A

PROJECT REPORT ON

Ecom Data Scraping and Comparison System

SUBMITTED BY

Ms. Chinmayee Uday Daithankar

(24535)

SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

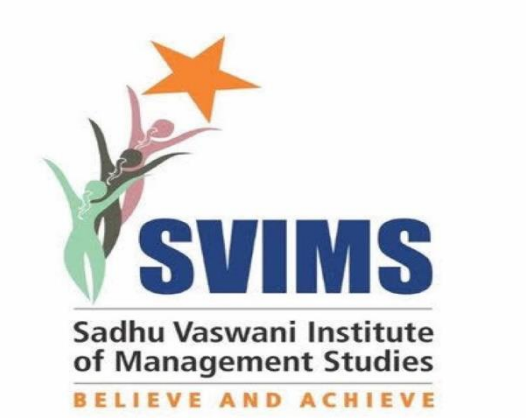
IN FULFILLMENT OF DEGREE

MASTER OF COMPUTER APPLICATION(SEM-1)

UNDER THE GUIDANCE OF

Dr . Shveti. Chandan

Through,



Sadhu Vaswani Institute of Management Studies for Girls,

Koregaon Park, Pune-411001

Certificate

Sadhu Vaswani Institute of Management Studies for Girls
Approved by AICTE (Unaided – Private)
Affiliated to Savitribai Phule Pune University
NAAC Accredited 'A' Grade
NBA Accreditation for MBA Programme (2024-2027)
ISO 9001: 2015 and ISO 14001:2015 Certified Institute

Mini Project Progress Report

Class: MCA – I Semester – II (Academic Year 2024-25)

Student Name:_____ **Project Title:**_____**Project Guide:(Institute)**_____

_____

Sr. No.	Activity to be completed	Date of Completion		Suggestions if any	Guide Sign
		Expected	Actual		
1.	**Preliminary discussion & project title finalization**	16/01/2025			
2.	**Synopsis submission & presentation**	30/01/2025			
3.	**CHAPTER 1: INTRODUCTION** 1.1 Client/Organization Profile 1.2 Need for System 1.3 Scope & Feasibility of Work 1.4 Operating Environment – H/w & S/w 1.5 Architecture of system 1.6 Detail Description of Technology Used	13/02/2025			
4.	**CHAPTER 2 : PROPOSED SYSTEM** 2.1 Proposed System 2.2 Objectives of System 2.3 User Requirements	20/02/2025			
5.	**CHAPTER 3 : ANALYSIS & DESIGN** 3.1 DFD 3.2 Table specifications (Database) 3.3 ERD	27/02/2025			
6.	3.4 Object Diagram 3.5 Class Diagram	06/03/2025			

	3.6 Use Case Diagrams 3.7 Web Site Map Diagram (if Website)				
7.	**CHAPTER 4: USER MANUAL** 4.1 User Interface Design (Screens etc.) 4.2 Limitations 4.3 Future enhancement **BIBLIOGRAPHY** **ANNEXURE:** Sample program code	20/03/2025			
8.	**Review / Presentation**	27/03/2025			
9.	**Project soft copy checkup**	17/04/2025			
10.	**Final Submission**	22/04/2025			

Note : i) Every student should report to their assigned project guide on given date of each phase as state above & take signature of project guide on this progress report card.

ii) Every time student shall bring this report card and at the time of final project submission submit this card to the project guide.

Project Coordinator

HOD (MCA)

DECLARATION BY STUDENT

To,

The Director,

SVIMS, Koregaon-Park, Pune

I , undersigned hereby declare that this project titled, “Ecom Data Scraping and Comparison System ” written and submitted by me to SPPU, Pune , in partial fulfilment of the requirement of the award of the degree of MASTER OF COMPUTER APPLICATION(MCA-1) under the guidance of , Dr . Shveti. Chandan, is my original work.

I further declare that to the best of my knowledge and belief, this project has not been submitted to this or any University or Institution for the award of any degree.

Place: Pune

Date: (Chinmayee Uday Daithankar)

ACKNOWLEDGEMENT

I extend my sincere gratitude to Dr. B. H. Nanwani , Dr. Neeta Raskar , and Dr. Shveti. Chandan for allowing me to carry out the study and for their constant encouragement, valuable, suggestion, and guidance during the research work.

I extend my special thanks to my parents and friends for supporting me for the research and inspiration.

I extend my special thanks to the guidance I received from the websites, videos, and books improve my knowledge to over come the challenges while developing the project.

Place: Pune

Date: (Chinmayee Uday Daithankar)

CHAPTER 1 : INTRODUCTION

1.1 Client / Organization Profile

Name : Syntax Tech Company Pvt

Location : Pune

About

Syntax Tech Company has achieved considerable growth over the past few years, expanding its product offerings and client base. Syntax Tech Solutions is an emerging tech company specializing in scalable e-commerce solutions. Their mission is to build smart, fast, and automated tools to enhance online shopping experiences for both consumers and sellers. They heavily rely on dynamic data handling, automation testing, and real-time comparisons to stay ahead in a competitive market

1.2 Need of the System

 Automated Data Collection

- Scrape product data (e.g., name, price, rating, stock status) from multiple e-commerce platforms.
- Handle dynamically loaded content (e.g., JavaScript-rendered elements).
- Schedule periodic runs (e.g., hourly/daily).

 Data Storage & Structure

- Store data in a structured format (e.g., PostgreSQL or MongoDB).
- Ensure timestamping and version tracking for comparison.

