Amazon Data Visualizer

A Major Project Report Submitted To

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal

Towards Partial Fulfilment for the Award Of

Bachelor of Technology

In

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Submitted By

Aditya Rao (0863CS191009)

Anshika Patel (0863CS191023)

Aatreyee Jalodkar (0863CS191003)

Abizer Lohawala (0863CS191007)

Under the Supervision of

Asst. Prof. Pragya Ranka

Session: 2022-2023

Department of Computer Science and Engineering,

Prestige Institute of Engineering, Management and Research, Indore (M.P.)

[An Institution Approved By AICTE, New Delhi & Affiliated To RGPV, Bhopal]

PRESTIGE INSTITUTE OF ENGINEERING MANAGEMENTAND RESEARCH

INDORE (M.P.)

DECLARATION

We **Aditya Rao, Anshika Patel, Aatreyee Jalodkar, and Abizer Lohawala** hereby declare that the project entitled “**Amazon Data Visualizer**”, which is submitted by us for the partial fulfilment of the requirement for the award of Bachelor of Technology in Computer Science & Engineering to the Prestige Institute of Engineering, Management and Research, Indore (M.P.). Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, comprises my own work and due acknowledgement has been made in text to all other material used**.**

Signature of Students:

Date:

Place:

PRESTIGE INSTITUTE OF ENGINEERING MANAGEMENTAND RESEARCH

INDORE (M.P.)

DISSERTATION APPROVAL SHEET

This is to certify that the dissertation entitled **"Amazon Data Visualizer"** submitted by **Aditya Rao (0863CS191009), Anshika Patel (0863CS191023), Aatreyee Jalodkar (0863CS191003), and Abizer Lohawala (0863CS191007)** to the Prestige Institute of Engineering Management and Research, Indore (M.P.) is approved as fulfilment for the award of the degree of "Bachelor of Technology in Computer Science & Engineering" by Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, (M.P.).

Internal Examiner	**External Examiner**
Date:	Date:

HOD, CSE

Dr. Piyush Choudhary

PIEMR, INDORE

PRESTIGE INSTITUTE OF ENGINEERING MANAGEMENTAND RESEARCH

INDORE (M.P.)

CERTIFICATE

This is certified that project entitled "**Amazon Data Visualizer**" submitted by **Aditya Rao, Anshika Patel, Aatreyee Jalodkar, and Abizer Lohawala** is a satisfactory account of the bonafide work done under our supervision and is recommended towards partial fulfilment for the award of the degree Bachelor of Technology in Computer Science & Engineering to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.).

Date:

Enclosed by:

Asst. Prof. Pragya Ranka	Asst. Prof**.** Pragya Ranka	Dr. Piyush Choudhary
Project Guide	**Project Coordinator**	**Professor & Head, CSE**

Dr. Manojkumar Deshpande

Director

PIEMR, Indore

PRESTIGE INSTITUTE OF ENGINEERING MANAGEMENTAND RESEARCH

INDORE (M.P.)

ACKNOWLEDGEMENT

After the completion of Major project work, words are not enough to express my feelings about all these who helped me to reach my goal; feeling above this is my indebtedness to the almighty for providing me this moment in life.

First and foremost, I take this opportunity to express my deep regards and heartfelt gratitude to my project guide and **Project Coordinator Asst. Prof. Pragya Ranka, Department of Computer Science and Engineering, PIEMR, Indore** for their inspiring guidance and timely suggestions in carrying out my project successfully. They are also the constant source of inspiration for me. Working under their guidance has been an opportunity for me to learn more and more.

I am extremely thankful to **Dr. Piyush Choudhary, (HOD, CSE)** for his co-operation and motivation during the project. I extend my deepest gratitude to **Dr. Manojkumar Deshpande, Director, PIEMR, and Indore** for providing all the necessary facilities and true encouraging environment to bring out the best of my endeavours.

I would like to thank all the teachers of our department for providing invaluable support and motivation. I remain indebted to all the non-teaching staff of our Institute who has helped me immensely throughout the project.

I am also grateful to my friends and colleagues for their help and co-operation throughout this work. Last but not least; I thank my family for their support, patience, blessings and understanding while completing my project.

Name of Students:

Aditya Rao (0863CS191009)

Anshika Patel (0863CS191023)

Aatreyee Jalodkar (0863CS191003)

Abizer Lohawala (0863CS191007)

INDEX

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

CHAPTER 2 BACKGROUND AND RELATED WORK

CHAPTER 3 DESIGN (UML AND DATA MODELING)

CHAPTER 4 IMPLEMENTATION

CHAPTER 5 PROJECT PLAN

CHAPTER 6: Project Screenshot

CHAPTER 7 CONCLUSION/ FUTURE SCOPE

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The World Wide Web consists of an interlinked network of information, which is presented through websites to the users. World Wide Web has significantly changed the way we share, collect, and publish data.

The amount of presented information grows constantly. As data grows in amount, variety, and importance, business leaders must focus their attention on the data that matters the most. Not all data is equally important to businesses or consumers.

Also with the usage of Web as a new marketing and sales channel the quantity of content multiplied. Online merchants offer large packs of data to describe their products. Knowledge base providers offer access to their databases.

With this unorganized growth, it is no longer possible to manually track and record all available sources. That moment, is when Web Scraping evolved. Automated techniques allow the collection of a massive amount of data from the Web compared to manual data extraction.

Together with Web Scraping another term became very important – Meta Data. Massive collection of data obtained by Web Scraping allows Meta Data analysis.

- A web scraper is a tool that automates the process of web scraping, allowing users to easily extract data from websites.
- Web scrapers can be used to collect information such as product prices, customer reviews, news articles, social media data, and more.

Web scrapers work by sending requests to web servers and then parsing the HTML content of the response to extract the desired data.

Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web (WWW) and save it to a file system or database for later retrieval or analysis. Commonly, web data is scrapped utilizing Hypertext Transfer Protocol (HTTP) or through a web browser. This is accomplished either manually by a user or automatically by a bot or web crawler. Due to the fact that an enormous amount of heterogeneous data is constantly generated on the WWW, web scraping is widely acknowledged as an efficient and powerful technique for collecting big data.

The Data Visualizer that we have made shows the analysis of data and let user visualize the data they have scraped which also provides meaningful insights and helps in decision making.

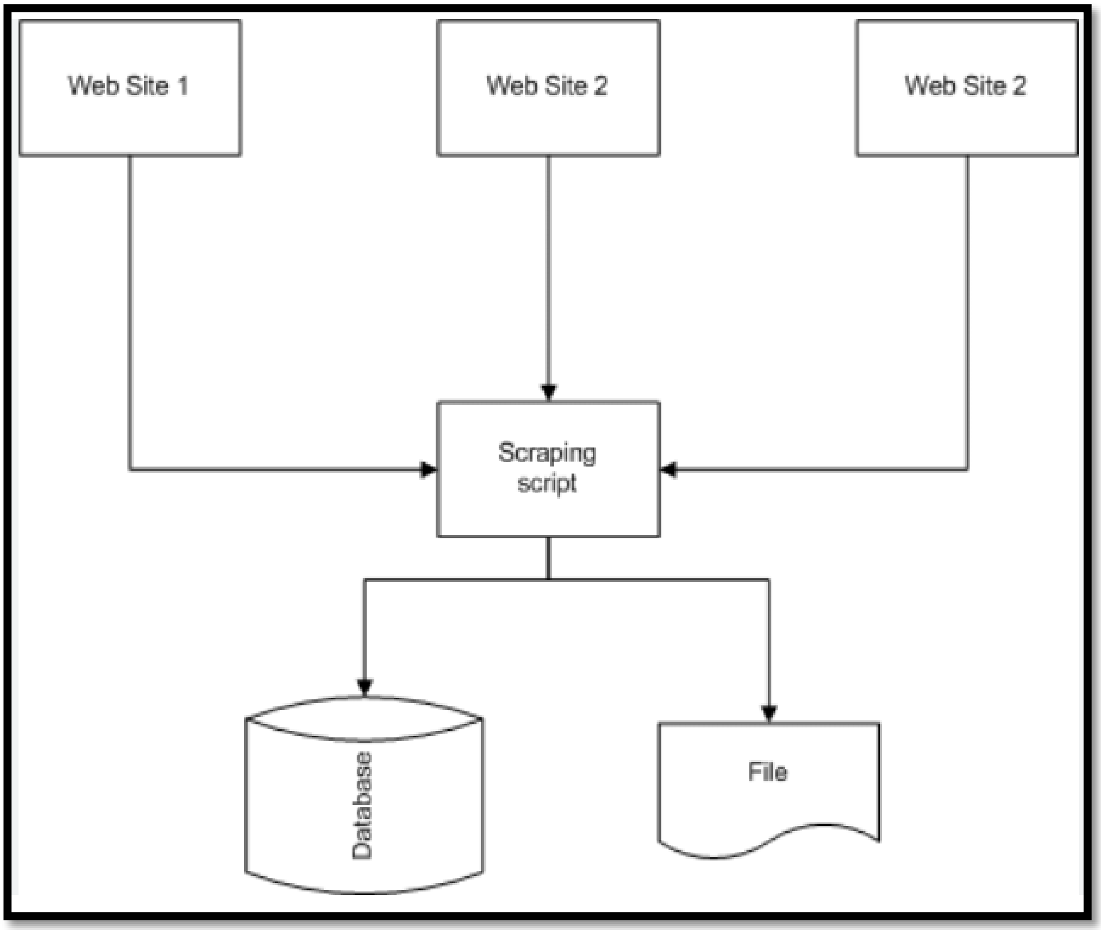


Figure 1.1 (Overview of Web Scraping System)

1.2 Motivation

Enormous amounts of source information, available on the World Wide Web, are still in the format of a Hypertext Mark-up Language (HTML) page. Automated extraction is difficult, because the intended reader was a human.

Rapid growth of the World Wide Web has significantly changed the way we share, collect, and publish data. Vast amount of information is being stored online, both in structured and unstructured forms. Regarding certain questions or research topics, this has resulted in a new problem - no longer is the concern of data scarcity and inaccessibility but, rather, one of overcoming the tangled masses of online data.

These utilizations are often only possible because the existence of automated Web Scraping. Without these techniques, it would be impossible to collect the amount of data repeatedly and in reasonable time.

The already available tools and websites in market only collects the data and allows user to download in .csv formats, this project will allow user to see data on the basis of different attributes on a single platform and also visualize it. The visualization will help them to gain insights from the data.

