M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur, Tamilnadu.

A Minor Project Report
On

Building a Web Scraping Framework for Data Extraction and Analysis

Under the guidance of

Ms. SRIMATHI V

CORPORATE TRAINER-IBM

Submitted by

1. VARSHINI.J	-927621BAD121
2. VASANTH KUMAR.S	-927621BAD122
3. VENKATRAGAV.N	- 927621BAD123

DEPARTMENT OF
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)
KARUR – 639113

TABLE OF CONTENTS

CHAPTER-1

INTRODUCTION

INTRODUCTION

PROBLEM STATEMENT:

OBJECTIVES: Develop a framework to automate the collection of data from diverse web sources.Handle Implement solutions to extract data from websites using python for content rendering. Establish processes for data validation, cleaning, and preprocessing. Design a scalable system to handle large volumes of data and concurrent scraping operations.Legal and Ensure the framework adheres to legal standards and website terms of service.Error Handling and Reliability: Incorporate robust error handling, logging, and monitoring mechanisms.

Empathize: Identify the needs and pain points of end-users, including data scientists, analysts, and business decision-makers who use Python.User Interviews: Conduct interviews with potential users to understand their challenges with current data collection methods and their familiarity with Python.Surveys and Feedback: Gather feedback from a broader audience to capture diverse requirements and preferences, focusing on Python users.

Define: Clearly articulate the core issues faced by Python users in web data extraction and analysis.Requirements Gathering: Document detailed functional and non-functional requirements based on user feedback, emphasizing Python-specific needs.Develop use cases to illustrate typical user scenarios and interactions with the Python framework.

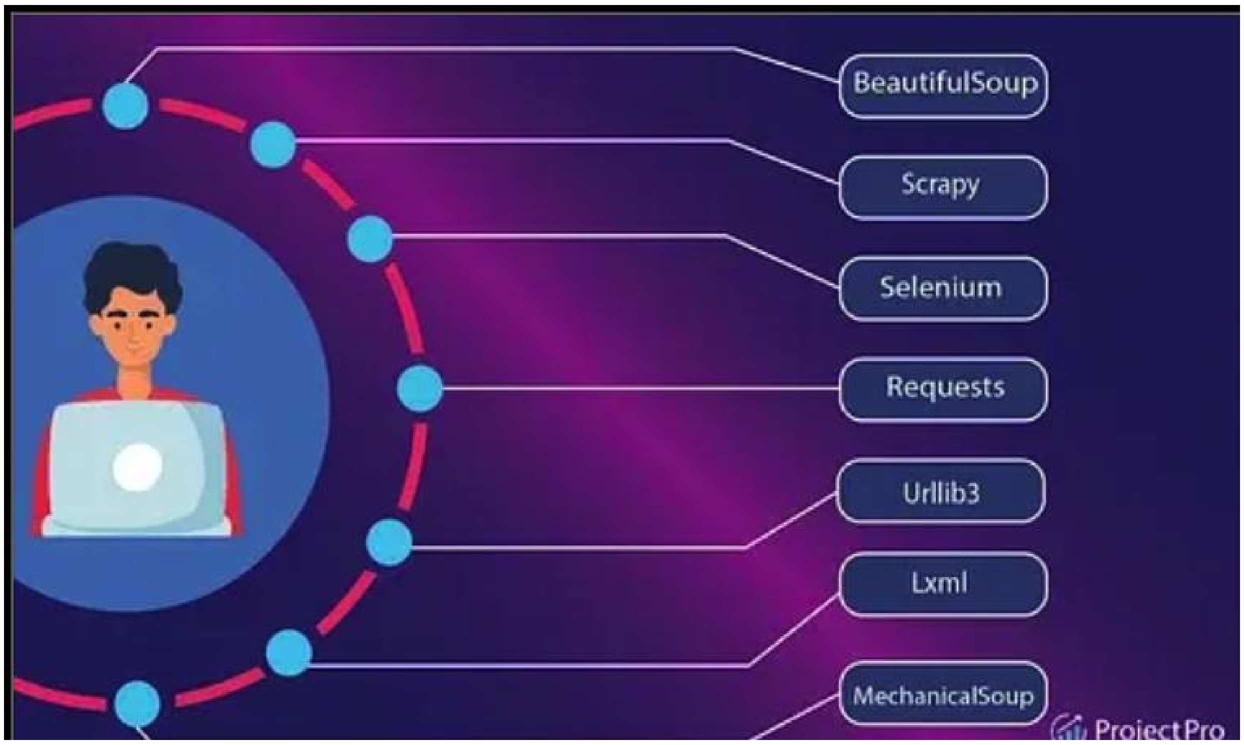
Ideate: Conduct brainstorming sessions to generate innovative ideas and approaches for the Python framework.Solution Mapping: Create mind maps and flowcharts to visualize potential Python-based solutions and their components.Evaluation: Assess the feasibility, scalability, and impact of different ideas to shortlist the most promising Python solutions.