 Automated Data Comparison

- Compare data across platforms to highlight:
    - Price changes
    - Availability differences
    - Product name or description mismatches

 Testing & Alerts

- Create test cases to validate:
    - Correct data extraction
    - Accurate comparisons
    - Detection of anomalies

1.3 Scope and Feasibility of Work

The scope of this automation testing project includes the end-to-end development and deployment of a system capable of collecting, comparing, and validating dynamic e-commerce data. The system will focus on:

1. Product Data Collection
    - Collecting real-time data from major e-commerce platforms (e.g., Amazon, Flipkart).
    - Capturing key attributes like price, availability, ratings, reviews, discounts, etc.
    - Managing dynamic content through JavaScript rendering.
2. Data Storage & Versioning
    - Designing a database to store historical and real-time data snapshots.
    - Enabling version control to track changes in data over time.
3. Automated Comparison Mechanism
    - Building comparison logic to detect differences in pricing, stock, and metadata.
    - Providing insights through logs or dashboards.
4. Automation Testing
    - Writing test scripts to validate:
        - Accuracy of collected data
        - Functionality of the comparison engine
        - Performance and exception handling
    - Integrating CI/CD pipelines for continuous testing.
5. Reporting & Alert System
    - Generating reports for test results and data differences.

1.4 Operating Environment - H/w & S/w

Client Side System Specification

Device	**Specification**
Laptop / Desktop	• Minimum AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz • Minimum 8.00 GB (5.85 GB usable) • Minimum Windows 8 and above

Software Specification

Particular	Specification
Operating System	• **Windows 7 and above** • **Linux 6.1 or above** • **Mac OS 10.1 or above**
Browser	• **Google Chrome** • **Microsoft Edge** • **Higher Internet Explorer**

Developer Side System Specification

Device	Specification
Laptop / Desktop	• **Minimum AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz** • **Minimum 8.00 GB (5.85 GB usable)** • **Minimum Windows 11**

Software Specification

Particular	Specification
Django Framework	**Django-admin -version (5.1.2)**
Visual Studio	**Vs Studio Frame Work 1.86**
Excel	**Microsoft Excel**
Browser	**Google Chrome**

1.5 Architecture Components & Workflow

1. Data Scraping (Selenium)

- Selenium scripts navigate to e-commerce product pages (e.g., Amazon, Flipkart).
- Extract:
  - Product name, price, ratings, availability, discounts, etc.
  - Use XPath, CSS selectors, or JavaScript DOM execution for dynamic content.
- Data is collected into a Python dictionary or directly into a Pandas DataFrame.

2. Data Processing & Comparison (Pandas + NumPy)

- Raw data is converted into DataFrames using Pandas.

- NumPy is used for efficient mathematical operations or array comparisons.
- Previous data (if stored in Excel or DB) is loaded and compared to new scraped data.
- Key comparisons:
    - Price changes
    - Out-of-stock alerts
    - New or missing items
- Matching rows are flagged, and changes are recorded**.**

3. Data Export (Excel Reports)

- Comparison results are exported to an Excel file (e.g., daily_comparison_2025-04-21.xlsx)
- Sheets can include:
    - All data
    - Changes only
    - Summary
- Excel is useful for non-technical users and business teams.

4. Automation Testing (via Django Views or PyTest)

- Create automated tests for:
    - Scraping functionality (e.g., checking valid response structure)
    - Data integrity (nulls, type mismatches)
    - Logical comparison tests (e.g., if price dropped, flag it)
- These can run via CLI or as Django commands (python manage.py runscript test_data_collection)

5. Web Interface (Django Dashboard – Optional)

- Build a simple dashboard to:
    - Upload new Excel files for comparison
    - Show historical trends (using Matplotlib or Plotly)
    - Trigger scraping from UI
    - Download reports

1.6 Detail Description of Technology Used

Django Frame-Work

Django is a high-level, open-source web framework for building robust and scalable web applications. It is written in Python and follows the Model-View-Template (MVT) architectural pattern, which is similar to the popular Model-View-Controller (MVC) pattern. Django was created to make web development easier and faster by providing a clean, pragmatic design and a

set of tools that simplify common web development tasks. Django is maintained by the Django Software Foundation (DSF), and its primary goal is to reduce the amount of code developers need to write while following best practices for security and scalability. Advantages Ease of Use and Rapid Development, Security Features , Built-in Admin Interface , Robust ORM (Object-Relational Mapping) , Clean and Readable Code HTML

(Hyper Text Markup Language)

HTML (Hyper Text Markup Language) is the standard markup language used to create and structure content on the web. It defines the structure of web pages using a system of elements (also called tags) that describe different parts of the content, such as headings, paragraphs, images, links, tables, and more. HTML is essential for building any web page, as it provides the foundational structure that browsers can interpret and display. HTML documents are composed of various elements enclosed in angle brackets (< >) like , ,
,
, etc. HTML is often combined with other technologies like CSS (for styling) and JavaScript (for interactivity) to create modern web applications.
Advantage
Foundation of Web Development , Ease of Learning and Use Compatibility with All Browser, Platform Independence, Rich Media Integration, Responsiveness and Mobile-Friendliness, Accessibility ,Integration with Other Technologies.