1.3 Objective

To create a web based application through which the user can scrape the data from amazon for whichever category they want to, download and visualize it simultaneously.

The user will also be able to select different attributes in a particular category while downloading or visualizing the data.

1.4 Analysis

1.4.1 Functional Requirements

asgiref==3.4.1
beautifulsoup4==4.9.3
bs4==0.0.1
certifi==2021.5.30
charset-normalizer==2.0.4
defusedxml==0.7.1
diff-match-patch==20200713
Django==3.2.6
django-import-export==2.5.0
django-rest-framework==0.1.0
djangorestframework==3.12.4
et-xmlfile==1.1.0
idna==3.2
MarkupPy==1.14
numpy==1.21.1
odfpy==1.4.1
openpyxl==3.0.7
pandas==1.3.1
python-dateutil==2.8.2
pytz==2021.1
PyYAML==5.4.1
requests==2.26.0
simple-chalk==0.1.0
six==1.16.0
soupsieve==2.2.1
sqlparse==0.4.1
tablib==3.0.0
urllib3==1.26.6
xlrd==2.0.1
xlwt==1.3.0

Table 1.1 (Functional Requirements)

1.4.2 Non - Functional Requirements

Requirement ID	**Statement**	**Must/Want**
NFR01	Products should be searched from Amazon Website.	Must
NFR02	Amazon specified keywords should be selected under categories section.	Must
NFR03	Microsoft Edge Browser required	Want

Table 1.2 (Non-Functional Requirements)

1.4.3 Use Case Diagram

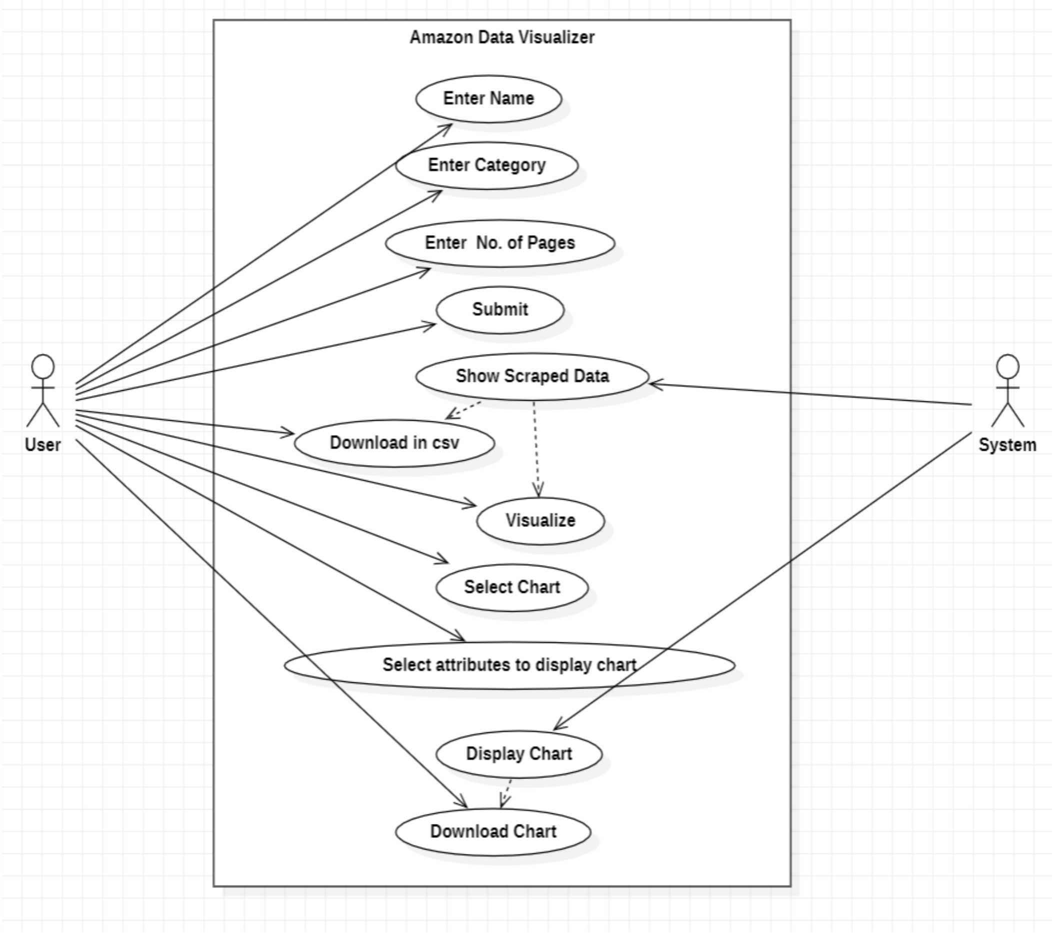


Figure 1.1 (Use-Case Diagram)

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Problem Statement

As we see how e-commerce websites plays an integral role when it comes to shopping.

Based on our prior observations we make the following hypothesizes:

People do not get a platform or tool to make observations and get insights about different things available online and tend to get results that are not as suitable by just scrolling over internet.

2.2 Background and Related Work

2.2.1 Earlier Literature Surveys

Authors Suganya, in their paper, use web scraping for web citation analysis which helps researchers in finding related papers for further analysis. They study and compare three methods: Particle Swarm Optimization, Hidden Markov Model algorithm, and Firefly Optimization algorithm based Web scraping to extract information regarding web citation based on the given query. Based on their experiments, it is found that Firefly Optimization Algorithm-based web scraping (FOAWS) performs better than the rest of the techniques.

Similarly, authors Rahmatulloh in their paper employ HTML DOM-based web scraping to make recapitulations of scientific article publications from Google Scholar to aid in research studies. The recapitulations are further programmed to be presented as a report either in a PDF or Excel file.

The authors Kolli show a customized news Internet search engine that focuses on constructing a repository of reporting stories by relating adept content data mining from a network information sheet from shifted e-information entrances.

In Li, the author proposes employing web scraping and natural language processing to decrease the time required to detect the research gap. This strategy is tested by looking at three different areas: safety awareness, home prices, sentiment, and artificial intelligence. First, the titles of the publications are scraped from Google Scholar and using tokenization. The titles are parsed. By ranking the collocations based on descending range of frequency, the set of keywords that are not used in the paper title is obtained, and the research void is determined.

In Breno, the paper proposes a scholarly production dataset focusing on COVID-19 to provide an overview of scientific research activities, making it easier to identify countries, scientists, and research groups most active in this corona virus disease task force. Between January 2019 and July 2020, a dataset containing 40,212 records of article metadata was extracted from various databases, namely Scopus, PubMed, arXiv, and bioRxiv using Python Web Scraping techniques and pre-processed with Pandas Data Wrangling using a pipeline versioned with the Data Version Control tool (DVC), making it easy to replicate and audit. To extract data from PubMed and Scopus, API was used, and Scrapy was used for scraping data from arXiv and bioRxiv databases.

2.3 Solution Approach (*methodology and technology used)*

Here, we have made a platform that will make it easier for people to collect data and scrape it as per their requirement.

People will also be able to visualize the data and then analyze it to make meaningful insights.

2.4 Methodologies used

The methods of Web Scraping evolved together with the World Wide Web. Not all listed methods were available at the beginning. There are two examples to mention, because these are presently the most used techniques.

Since 2000 the Document Object Model (DOM) became more popular in DHTML. A broader acceptance later on allowed the HTML Parsing technique to evolve to DOM Parsing.

Second method is Application Programming Interfaces (APIs). This technique is the youngest on the list, the growth of available content APIs is dated from 2005. According to ProgrammableWeb.com the number of APIs has grown within 8 years from 0 to 10302.

2.4.1 Manual Scraping

Manual scraping is still an option in specific situations. These situations are:

- When the amount of data is minimal,
- When the data being scraped does not require a repetitive task,
- When setting up automated scraping would take longer than the data collection itself.
- Possibly security measures or specific characteristics of the website do not allow automated methods.

2.4.2 HTML Parsing

Sites don’t always provide their data in comfortable formats such as .csv or .json files. HTML Pages are created by the server as a response to a user’s request. At this point server software is not relevant, rather the output in the browser is important.

Analysis of the HTML structure in the provided web page will show repeated elements. With a programming language script or Web Scraping tool, each page with similar pattern can be used as a source for data.

2.4.3 DOM Parsing

Document Object Model (DOM) Parsing is an evolution of HTML Parsing based on developments of the language and browsers which lead to the introduction of the Document Object Model.

DOM is heavily used for Cascading Style sheets (CSS) and JavaScript. Integration of DOM revealed new possibilities for addressing some specific parts of the webpage. These are used in Web Scraping for easier navigation through webpage content.

2.4.4 XPath

Similar addressing possibility as DOM provides XPath (XML Path Language). The name suggests a usage for XML documents. It is applicable also to HTML format. XPath requires a more precisely structured webpage than DOM and has the same possibility to address segments within the webpage. Figure 5 shows the document structure as interpreted in XPath.

2.4.5 APIs

Whilst the previous methods work to scrape human-readable outputs.

Application Programming Interface(API) expects an application as a communication partner. Thus APIs are often named as machine-readable interfaces (versus human-readable). Even APIs were introduced much later than the WWW, and their growth was very fast. The world of APIs is fragmented. For a simple overview and orientation were API Directories created. Most of the available APIs are registered and described in the directory with relevant links to the sources.

API Directories also provide their own API, which allows users to search in their database for API Sources. A standard HTTP Request sent to an API Endpoint returns an answer from server. Each API has its own specification and options. The format of the answer can be set as option in the request.

The most widely used format for API communication is JSON.

2.5 Error-detection and correction

Up until the grouping each character has been treated separately, and the context in which each character appears has usually not been exploited. However, in advanced optical text recognition problems, a system consisting only of single-character recognition will not be sufficient. Even the best recognition systems will not give 100% percent correct identification of all characters, but some of these errors may be detected or even corrected by the use of context.

There are two main approaches, where the first utilizes the possibility of sequences of characters appearing together. This may be done by the use of rules defining the syntax of the word, by saying for instance that after a period there should usually be a capital letter. Also, for different languages the probabilities of two or more characters appearing together in a sequence can be computed and may be utilized to detect errors. For instance, in the English language the probability of a “k” appearing after an “h” in a word is zero, and if such a combination is detected an error is assumed. Another approach is the use of dictionaries, which has proven to be the most efficient method for error detection and correction. Given a word, in which an error may be present, the word is looked up in the dictionary. If the word is not in the dictionary, an error has been detected, and may be corrected by changing the word into the most similar word. Probabilities obtained from the classification, may help to identify the character which has been erroneously classified.

