the repository

```
git clone https://github.com/dhanalakshmim-eng/cantilever.git

cd cantilever/WebScraping_Ecommerce
```

2. Install required libraries

```
pip install -r requirements.txt
```

3.Run the app

```
python app.py
```

4. Visit in browser:
   `http://localhost:5000/`

📈 9. Optional Visualizations (Extension)

- Use Matplotlib/Seaborn to visualize:
  - Distribution of ratings
  - Price ranges
  - Most frequent words in book titles/descriptions

✅ 10. Key Highlights

- Practical real-world scraping practice
- Excel output for analysis
- Clean and responsive UI
- Basic pagination and search
- Fully local project (no external APIs used)

🛡️ 11. Limitations

- Site structure-dependent (changes may break scraper)
- Scraping limited to open-access pages
- No database backend (can be extended)

💡 12. Future Enhancements

- Integrate SQLite or MongoDB to store data persistently
- Add sorting filters (e.g., price, rating)
- Auto-refresh scraper to get new books weekly
- Add charts and dashboards for visual analytics

👨‍💻 13. Author

DHANA LAKSHMI M
B.E. Computer Science Student
Email: dd406652dhana@gmail.com
GitHub: https://github.com/dhanalakshmim-eng/cantilever.git