Python

Python is a high-level, interpreted programming language that is known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and enables developers to write clean, maintainable code with fewer lines. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. 11 Python's syntax is designed to be intuitive and straightforward, making it an excellent choice for beginners and experienced programmers alike. It has an extensive standard library and a large ecosystem of third-party packages, which allows it to be used in a wide range of fields, from web development to data science and

automation. Advantages Easy to Learn and Use, Versatility and Wide Range of Applications, Large and Active Community , Cross-Platform Development , Rich Ecosystem of Libraries and Frameworks , Rich Ecosystem of Libraries and Frameworks and High Productivity and Readability

Selenium
Selenium is a powerful and widely-used open-source framework for automating web browser testing. It allows developers and testers to validate web applications across different browsers and platforms using various programming languages such as Java, C#, Python, and more[1][2]. Selenium is particularly popular for its ability to perform cross-browser testing and is considered one of the most reliable tools for web application automation[3]

Pandas
Pandas is a powerful and versatile open-source Python library designed for data manipulation and analysis. It provides data structures and functions needed to work efficiently with structured data, such as spreadsheets or SQL tables[1][2]. The name "Pandas" is derived from "Panel Data" and "Python Data Analysis"

- **Data Cleaning**: Pandas can handle missing data (represented as NaN) and allows for easy removal or filling of these values[2].
- **Data Transformation**: It provides powerful tools for merging, joining, and reshaping datasets[2].
- **Data Analysis**: Pandas can perform various statistical operations, such as calculating mean, median, and correlation between columns[1].
- **Data Visualization**: It integrates well with libraries like Matplotlib for plotting and visualizing data[2].

CHAPTER 2

Proposed System

The proposed system is an intelligent, automated data collection and comparison tool designed for **Syntax Tech Solutions**. It aims to automate the tracking of product details from various e-commerce platforms, compare them

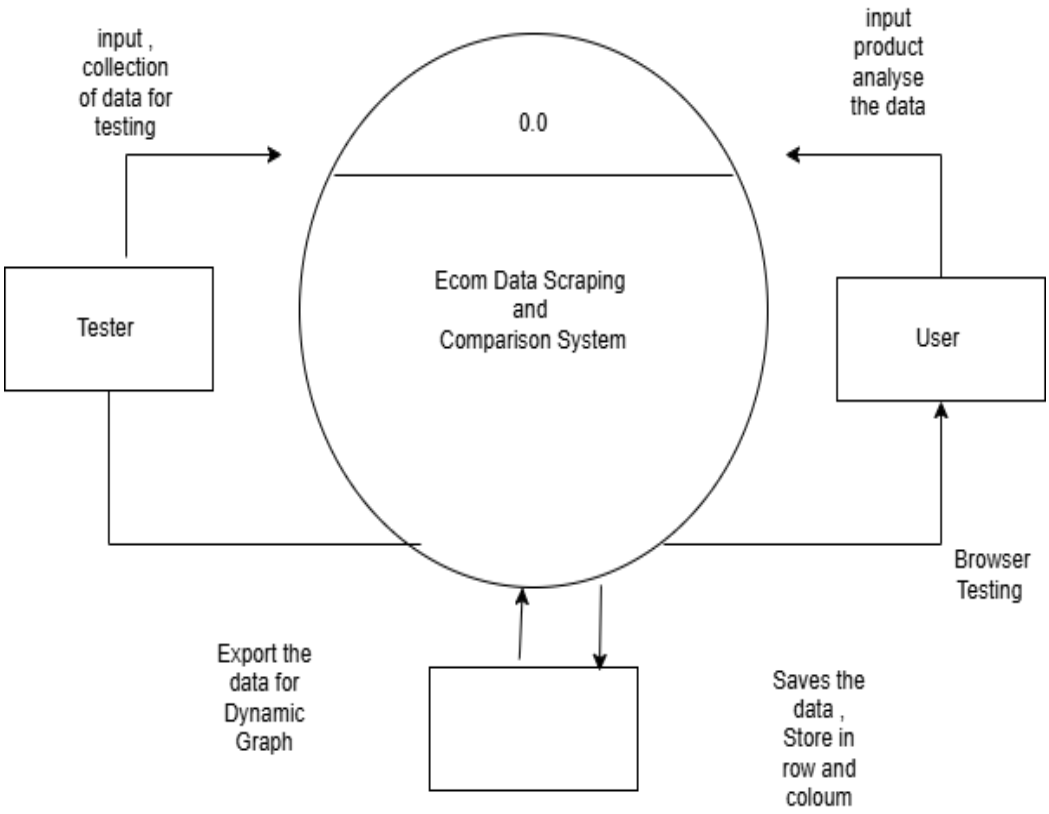
over time or across sites, and generate actionable insights in real-time. The system will use **Selenium for scraping**, **Pandas and NumPy for processing**, **Excel for reporting**, and **Django** for integration, orchestration, and optional UI interaction.