If the word is present in the dictionary, this does unfortunately not prove that no error occurred. An error may have transformed the word from one legal word to another, and such errors are undetectable by this procedure. The disadvantage of the dictionary methods is that the searches and comparisons implied are time-consuming.

CHAPTER 3

DESIGN (UML AND DATA MODELING)

3.1 Modules Specifications

```
asgiref==3.4.1
beautifulsoup4==4.9.3
bs4==0.0.1
certifi==2021.5.30
charset-normalizer==2.0.4
defusedxml==0.7.1
diff-match-patch==20200713
Django==3.2.6
django-import-export==2.5.0
django-rest-framework==0.1.0
djangorestframework==3.12.4
et-xmlfile==1.1.0
idna==3.2
MarkupPy==1.14
numpy==1.21.1
odfpy==1.4.1
openpyxl==3.0.7
pandas==1.3.1
python-dateutil==2.8.2
pytz==2021.1
PyYAML==5.4.1
requests==2.26.0
simple-chalk==0.1.0
six==1.16.0
soupsieve==2.2.1
sqlparse==0.4.1
tablib==3.0.0
urllib3==1.26.6
xlrd==2.0.1
xlwt==1.3.0
```

3.2 UML Modelling

UML stands for Unified Modelling Language. Taking SRS document of analysis as input to the design phase drawn UML diagrams. The UML is only language so is just one part of the software development method. The UML is process independent, although optimally it should be used in a process that should be driven, architecture-centric, iterative, and incremental. The UML is language for visualizing, specifying, constructing, documenting the articles in a software intensive system.

A modelling language is a language whose vocabulary and rules focus on the conceptual and physical representation of the system. A modelling language such as UML is thus a standard language for software blueprints. The UML is a graphical language, which consists of all interesting systems. There are also different structures that can transcend what can be represented in a programming language. These are different diagrams in UML.

Use Case Diagram

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behaviour of the system as seen from the actor’s point of view. It describes the function provided by the system as a set of events that yield a visible result for the actor.

Use case Scenario:

Use case name	Amazon Data Visualizer
Participating actors	User, System
Flow of events	Enter required attributes (u) Submit (u) Show scraped data (s) Download in .csv (u) To Visualize (u) Select attributes of chart(u) Displays Chart(s) Download Chart(u)
Entry condition	-
Exit condition	-
Quality Requirements	High internet speed

Table 3.1 (Use case scenario for Amazon Data Visualizer)

3.2.1 Class Diagram

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagram describe the different perspective when designing a system-conceptual, specification and implementation.

Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

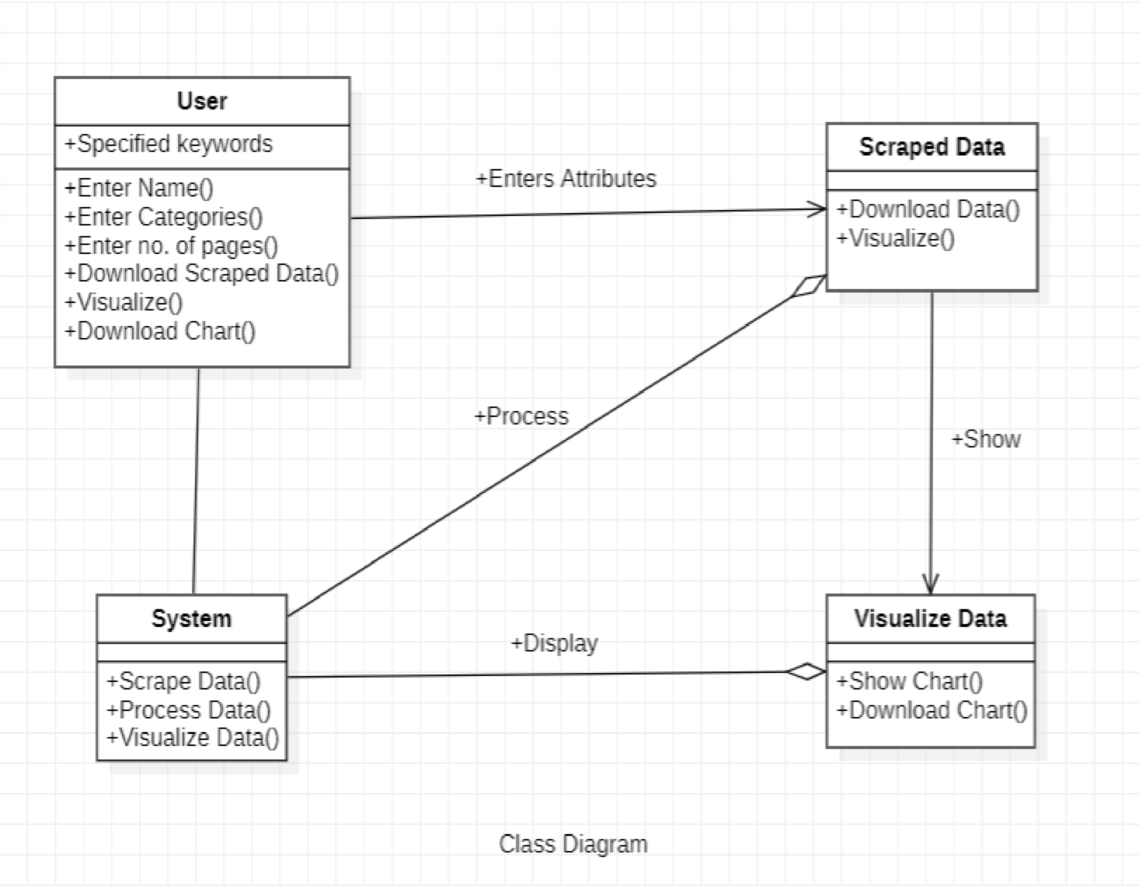


Figure 3.1 (Class Diagram)

3.2.2 Sequence Diagram

Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

Objects: Object can be viewed as an entity at a particular point in time with specific value and as a holder of identity.

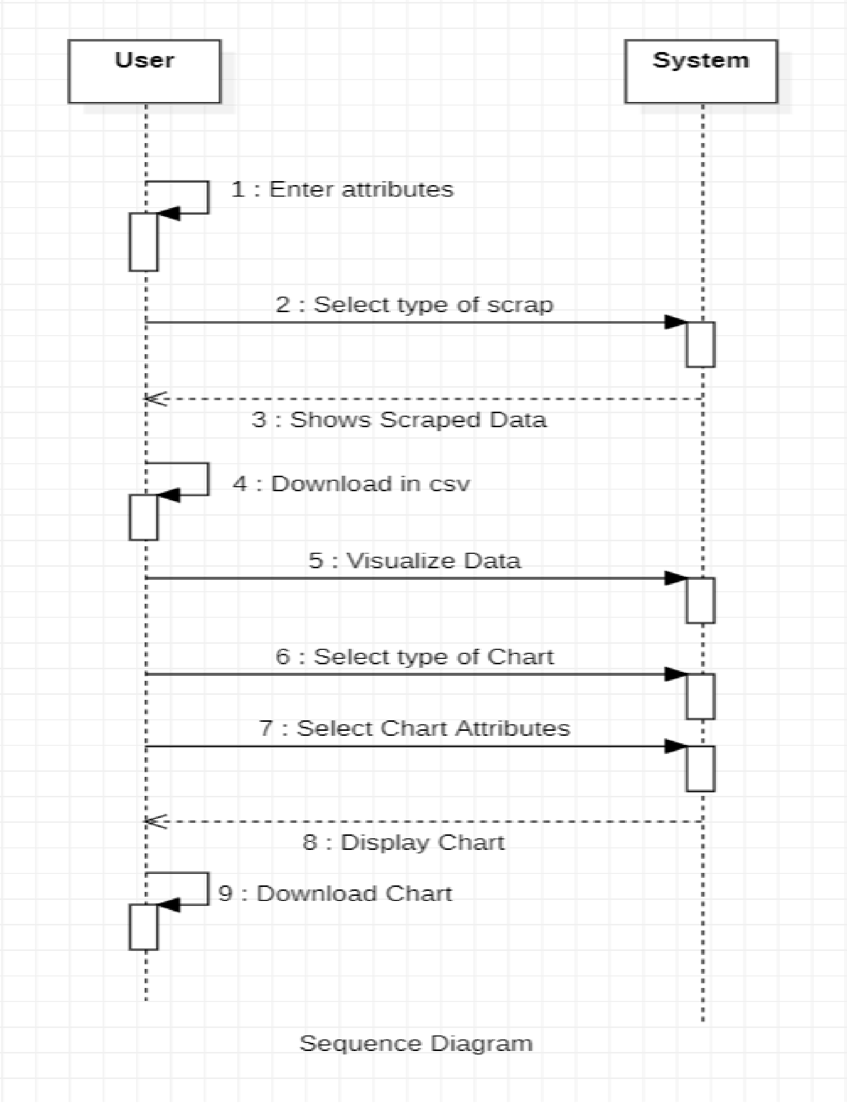


Figure 3.2 (Sequence Diagram)

3.2.3 Activity Diagram

An activity diagram shows the flow from activity to activity. An activity is a going, on-atomic execution within a state machine. An activity results in some action, results in a change of state or return of a value.

Activity Diagram commonly contains:

1. Activity states and action states.

2. Transition.

3. Objects, it may contain nodes and constraints.

Activity states and action states: An executable atomic computation is called action state, which cannot be decomposed. Activity state is non-atomic, decomposable and takes some duration to execute.

Transition:

It is the path from one state to the next state, represented as simple directed line.

Branching:

When an alternate path exists, branching arises which is represented by diamond. It has and incoming transition, two or more outgoing transitions.

Forking and Joining:

The synchronization bar when split one flow into two or more flows is called fork. When two or more flows are combined at synchronization bar, the bar is called join.

Swim Lanes:

Group work flow is called swim lanes. All groups are portioned by vertical solid lines. Each swim lane specifies locus of activities and has a unique name. Each swim lane is implemented by one or more classes. Transition may occur between objects across swim lanes.

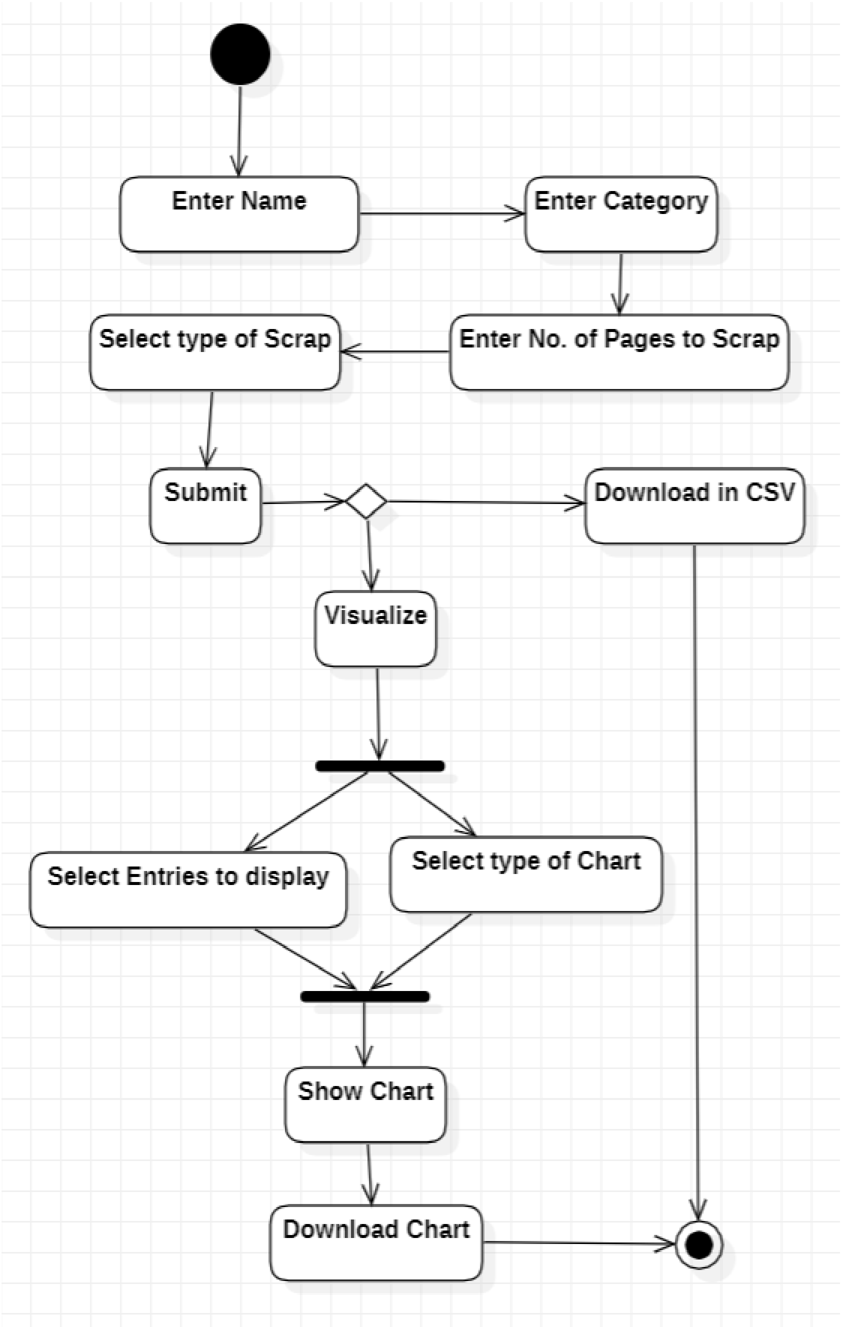


Figure 3.3 (Activity Diagram)

3.2.4 Component Diagram

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems. Component diagrams can also be described as a static implementation view of system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

Before drawing a component diagram, the following artifacts are to be identified clearly.

1. Files used in the system.

2. Libraries and other artifacts relevant to the application.

3. Relationships among the artifacts.

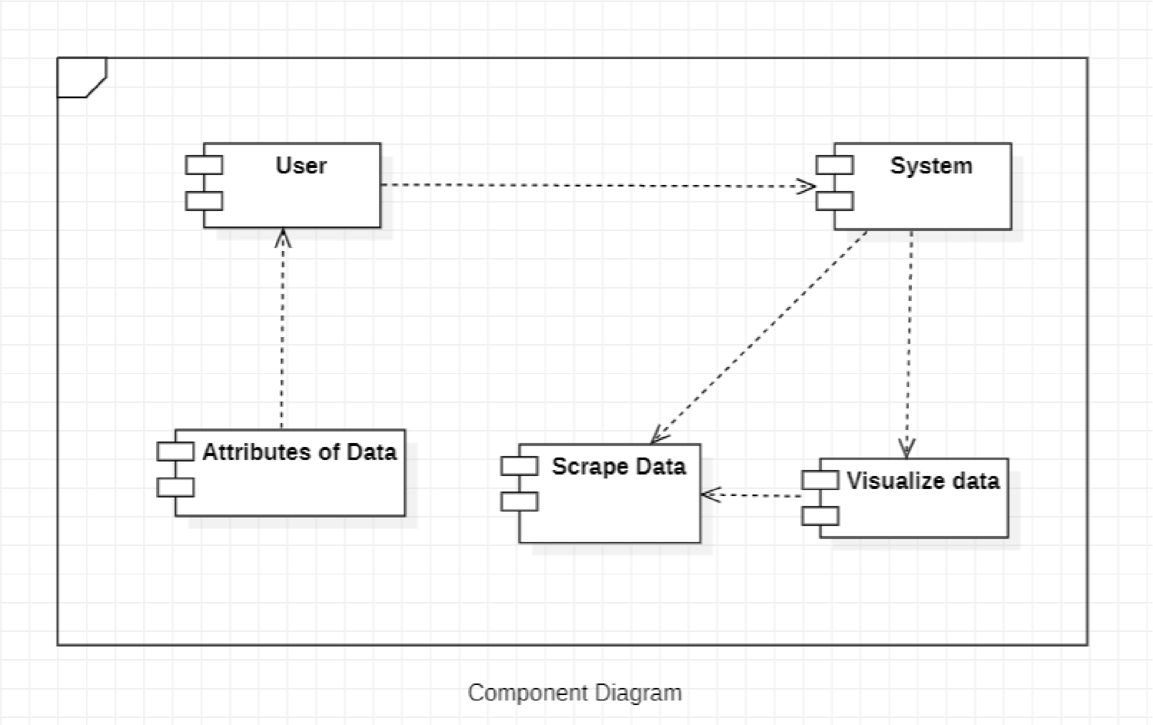


Figure 3.4 (Component Diagram)

CHAPTER 4

IMPLEMENTATION

4.1 Tools Used

1. VS Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE. This enables utilizing the debugging features of VSCode. Make sure the VSCode is configured with the right python environment.

2. Django

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

This framework uses a famous tag line: “The web framework for perfectionists with deadlines”.

By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

3. jQuery

jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

jQuery is widely famous with its philosophy of “Write less, do more.” This philosophy can be further elaborated as three concepts:

- Finding some elements (via CSS selectors) and doing something with them (via jQuery methods) i.e. locate a set of elements in the DOM, and then do something with that set of elements.
- Chaining multiple jQuery methods on a set of elements.
- Using the jQuery wrapper and implicit iteration.

4.2 Technology

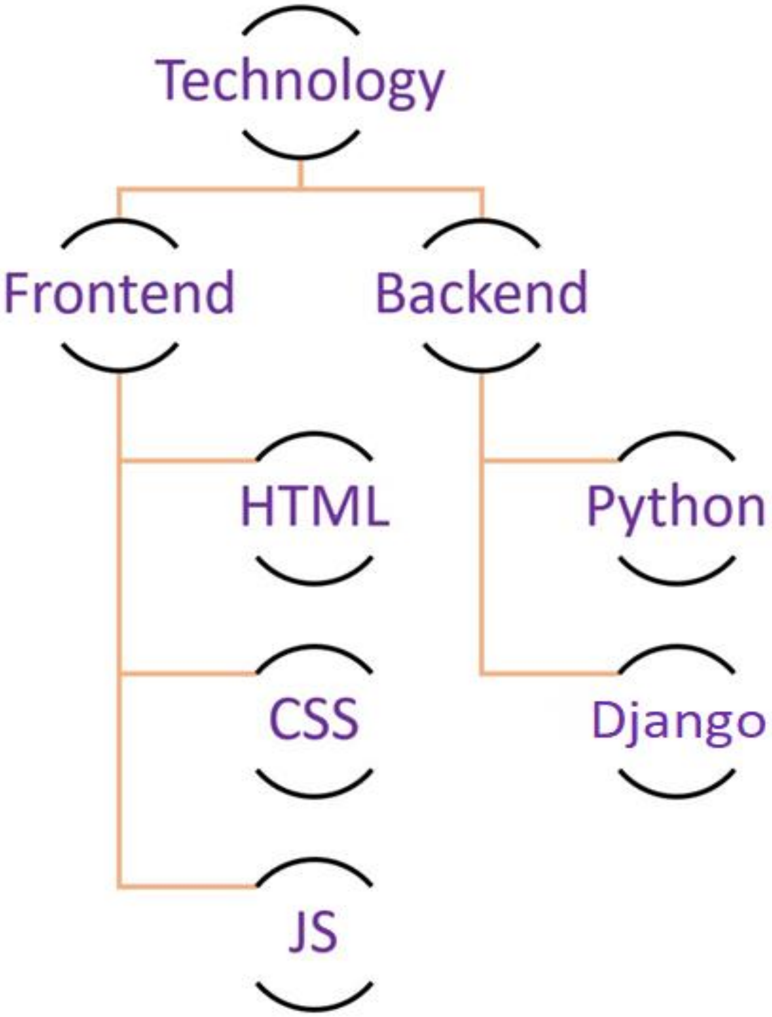


Figure 4.2 (Technology Used)

FRONTEND

- **HTML**
  HTML stands for Hyper Text Mark-up Language. HTML is the standard mark-up Language for creating Web pages. HTML describes the structure of a Web page. HTML consists of a series of elements. HTML elements tell the browser how to display the content.

- **CSS**
  Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

- **JavaScript**
  JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

BACKEND

- **Django**
  Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django follows the MVT design pattern (Model View Template).
  - Model - The data you want to present, usually data from a database.
  - View - A request handler that returns the relevant template and content - based on the request from the user.
  - Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

- **Python**
  Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems

4.2.1 Software Requirements

- Operating System: Windows.
- Web Technologies: Html, JavaScript, CSS, Python, Django
- Web Browser: IE, Mozilla Firefox or Google Chrome
- Browser Configuration: JavaScript must be enabled.