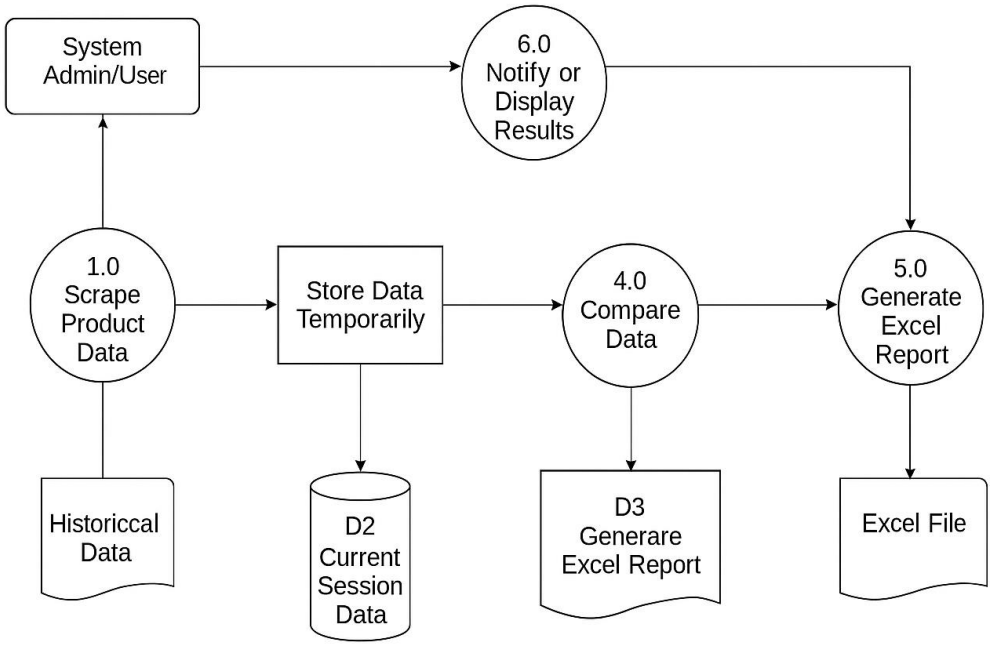
Objectives of the Proposed System

- **Automate product data collection** from dynamic e-commerce sites.
- **Compare product details** (price, availability, ratings) across multiple platforms or over time.
- **Generate Excel reports** with detailed insights and highlights of changes.
- **Enable testing and validation** of data accuracy and scraper reliability.
- **Provide a simple web interface (optional)** to manage tasks, view results, and download reports.