4.2.2 Hardware Requirements

- Processor: Intel Pentium 4 or higher or any other processor.
- RAM: Minimum of 512MB or higher
- Hard-drive space: 60 MB or higher
- Internet Connection: 4 Mbps or higher
- Controller: Keyboard and a Mouse

4.3 Sample Code

File Name: (fast_scrap.py)

```
from bs4 import BeautifulSoup
import pandas as pd
import requests
from .debug_utils import debug

context = {
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.5',
    'Host': 'www.amazon.in',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101
Firefox/89.0'
}
class FastScrapper:
    """A class to scrap amazon data"""
    def __init__(self, user_name, category, pages):
        self.df = pd.DataFrame(
            columns=[
                'user_name',
                'categories',
                'names',
                'rating',
                'total_rating',
                'cost_price',
                'selling_price',
                'links'
            ],
        )
        self.df = self.df.rename_axis('ID')
        self.category = category
        self.pages = int(pages)
        self.user_name = user_name
        self.data = dict()
        self.urls = list()
    def url_linker(self, *args):
            for page in range(0, self.pages):
            page_link                                                    =
f'https://www.amazon.in/s?k={self.category}&page={page+1}&ref=nb_sb_noss'
            self.urls.append(page_link)

    def scrapper(self, urls):
        names = list()
        SP = list()
        CP = list()
        rating = list()
        people = list()
        links = list()
        categories = list()
        user_names = list()

        for url in urls:
            response = requests.get(url, headers=context).content
            soup = BeautifulSoup(response, 'html.parser')
            # OPTIONS TRIED:
            #  a-section
            #  sg-col-inner
            all_blocks = soup.findAll('div', class_="sg-col-inner")
            #if len(all_blocks) == 0:
            #   debug(type='w', message="[WARNING] NO elements under this class found.")
            #    #pass
            for product in all_blocks:
                try:
                    name_obj = product.find('h2')
                    if name_obj is None:
                        #debug(type='e', message=f"OBJECT IS NONE!")
                        #debug(type='e', message="[PRODUCT IS SKIPPED]\n")
                        continue
                    else:
                        name = name_obj.get_text()
                    selling_price = product.find(
                                                 'span',
                                                 class_='a-price-whole'
                                                ).get_text()
                    cost_price = product.findAll(
                                                    'span',
                                                    class_='a-offscreen'
                                                )[-1].get_text().split('₹')[1]

                    try:
                        rat = product.find(
                                            'span',
                                            class_='a-icon-alt'
                                            ).get_text().split()[0]
                        peps = product.find(
                                            'span',
                                            class_='a-size-base'
                                            ).get_text()
                    except Exception as e:
                        debug(type='w', message=e)
                        continue

                    # Inside Link
                    extracted_href = product.find('a', class_='a-link-normal').get('href')
                    link = str()
                    if extracted_href.startswith('https:'):
                        continue
                    else:
                        link = 'https://www.amazon.in' + extracted_href
                    names.append(name)
                    SP.append(selling_price)
                    CP.append(cost_price)
                    rating.append(rat)
                    people.append(peps)
                    links.append(link)
                    categories.append(self.category)
                    user_names.append(self.user_name)

                except Exception as e:
                    debug(type='w', message=e)

        self.data['user_name'] = user_names
        self.data['categories'] = categories
        self.data['names'] = names
        self.data['rating'] = rating
```

```
        self.data['total_rating'] = people
        self.data['cost_price'] = CP
        self.data['selling_price'] = SP
        self.data['links'] = links

    def convert_to_df(self):
        df1 = pd.DataFrame(self.data)
        self.df = pd.concat((self.df, df1))

    def scrap(self):
        self.url_linker(self.category)
        self.scrapper(self.urls)
        self.convert_to_df()
        self.df['cost_price'] = self.df['cost_price'].map(lambda x: x.replace(',', "))
        self.df['selling_price'] = self.df['selling_price'].map(lambda x: x.replace(',', "))
        self.df['total_rating'] = self.df['total_rating'].map(lambda x: x.replace(',', "))
        my_df = self.df
        return my_df

    def save_to_csv(self):
        self.df.to_csv('csv/' + self.user_name + '.csv')
```

File Name: (slow_scrap.py)

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

context = {
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.5',
    'Host': 'www.amazon.in',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:89.0) Gecko/20100101
Firefox/89.0'
}


class Scrapper:
    """A class to scrap amazon data"""
    def __init__(self, user_name, category, pages):
        self.df = pd.DataFrame(
            columns=[
                'user_name',
                'brands',
                'categories',
                'names',
                'rating',
                'total_rating',
                'cost_price',
                'selling_price',
                'discount',
                'discount_per',
                'links'
            ],
        )
        self.df = self.df.rename_axis('ID')
        self.category = category
        self.pages = int(pages)
        self.user_name = user_name
        self.data = dict()
        self.urls = list()

    def url_linker(self, *args):
        for page in range(0, self.pages):

self.urls.append(f'https://www.amazon.in/s?k={self.category}&page={page+1}&ref=nb_
sb_noss')

    def scrapper(self, urls):
        names = list()
        SP = list()
        CP = list()
        rating = list()
        people = list()
        discount = list()
        discount_per = list()
        brand_name = list()
        links = list()
        categories = list()
        user_names = list()
        for url in urls:
            response = requests.get(url, headers=context).content
            soup = BeautifulSoup(response, 'html.parser')
            all_blocks = soup.findAll('div', class_='s-asin')
            all_blocks = all_blocks[:5]
            for product in all_blocks:
                try:
                    name = product.find('h2').get_text()
                    selling_price = product.find(
                        'span', class_='a-price-whole').get_text()
                    cost_price = product.findAll(
                        'span', class_='a-offscreen')[-1].get_text().split('₹')[1]
                    rat = product.find(
                        'span', class_='a-icon-alt').get_text().split()[0]
                    peps = product.find(
                        'span', class_='a-size-base').get_text()
                    # Inside Link
                    link = 'https://www.amazon.in' + \
                        product.find('a', class_='a-link-normal').get('href')
                    new_res = requests.get(link, headers=context).content
                    new_soup = BeautifulSoup(new_res, 'html.parser')
                    disc, disc_per = new_soup.find(
                        'td', 'a-span12 a-color-price a-size-base
priceBlockSavingsString').get_text().split()
                    disc = disc.split('₹')[1].replace(',', "")
                    disc_per = disc_per.lstrip('(').rstrip(')')[:2]

                    brand = dict()
                    lst = []
                    for i in new_soup.findAll('tr', 'a-spacing-small'):
                        lst.append([i.get_text().strip()])
                    for i in lst:
                        i = i[0].split('\n')
                        brand[i[0]] = i[-1]

                    brand_ = brand.get('Brand')
                    if not brand_:
                        continue
                    else:
                        brand_name.append(brand_)

                    names.append(name)
                    SP.append(selling_price)
                    CP.append(cost_price)
                    rating.append(rat)
                    people.append(peps)
                    discount.append(disc)
                    discount_per.append(disc_per)
                    links.append(link)
                    categories.append(self.category)
                    user_names.append(self.user_name)

                except Exception as e:
                    print(e)

        self.data['user_name'] = user_names
        self.data['brands'] = brand_name
        self.data['categories'] = categories
        self.data['names'] = names
        self.data['rating'] = rating
        self.data['total_rating'] = people
        self.data['cost_price'] = CP
        self.data['selling_price'] = SP
        self.data['discount'] = discount
        self.data['discount_per'] = discount_per
        self.data['links'] = links


    def convert_to_df(self):
        df1 = pd.DataFrame(self.data)
        self.df = pd.concat((self.df, df1))

    def scrap(self):
        self.url_linker(self.category)
        self.scrapper(self.urls)
        self.convert_to_df()
        # self.save_to_csv()
        self.df.discount_per = self.df.discount_per.map(lambda x: x.split('%')[0])
        self.df['cost_price'] = self.df['cost_price'].map(lambda x: x.replace(',', ''))
        self.df['selling_price'] = self.df['selling_price'].map(lambda x: x.replace(',', ''))
        self.df['total_rating'] = self.df['total_rating'].map(lambda x: x.replace(',', ''))
        my_df = self.df
        return my_df

    def save_to_csv(self):
        self.df.to_csv('csv/' + self.user_name + '.csv')

if __name__ == '__main__':
    scrapper = Scrapper('jay', 'skincream', 1)
    scrapper.scrap()
```

4.4 Testing

The purpose of testing is to discover errors. Testing is a process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for well plan through testing.

Testing Objectives:

There are several rules that can serve as testing objectives they are:

- Testing is a process of executing program with the intent of finding an error.
- A good test case is the one that has a high probability of finding an undiscovered error.

Types of Testing:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at different phases of software development are:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

[1] Unit Testing:

Unit testing is done on individual models as they are completed and becomes executable. It is confined only to the designer's requirements. Unit testing is different from and should be preceded by other techniques, including:

- Inform Debugging

- Code Inspection

Each module can be testing using the following two strategies:

Black Box testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been used to find error in the following categories: Incorrect or missing functions

- Interface errors.

- Errors in data structures are external database access.

- Performance error.

- Initialization and termination of errors.

- In this testing only the output is checked for correctness.

- The logical flow of data is not checked.

White Box testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been used to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.

- Execute all loops at their boundaries and within their operational bounds.

- Execute internal data structures to ensure their validity.

[2] Integration Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. It is typically performed by developers, especially at the lower, module to module level. Testers become involved in higher levels.

[3] System Testing

Involves in house testing of the entire system before delivery to the user. The aim is to satisfy the user the system meets all requirements of the client's specifications. It is conducted by the testing organization if a company has one. Test data may range from and generated to production. Requires test scheduling to plan and organize:

1. Inclusion of changes/fixes.

2. Test data to use

One common approach is graduated testing: as system testing progresses and (hopefully) fewer and fewer defects are found, the code is frozen for testing for increasingly longer time periods.

[4] Acceptance Testing

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

User Acceptance Test (UAT) "Beta testing": Acceptance testing in the customer environment. Requirements traceability:

- Match requirements to test cases.
- Every requirement has to be cleared by at least one test case.
- Display in a matrix of requirements vs. test cases.

4.5 Verification

The set of Test Cases are used to test the functionality of each module if that module works properly then that Test Cases marked as Pass or else Fail.

Test Id	Test case	Input Description	Expected Output	Test Status
1	Enter required Information.	User will fill entries as required and select type of scraping according to data and time.	Data will be scraped and all attributes of a product will be shown.	Pass
2	To download data	The scraped data will be downloaded by pressing download button	Data downloaded in csv	Pass
3	To visualize data	By clicking on visualize button, the scraped data will be ready to be visualized.	Visualizer window will open and options to visualize will be shown	Pass
4	Select chart and attributes to visualize.	By selecting any one of the given chart, no. of required rows and columns.	The chart will be shown.	Pass
5	Output	Chart will be displayed as chosen and insights can we concluded from it.	Chart can be downloaded for future references	Pass

Table 4.1 (Verification of test cases.)

4.6 Validation

Validation is often conducted after the completion of the entire software development process. It checks if the client gets the product they are expecting. Validation focuses only on the output; it does not concern itself about the internal processes and technical intricacies of the development process.

Validation helps to determine if the software team has built the right product. Validation is a one-time process that starts only after verifications are completed. Software teams often use a wide range of validation methods, including White Box Testing (non-functional testing or structural/design testing) and Black Box Testing (functional testing).

White Box Testing is a method that helps validate the software application using a predefined series of inputs and data. Here, testers just compare the output values against the input values to verify if the application is producing output as specified by the requirements.

There are three vital variables in the Black Box Testing method (input values, output values, and expected output values). This method is used to verify if the actual output of the software meets the anticipated or expected output.

The main advantages of validation processes are:

It ensures that the expectations of all stakeholders are fulfilled.

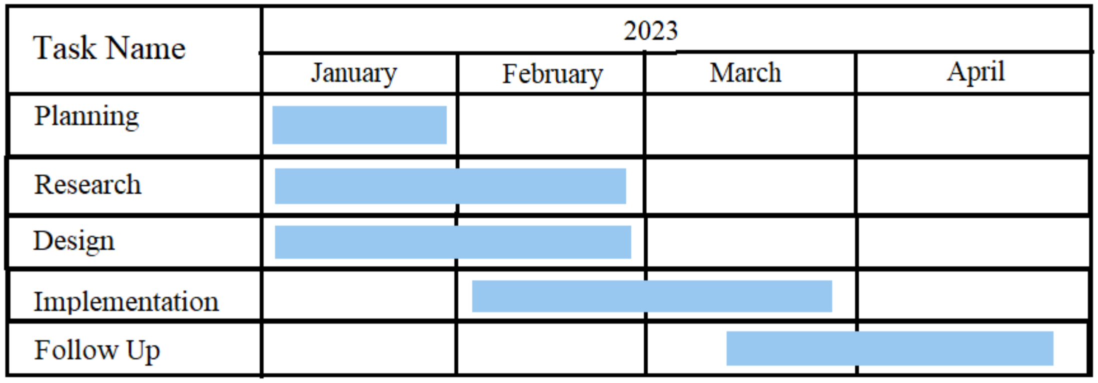
It enables software teams to take corrective action if there is a mismatch between the actual product and the anticipated product.

It improves the reliability of the end-product.

CHAPTER 5

PROJECT PLAN

5.1 Gantt Chart



Task Name	2023			
	January	February	March	April
Planning	■			
Research	■	■		
Design	■	■		
Implementation		■	■	
Follow Up			■	■

Table 5.1 (Gantt chart)

5.2 Effort Schedule & Cost estimation

Estimates of effort and cost are generally based on results of analysis using models or historical data applied to size, activities, and other planning parameters. Confidence in these estimates is based on rationale for the selected model and the nature of the data. There can be occasions when available historical data do not apply, such as when efforts are unprecedented or when the type of task does not fit available models. For example, an effort can be considered unprecedented if the organization has no experience with such a product.

Unprecedented efforts are more risky, require more research to develop reasonable bases of estimate, and require more management reserve. The uniqueness of the project should be documented when using these models to ensure a common understanding of any assumptions made in the initial planning phases.

1. Collect models or historical data to be used to transform the attributes of work products and tasks into estimates of labour hours and costs, then it makes a parametric model.

Many parametric models have been developed to help estimate cost and schedule. The use of these models as the sole source of estimation is not recommended because these models are based on historical project data that may or may not be pertinent to the project. Multiple models and methods can be used to ensure a high level of confidence in the estimate.