CHAPTER 3

ANALYSIS & DESIGN

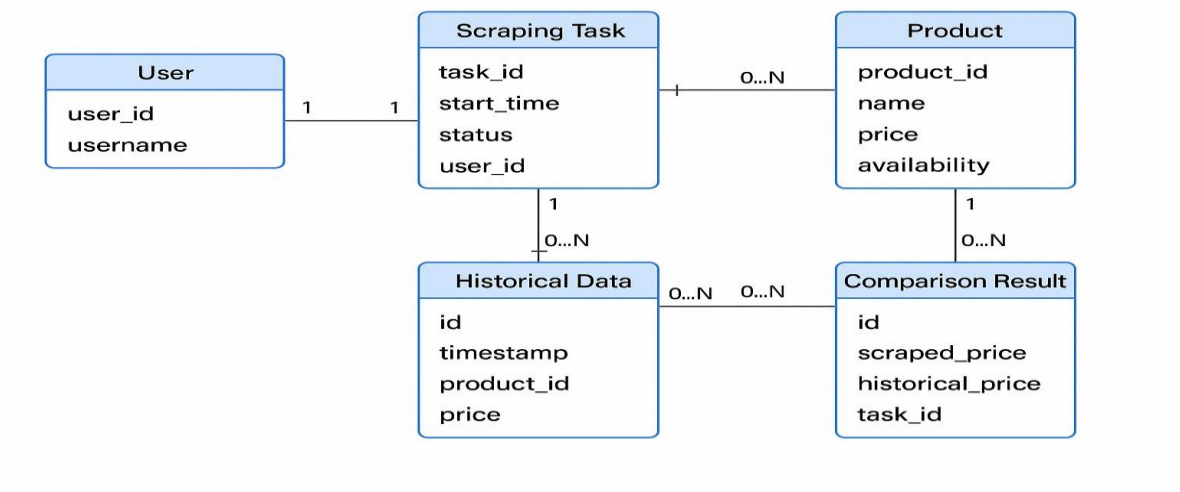




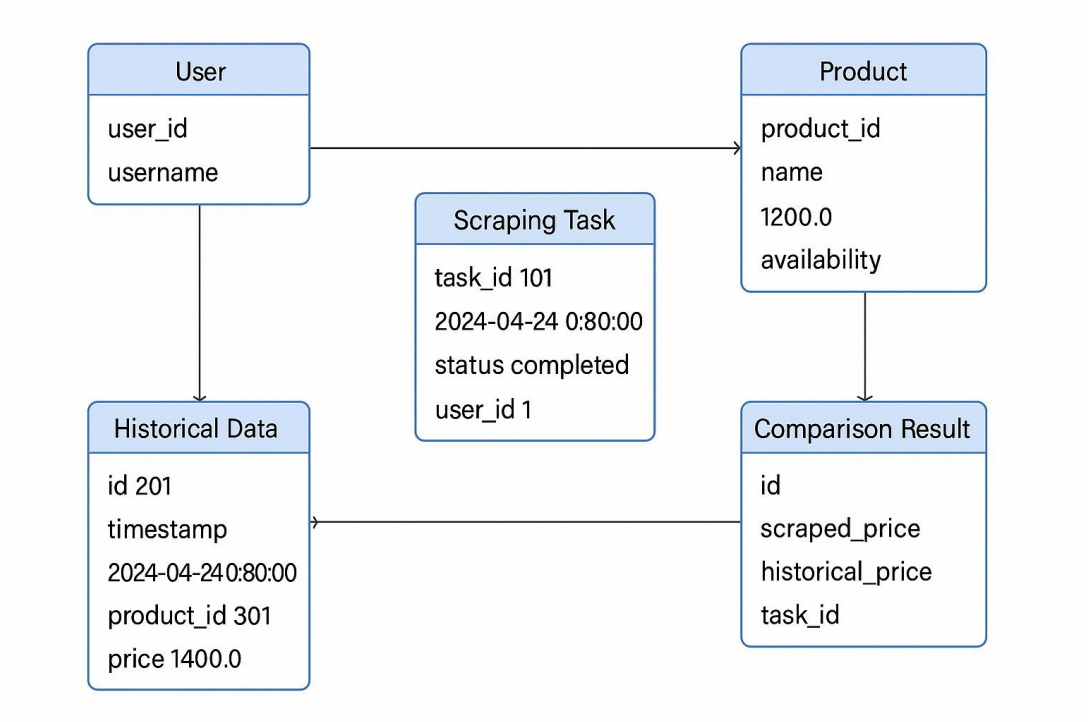
3.2 Table Specification

Name of Product	Price of The Product
Title name of Product	Price

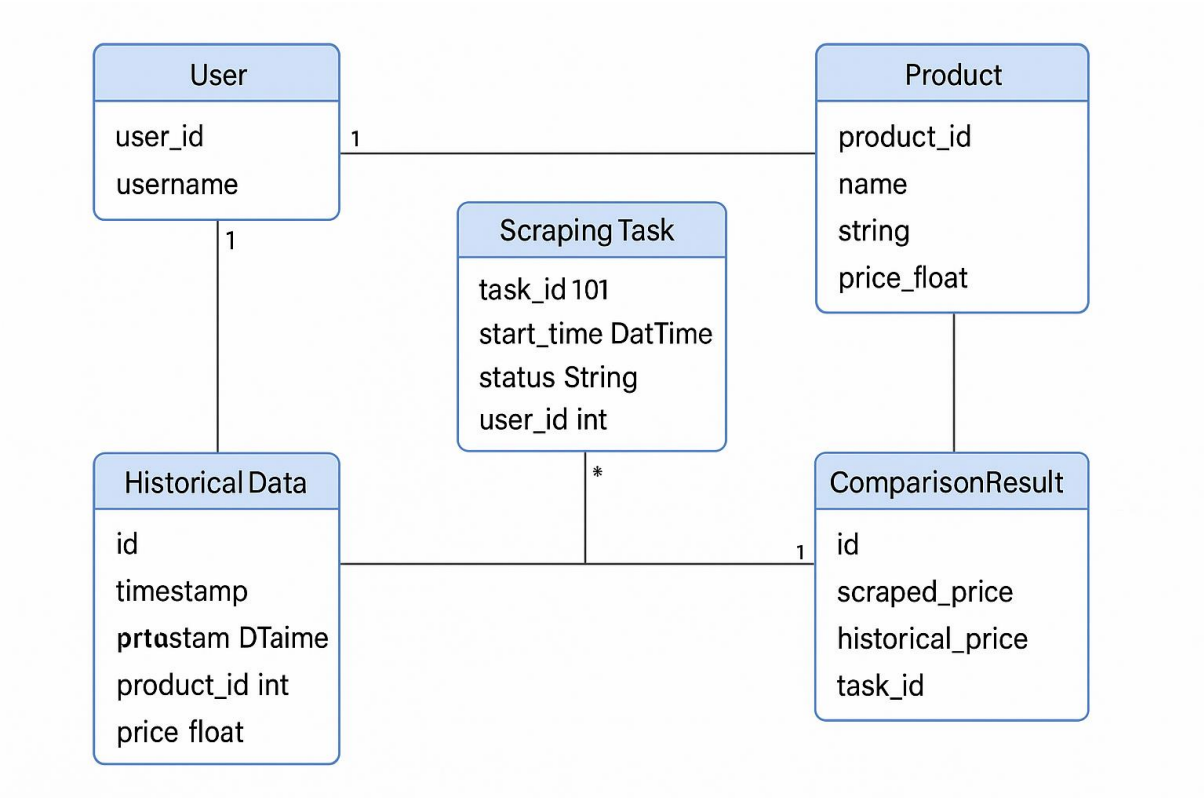
3.3 ERD (Entity Relationship Diagram)



3.4 Object Diagram

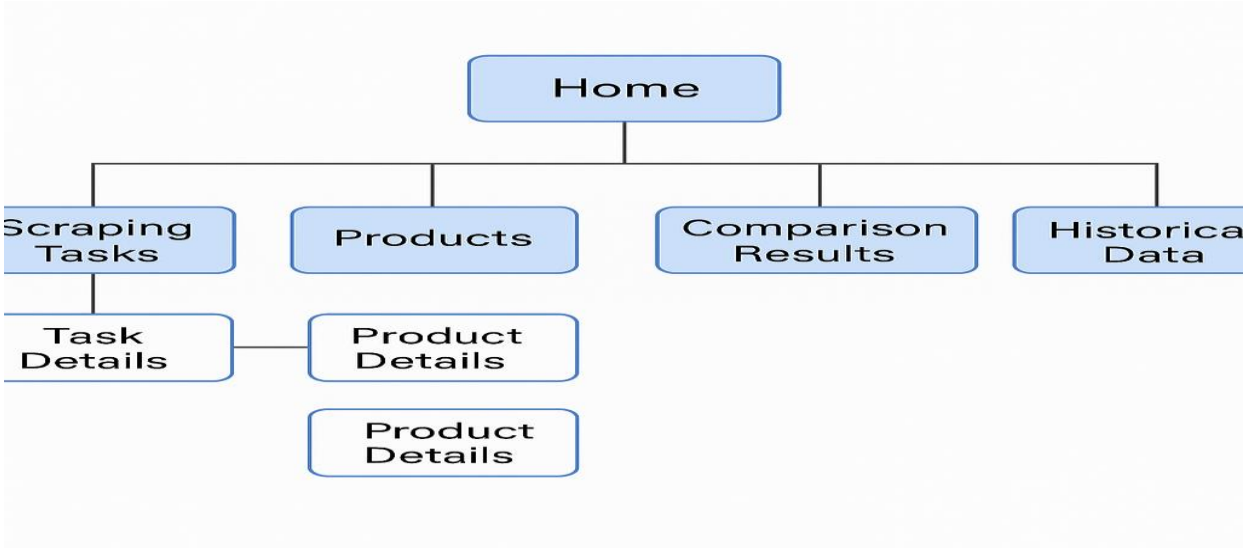


3.6 Class Diagram



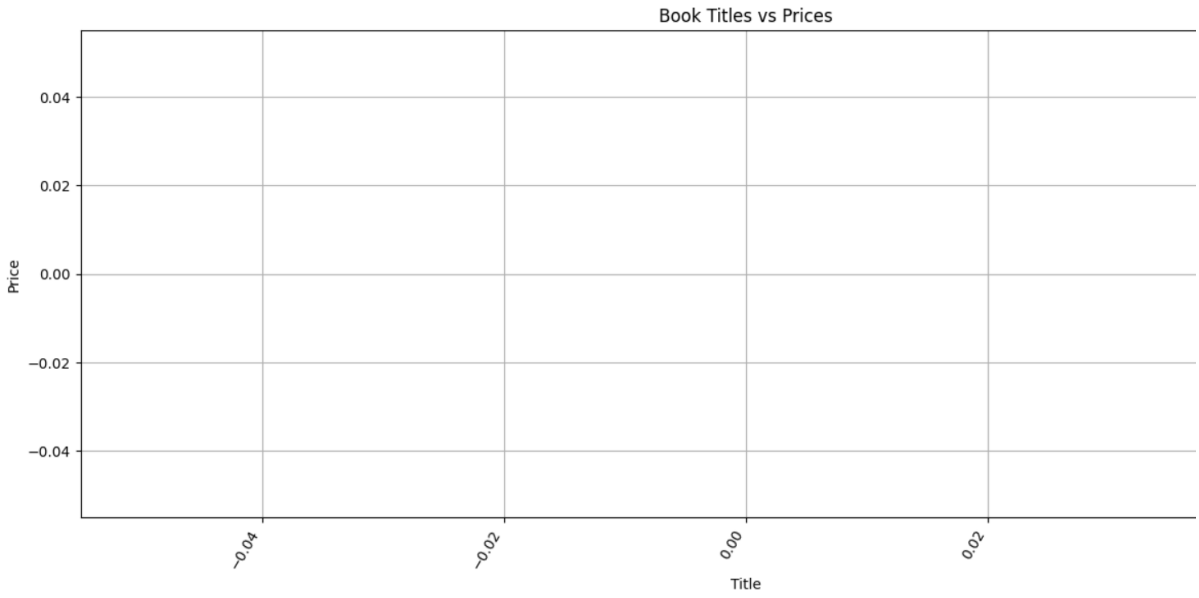
Use Case Diagram

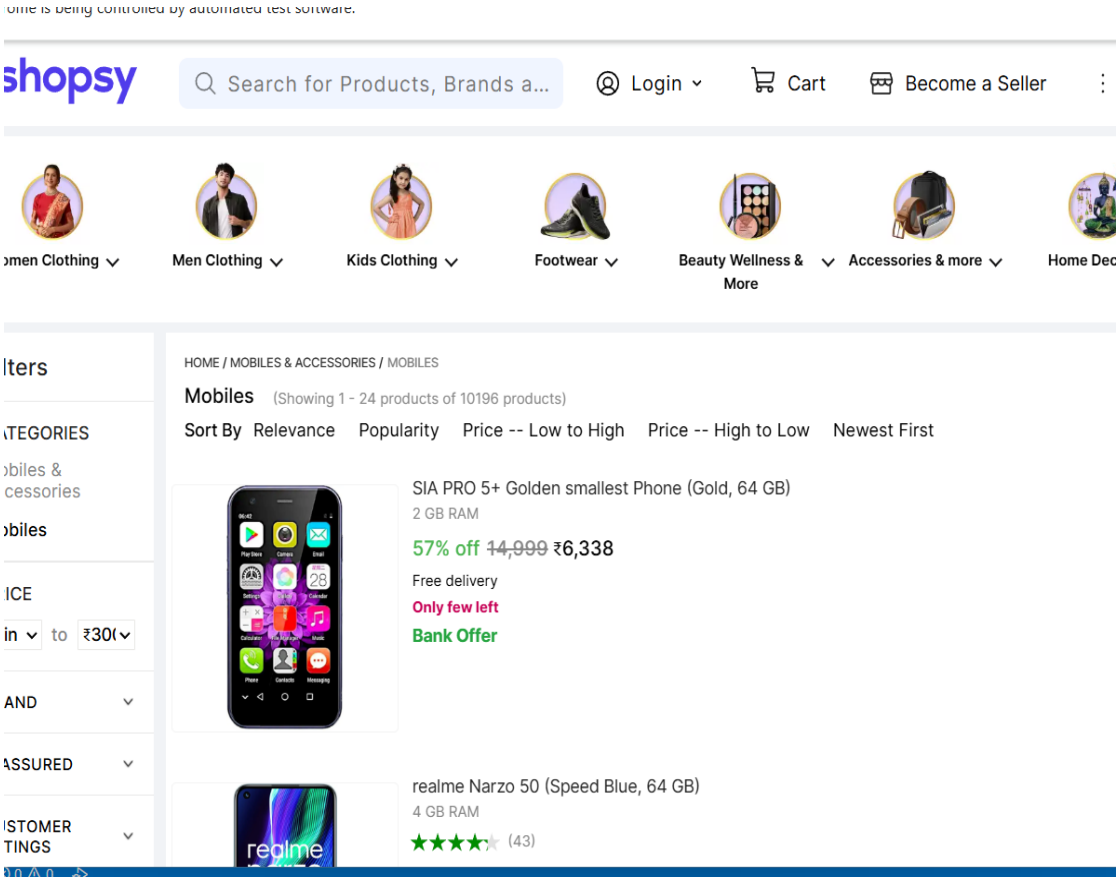
3.8 Web site Map Diagram

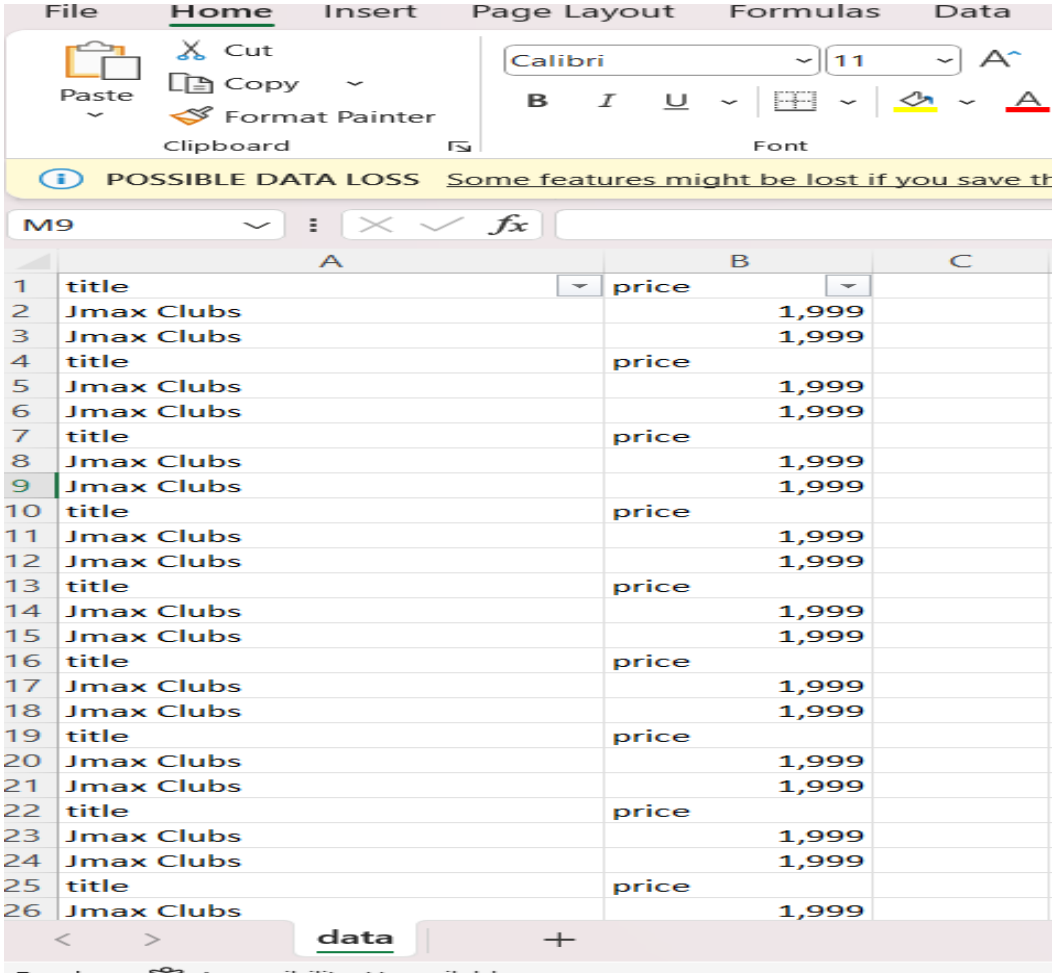


Chapter 4 User Manul

Book Titles vs Prices







Limitations

Limitations of the Project

While the **E-commerce Data Collection and Comparison Automation System** provides robust features and automation, it also comes with a few limitations due to technical, operational, and environmental factors:

---

1. Dynamic Website Structure

- E-commerce platforms frequently update their HTML/CSS/JavaScript structure.

- Any change in DOM structure may break the Selenium-based scraping scripts.
- Requires constant maintenance to keep scrapers up-to-date.

---

2. Website Anti-Bot Measures

- Sites like Amazon and Flipkart have anti-scraping protections (e.g., CAPTCHAs, rate-limiting, bot detection).
- Excessive or fast scraping may lead to IP blocking or access denial.
- Requires delays, headers rotation, or proxy servers to bypass.

---

3. Scalability Constraints

- The system may struggle with scalability if scraping and comparison are extended to hundreds/thousands of products or multiple platforms without optimization.
- Excel-based reporting is not ideal for handling large datasets.