Historical data should include the cost, effort, and schedule data from previously executed projects and appropriate scaling data to account for differing sizes and complexity.

2. Include supporting infrastructure needs when estimating effort and cost.

The supporting infrastructure includes resources needed from a development and sustainment perspective for the product and then creates environment product.

Consider the infrastructure resource needs in the development environment, the test environment, the production environment, the operational environment, or any appropriate combination of these environments when estimating effort and cost.

5.3 Work Breakdown Structure

A Work Breakdown Structure includes dividing a large and complex project into simpler, manageable and independent tasks. The root of this tree (structure) is labelled by the Project name itself. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities becomes undividable and independent. It follows a Top-Down approach.

Firstly, the project managers and top level management identifies the main deliverables of the project.

After this important step, these main deliverables are broke down into smaller higher-level tasks and this complete process is done recursively to produce much smaller independent tasks. It depends on the project manager and team that up to which level of detail they want to break down their project.

Generally the lowest level tasks are the most simplest and independent tasks and takes less than two weeks' worth of work. Hence, there is no rule for up to which level we may build the work breakdown structure of the project as it totally depends upon the type of project we are working on and the management of the company.

The efficiency and success of the whole project majorly depends on the quality of the Work Breakdown Structure of the project and hence, it implies its importance.

To use a work breakdown structure effectively, it is important to include all components of a project (remember that 100% rule described above) but without too much detail. Turns out, there can be too much of a good thing when it comes to the WBS.

To create a WBS:

1. Define the project. The first step in creating a work breakdown structure is to clearly establish the project. For some projects, this might be fairly straightforward. For other projects, it might require refining the actual scope of the project so that the WBS is scaled appropriately and doesn't become unwieldy.

2. Set project boundaries. Once the project is defined and described, you can set boundaries on what is and isn't included in the WBS.

3. Identify project deliverables. This will include high-level deliverables associated with the project, such as a Project Scope Statement or Mission Statement.

4. Define Level 1 elements. Remember the 100% rule while creating the Level 1 deliverables.

5. Break down each of the Level 1 elements. The process of breaking down Level 1 elements is called decomposition. It consists of breaking down a task into smaller and smaller pieces, applying the 100% rule at each level. At each subsequent level, ask yourself whether further decomposition would improve project management. Continue breaking down the elements until the answer to that question is "no." When you've completed the decomposition process for each element in Level 1, the WBS is complete.

6. Identify team members. Identify an individual or team who is responsible for each element.

7. Create a Gantt chart to accompany the WBS. A Gantt chart shows activities over time so that you can visually see information related to the schedule of the project and its various activities.

5.4 Performance Measures

The accuracy of text recognition from images depends on the text extraction. The problem of correct segmentation of isolated characters is the most difficult process in text extraction. A simple solution is proposed to this problem based on efficient edge and connected component-based segmentation and optical character recognition techniques. The text regions have been successfully extracted irrespective of the text font and size using the proposed system. The proposed method has provided a comprehensive model of text extraction and recognition in images. In order to evaluate the performance of the proposed method, images from a variety of sources such as logos, book covers, printed advertisements and text effects have been tested on the system. The proposed system is applied with the help of fuzzy logic to extract the text regions from background and recognize the characters from extracted text regions.

There are several performance evaluations to estimate the algorithm for text extraction. Most of the approaches quoted here used Precision, Recall and F-Score metrics to evaluate the performance of the algorithm. Precision, Recall and F-Score rates are computed based on the number of correctly detected characters (CDC) in an image, in order to evaluate the efficiency and robustness of the fuzzy algorithm.

The performance metrics are as follows:

False Positives: False Positives (FP) / False alarms are those regions in the image which are actually not characters of a text, but have been detected by the algorithm as text.

False Negatives: False Negatives (FN)/ Misses are those regions in the image which are actually text characters, but have not been detected by the algorithm.

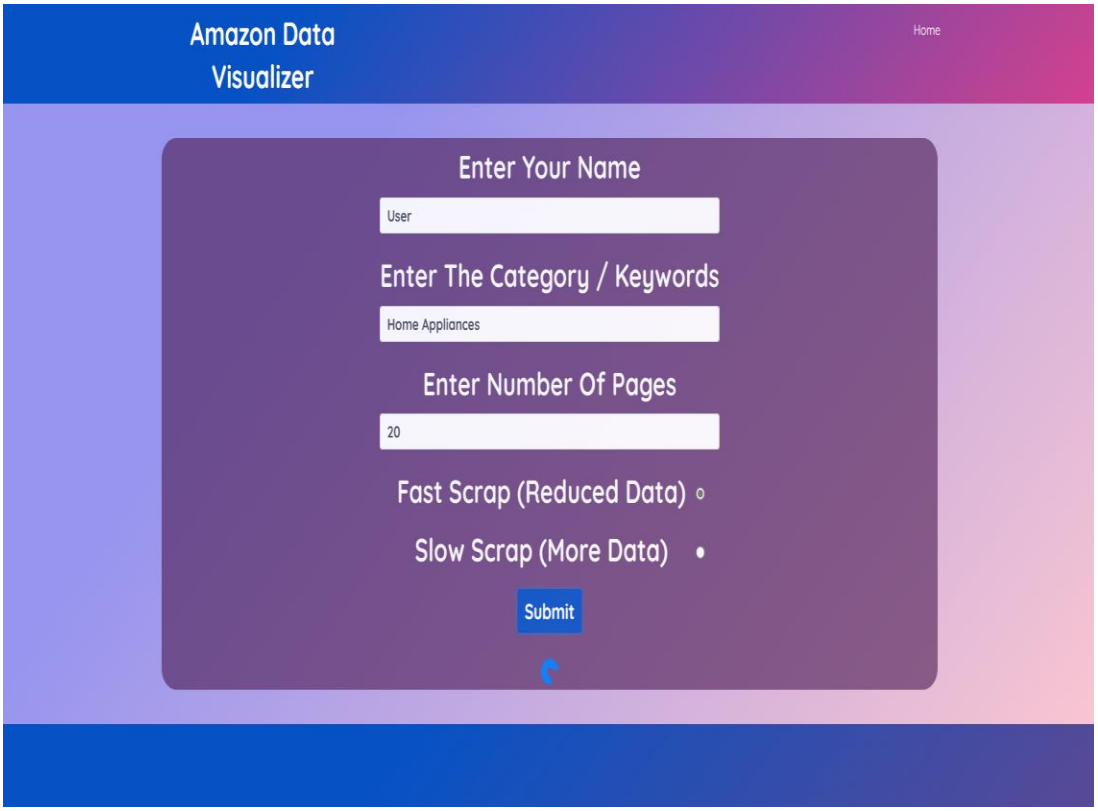
Precision rate: Precision rate (P) is defined as the ratio of correctly detected characters to the sum of correctly detected characters plus false positives.

Recall rate: Recall rate (R) is defined as the ratio of the correctly detected characters to sum of correctly detected characters plus false negatives.

F-score: F-Score is the harmonic mean of recall and precision rates. The goal of performance evaluation for a text extraction system is to measure the difference between the expected text output and the actual text output of the system. Good performance evaluation methods can provide valuable information not only for system evaluation and comparison, but also for system selection and improvement.

CHAPTER 6

PROJECT SCREENSHOT



Amazon Data Visualizer

Enter Your Name

User

Enter The Category / Keywords

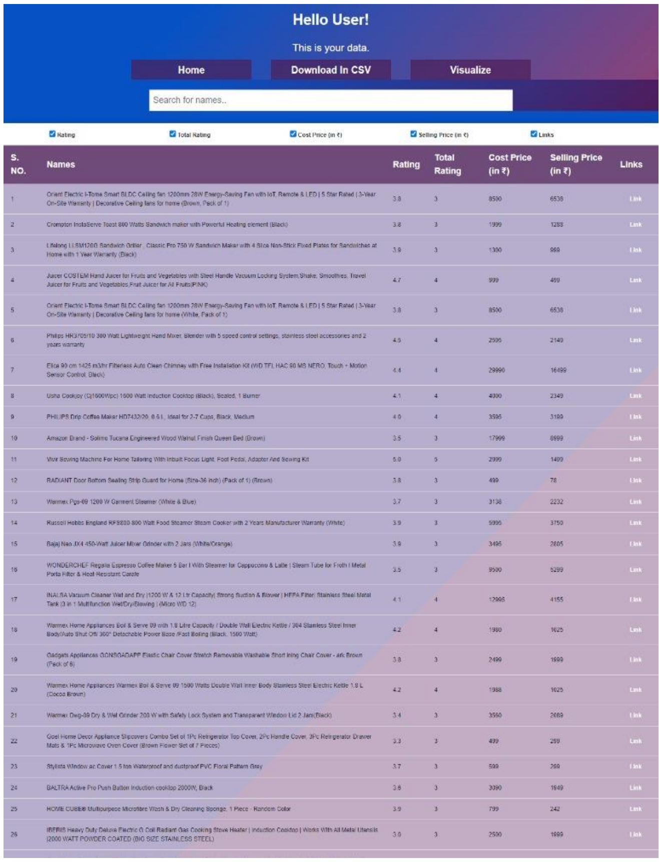
Home Appliances

Enter Number Of Pages

20

Fast Scrap (Reduced Data) ○

Slow Scrap (More Data) ●

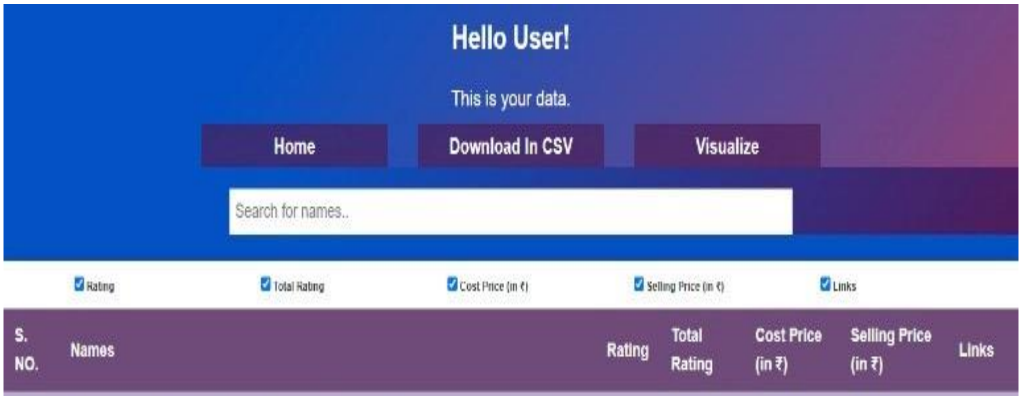


Hello User!

This is your data.

☑ Rating ☑ Total Rating ☑ Cost Price (in ₹) ☑ Selling Price (in ₹) ☑ Links

S. NO.	Names	Rating	Total Rating	Cost Price (in ₹)	Selling Price (in ₹)	Links
1	Orient Electric I-Tome Smart BLDC Ceiling fan 1200mm 28W Energy-Saving Fan with IoT, Remote & LED | 5 Star Rated | 3-Year On-Site Warranty | Decorative Ceiling fans for home (Brown, Pack of 1)	3.8	3	8500	6538	Link
2	Crompton InstaServe Toast 800 Watts Sandwich maker with Powerful Heating element (Black)	3.8	3	1999	1288	Link
3	Lifelong LLSM120G Sandwich Griller , Classic Pro 750 W Sandwich Maker with 4 Slice Non-Stick Fixed Plates for Sandwiches at Home with 1 Year Warranty (Black)	3.9	3	1300	999	Link
4	Juicer COSTEM Hand Juicer for Fruits and Vegetables with Steel Handle Vacuum Locking System,Shake, Smoothies, Travel Juicer for Fruits and Vegetables,Fruit Juicer for All Fruits(PINK)	4.7	4	999	499	Link
5	Orient Electric I-Tome Smart BLDC Ceiling fan 1200mm 28W Energy-Saving Fan with IoT, Remote & LED | 5 Star Rated | 3-Year On-Site Warranty | Decorative Ceiling fans for home (White, Pack of 1)	3.8	3	8500	6538	Link
6	Philips HR3705/10 300 Watt Lightweight Hand Mixer, Blender with 5 speed control settings, stainless steel accessories and 2 years warranty	4.5	4	2595	2149	Link
7	Elica 90 cm 1425 m3/hr Filterless Auto Clean Chimney with Free Installation Kit (WD TFL HAC 90 MS NERO, Touch + Motion Sensor Control, Black)	4.4	4	29990	16499	Link
8	Usha Cookjoy (Cj1600Wpc) 1600 Watt Induction Cooktop (Black), Sealed, 1 Burner	4.1	4	4000	2349	Link
9	PHILIPS Drip Coffee Maker HD7432/20, 0.6 L, Ideal for 2-7 Cups, Black, Medium	4.0	4	3595	3199	Link
10	Amazon Brand - Solimo Tucana Engineered Wood Walnut Finish Queen Bed (Brown)	3.5	3	17999	8999	Link
11	Vivir Sewing Machine For Home Tailoring With Inbuilt Focus Light, Foot Pedal, Adapter And Sewing Kit	5.0	5	2999	1499	Link
12	RADIANT Door Bottom Sealing Strip Guard for Home (Size-36 inch) (Pack of 1) (Brown)	3.8	3	499	78	Link
13	Warmex Pgs-09 1200 W Garment Steamer (White & Blue)	3.7	3	3138	2232	Link
14	Russell Hobbs England RFS800 800 Watt Food Steamer Steam Cooker with 2 Years Manufacturer Warranty (White)	3.9	3	5995	3750	Link
15	Bajaj Neo JX4 450-Watt Juicer Mixer Grinder with 2 Jars (White/Orange)	3.9	3	3495	2605	Link
16	WONDERCHEF Regalia Espresso Coffee Maker 5 Bar I With Steamer for Cappuccino & Latte | Steam Tube for Froth I Metal Porta Filter & Heat-Resistant Carafe	3.5	3	9500	5299	Link
17	INALSA Vacuum Cleaner Wet and Dry |1200 W & 12 Ltr Capacity| Strong Suction & Blower | HEPA Filter| Stainless Steel Metal Tank |3 in 1 Multifunction Wet/Dry/Blowing | (Micro WD 12)	4.1	4	12995	4155	Link
18	Warmex Home Appliances Boil & Serve 09 with 1.8 Litre Capacity / Double Wall Electric Kettle / 304 Stainless Steel Inner Body/Auto Shut Off/ 360° Detachable Power Base /Fast Boiling (Black, 1500 Watt)	4.2	4	1980	1025	Link
19	Gadgets Appliances GONSGADAPP Elastic Chair Cover Stretch Removable Washable Short lning Chair Cover - ark Brown (Pack of 6)	3.8	3	2499	1899	Link
20	Warmex Home Appliances Warmex Boil & Serve 09 1500 Watts Double Wall Inner Body Stainless Steel Electric Kettle 1.8 L (Cocoa Brown)	4.2	4	1988	1025	Link
21	Warmex Dwg-09 Dry & Wet Grinder 200 W with Safety Lock System and Transparent Window Lid 2 Jars(Black)	3.4	3	3560	2089	Link
22	Goel Home Decor Appliance Slipcovers Combo Set of 1Pc Refrigerator Top Cover, 2Pc Handle Cover, 3Pc Refrigerator Drawer Mats & 1Pc Microwave Oven Cover (Brown Flower Set of 7 Pieces)	3.3	3	499	299	Link
23	Stylista Window ac Cover 1.5 ton Waterproof and dustproof PVC Floral Pattern Grey	3.7	3	599	299	Link
24	BALTRA Active Pro Push Button Induction cooktop 2000W, Black	3.6	3	3090	1849	Link
25	HOME CUBE® Multipurpose Microfibre Wash & Dry Cleaning Sponge, 1 Piece - Random Color	3.9	3	799	242	Link
26	IBERIS Heavy Duty Deluxe Electric G Coil Radiant Gas Cooking Stove Heater | Induction Cooktop | Works With All Metal Utensils |2000 WATT POWDER COATED (BIG SIZE STAINLESS STEEL)	3.0	3	2500	1899	Link

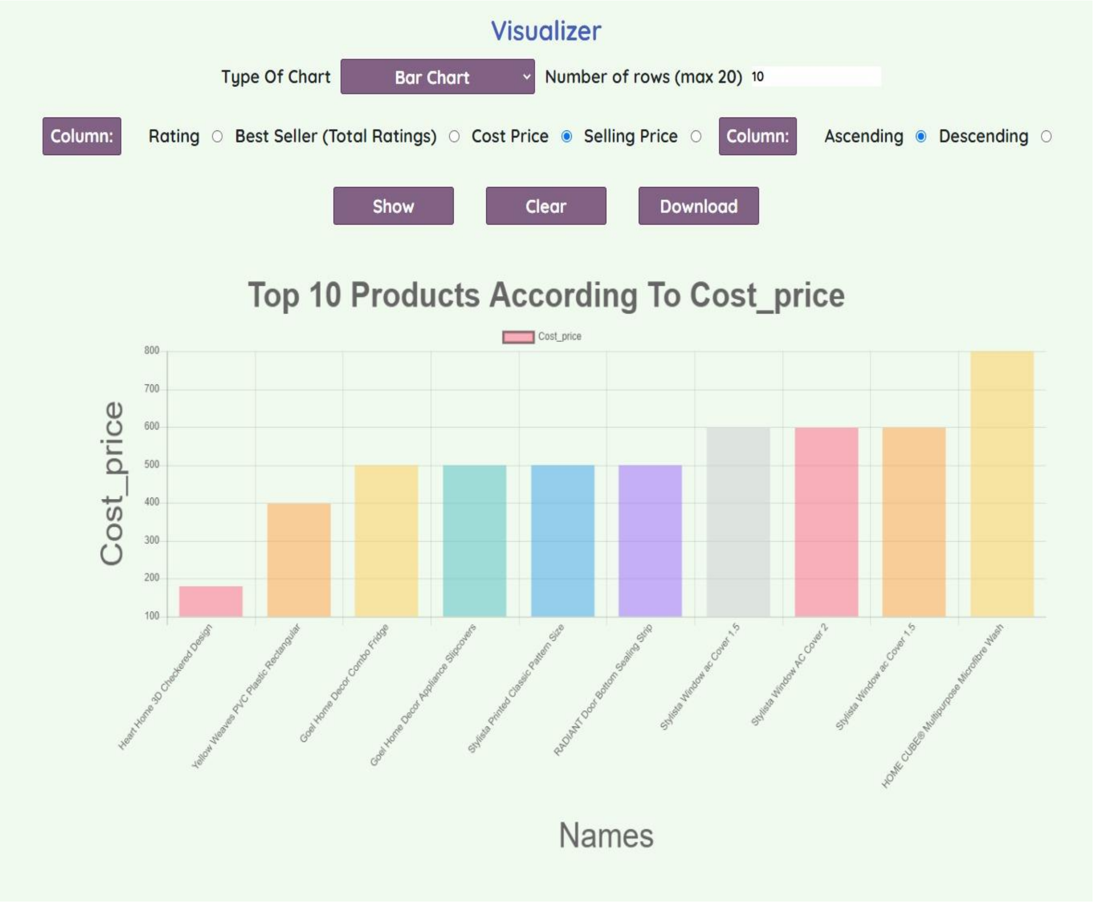


Hello User!

This is your data.

☑ Rating ☑ Total Rating ☑ Cost Price (in ₹) ☑ Selling Price (in ₹) ☑ Links

S. NO.	Names	Rating	Total Rating	Cost Price (in ₹)	Selling Price (in ₹)	Links



Visualizer

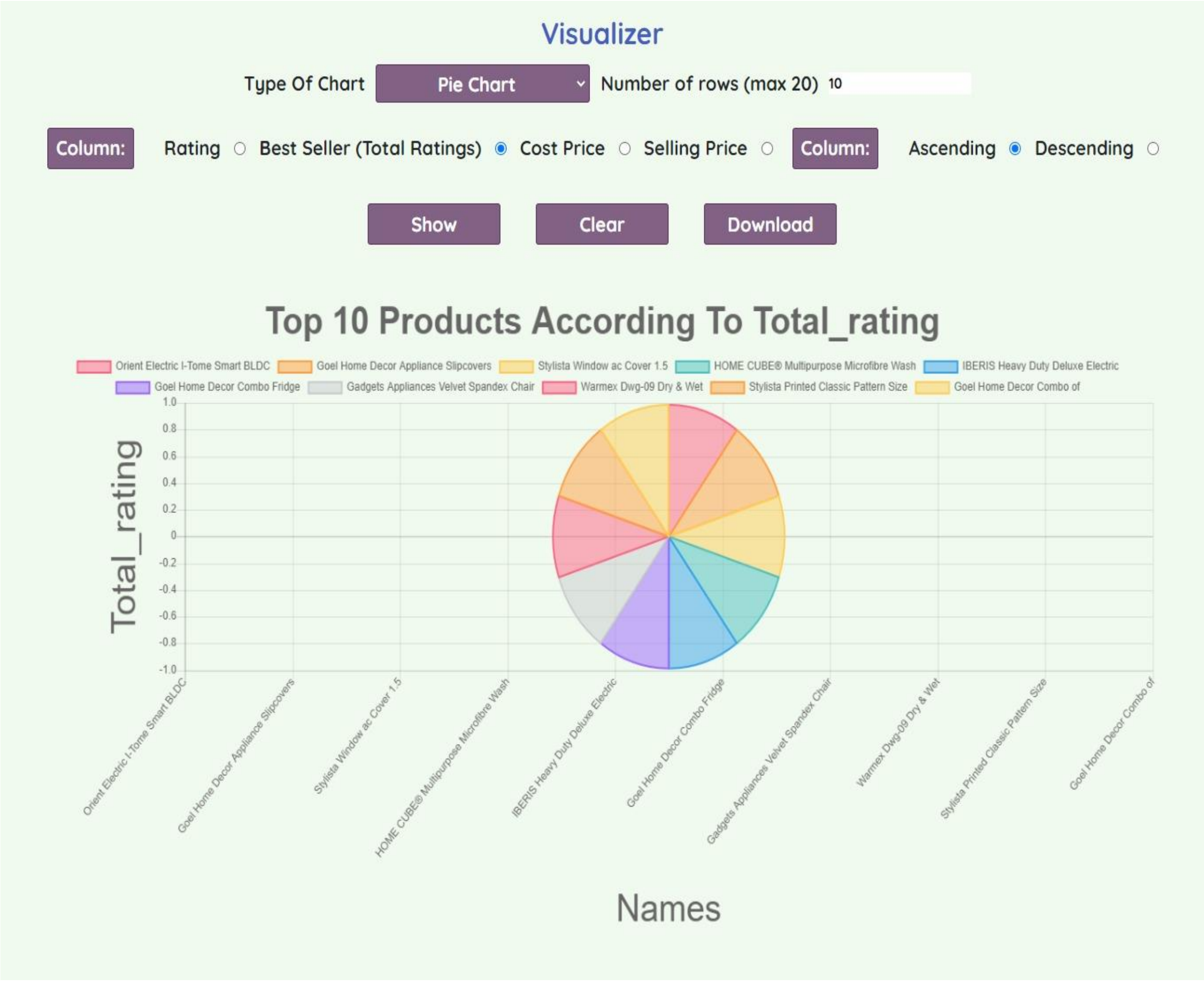
Type Of Chart: Bar Chart

Number of rows (max 20): 10

Column: Rating ○ Best Seller (Total Ratings) ○ Cost Price ● Selling Price ○

Column: Ascending ● Descending ○

Show
Clear
Download
Top 10 Products According To Cost_price
Cost_price
Cost_price
800
700
600
500
400
300
200
100
Heart Home 3D Checkered Design
Yellow Weaves PVC Plastic Rectangular
Goel Home Decor Combo Fridge
Goel Home Decor Appliance Slipcovers
Stylista Printed Classic Pattern Size
RADIANT Door Bottom Sealing Strip
Stylista Window ac Cover 1.5
Stylista Window AC Cover 2
Stylista Window ac Cover 1.5
HOME CUBE® Multipurpose Microfibre Wash
Names



Visualizer

Type Of Chart: Pie Chart

Number of rows (max 20): 10

Column: Rating ○ Best Seller (Total Ratings) ● Cost Price ○ Selling Price ○

Column: Ascending ● Descending ○

Top 10 Products According To Total_rating

Show
Clear
Download
🟥 Orient Electric I-Tome Smart BLDC
🟧 Goel Home Decor Appliance Slipcovers
🟨 Stylista Window ac Cover 1.5
🟩 HOME CUBE® Multipurpose Microfibre Wash
🟦 IBERIS Heavy Duty Deluxe Electric
🟪 Goel Home Decor Combo Fridge
⬜ Gadgets Appliances Velvet Spandex Chair
🟥 Warmex Dwg-09 Dry & Wet
🟧 Stylista Printed Classic Pattern Size
🟨 Goel Home Decor Combo of
1.0
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1.0
Total_rating
Orient Electric I-Tome Smart BLDC
Goel Home Decor Appliance Slipcovers
Stylista Window ac Cover 1.5
HOME CUBE® Multipurpose Microfibre Wash
IBERIS Heavy Duty Deluxe Electric
Goel Home Decor Combo Fridge
Gadgets Appliances Velvet Spandex Chair
Warmex Dwg-09 Dry & Wet
Stylista Printed Classic Pattern Size
Goel Home Decor Combo of
Names

CHAPTER 7

CONCLUSION/ FUTURE SCOPE

7.1 Conclusion

Web scraping is a powerful tool for automatically extracting information from websites.

It is important to use web scraping responsibly and be mindful of the website’s performance to avoid any legality issues. Overall, it can help us to improve our products, and services, analyse customer feedback and stay ahead of the competition.

In this project, we have made a web based platform, to scrape data from Amazon, users can collect data as per their needs, they can visualize it graphically and get meaningful insights from the data.

7.2 Future Scope

The proposed solution used for Amazon Data Visualizer can be extended further, apart from providing analysis and visualization of only one website or application.

Web Scraping and its visualization can be done for different platforms in a single application. It will provide more functionality in the application and can be extended with more features according to future requirements.

Bibliography

1. Vojtech Drax (February 2018), Data Extraction from websites, [13-15].

2. Berlind, D., (17 November 2017), APIs Are Like User Interfaces Just With Different Users in Mind [Online] [Zugriff].

3. Myriam Ertz, (2021), Web Scraping Techniques and Applications, [384 -385]

4. Google Research through multiple websites like geeksforgeeks, bplogix.com, w3schools.com, etc.