---

4. Limited Real-Time Functionality

- Since the system relies on scheduled scraping, the data may not always be real-time.
- Delays between scraping sessions can lead to missed short-term changes like flash sales.

---

5. No NLP or Sentiment Analysis

- The current system does not analyze user reviews, sentiment, or unstructured feedback.
- Focus is mainly on structured data like price, ratings, and stock status.

---

6. Dependency on Third-Party Tools

- Relies heavily on open-source tools like Selenium, Pandas, NumPy, and Excel libraries.
- Any updates or deprecations in these libraries may affect performance or compatibility.

---

7. Manual Report Interpretation

- While the Excel report is business-friendly, it still requires manual interpretation unless integrated with a dashboard or visual tool.

---

8. Error Handling Complexity

- Handling dynamic loading issues, timeouts, or partial data collection needs sophisticated error logging and retry mechanisms.

Future Enhancement

1. Integration with a Database (PostgreSQL/MongoDB)

- Replace or supplement Excel files with a proper relational or NoSQL database.
- Enable advanced querying, historical tracking, and faster comparisons.

---

2. Dashboard and Data Visualization

- Develop a **real-time Django dashboard** using tools like Chart.js or Plotly.
- Visualize trends in price changes, stock levels, or competitor movement over time.
- Provide filterable, downloadable reports directly from the UI.

---

3. Proxy & CAPTCHA Handling

- Integrate proxy rotation, headless browsers (like Playwright), or services like 2Captcha to handle:

- Rate-limiting issues
- IP bans
- CAPTCHA challenges

---

4. Real-Time Scraping and Notification System

- Set up **event-based triggers** (like webhook notifications or price thresholds) to get real-time alerts via:
  - Email
  - SMS
  - Slack/Telegram

---

5. Product Comparison Across Multiple Platforms

- Automatically match products between Amazon, Flipkart, etc., and compare:
  - Price differences
  - Shipping details
  - Delivery times
  - Availability

---

6. Sentiment Analysis on Reviews (NLP Integration)

- Use Natural Language Processing (NLP) libraries like **NLTK** or **spaCy** to:
  - Analyze customer reviews
  - Extract user sentiment (positive/negative/neutral)
  - Include review trends in reports

---

7. Mobile App Integration

- Develop a lightweight Android/iOS app for business users to:

- Monitor trends
- Get instant alerts
- View mini-dashboards

---

8. Cloud Deployment & Scalability

- Deploy the entire system to the cloud (e.g., AWS, Heroku, or Azure).
- Use task queues like **Celery + Redis** for background scraping and processing.
- Enable horizontal scaling to support more sites or frequent scraping intervals.

---

9. AI-Powered Price Prediction

- Use historical data to train a basic ML model to **predict future price drops** or stock shortages.
- Useful for making business decisions or preemptive actions.

ANNEXURE: Sample program code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from webdriver_manager.chrome import ChromeDriverManager
import time
```

```
# Set up Chrome options (enable headless mode if needed)
options = Options()
# Uncomment the line below to run in headless mode
# options.headless = True

# Set up the Chrome WebDriver with the appropriate driver manager
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
q1="Mobile"
i= 1
# Navigate to the desired URL
driver.get(f"https://www.shopsy.in/search?q="+q1+"&as=on&as-show=on&page=")

try:
    # Wait for the element to be visible before interacting with it (increase wait time to 30 seconds)
    elem = WebDriverWait(driver, 30).until(
        EC.visibility_of_element_located((By.CLASS_NAME, "css-175oi2r"))
    )

    # If the element is found, print the text of the element or perform other actions
    print(elem.text)

except Exception as e:
    print(f"An error occurred: {e}")
```

```
finally:
    # Wait a few seconds to ensure the page is fully loaded (adjust timing if needed)
    time.sleep(2)

    # Close the browser session gracefully
    driver.quit()
```

```
from bs4 import BeautifulSoup
import os
import pandas as pd

# Example dictionary
d = {'title': [], 'price': []}

# Directory containing HTML files
directory = "data"

# Iterate through files in the directory
for file in os.listdir(directory):
    try:
        with open(f"{directory}/{file}", encoding="utf-8") as f:
            html_doc = f.read()
```

```
        soup = BeautifulSoup(html_doc, "html.parser")


        # Fix: Find the element correctly (assuming it's a 'span' or 'div' with a
class)
        t = soup.find(attrs={"style": "max-height: 32px; line-height: 16px; -
webkit-line-clamp: 2;"})
        p = soup.find(attrs={"class":"css-146c3p1 r-1h7g6bg r-1vgyyaa r-1rsjblm
r-142tt33 r-11wrixw"})


        # If both title and price are found, append them
        if t and p:
            title = t.get_text(strip=True)
            price = p.get_text(strip=True)
            print(f"Title: {title}, Price: {price}")
            d['title'].append(title)
            d['price'].append(price)
        else:
            # If title or price is missing, append None for each
            d['title'].append(None)
            d['price'].append(None)


    except Exception as e:
        print(e)


# Create DataFrame and save to CSV
df = pd.DataFrame(data=d)
df.to_csv("data.csv", index=False)
```

Bibliography

I have been successfully completed by project through the following resources For enhancing my knowledge on.

Links :

https://www.learnvern.com/python-tutorial-django

https://docs.djangoproject.com/en/5.1/ From Books : Django Frame Works

Restful Django API

https://www.selenium.dev/documentation/

https://pandas.pydata.org/docs/