Prototype: Develop a minimum viable product (MVP) in Python to demonstrate key functionalities of the framework.User Interface Mockups: Create mockups of the user interface in Python to visualize the user experience and gather feedback.Iterative Development: Implement the prototype in stages, continuously refining based on user feedback and testing results, using Python development tools..

Test: Conduct usability tests with real Python users to identify any issues with the user interface and overall user experience.Use tools like Pytest for unit testing and Selenium for end-to-end testing to simulate user interactions.Functionality Testing:Test all features of the Python framework to ensure they work as expected under various scenarios.Implement automated testing with unittest or pytest to verify the correctness of code. Performance Testing:Assess the framework's performance and scalability under different loads and conditions using tools like Locust or Apache JMeter.Conduct stress testing to determine the maximum capacity of the framework and identify bottlenecks.Security Testing Perform security testing to identify vulnerabilities and ensuredat protetion using tools like Bandit for static analysis and for dynamic analysis.

CHAPTER-2

PROJECT&PROPOSEDMANAGEMENT



Project Management: .

Project Planning: Effective project planning is critical for laying a strong foundation for the project. It includes defining the project scope, setting objectives, and determining deliverables. A detailed project timeline with clear milestones and deadlines is established, along with resource allocation to ensure that the necessary human, financial, and technical resources are available.

Requirements Gathering and Analysis: Gathering and analyzing requirements is vital for understanding the needs of stakeholders and users. This process involves conducting stakeholder interviews to capture their expectations and requirements. Creating user stories and use cases helps in documenting functional requirements. Technical specifications are defined to outline the software, hardware, and integration needs. Prioritizing requirements based on importance and feasibility ensures that critical features are developed first.

Development and Implementation: The development phase focuses on building the web scraping framework. This includes designing the framework's architecture, coding, and integrating various libraries and tools such as BeautifulSoup, Scrapy, . Rigorous testing is conducted through unit tests, integration tests, and system tests to ensure functionality and reliability. Comprehensive documentation is created to guide users and developers on

Monitoring and Control: Monitoring and control are essential for keeping the project on track. Progress is tracked using project management tools like Jira or Trello. Quality assurance measures are implemented to maintain high standards. Issue management processes are established to log, track, and resolve bugs and issues promptly. Change management is practiced to handle modifications to the project scope, requirements, or timeline systematically.

Deployment and Maintenance:Deploying the web scraping framework and providing ongoing maintenance are crucial for its success. Deployment planning ensures a smooth rollout, including user training and support setup. Collecting user feedback helps identify areas for improvement. Regular maintenance, bug fixes, and updates are provided to keep the framework up-to-date and functional. Continuous performance monitoring ensures the

PROPOSED SYSTEM:

This the proposed system based on our ideology:

The proposed Python-based web scraping framework aims to democratize data extraction and analysis, making it accessible and usable for individuals and organizations of all sizes. This system leverages Python's powerful libraries to create an efficient, user-friendly, and flexible solution that addresses the needs of various users, from beginners to experts, without requiring extensive technical knowledge. The framework is designed with an ideology of inclusivity, providing a seamless experience for users at different skill levels. Its modular architecture enhances scalability, maintainability, and flexibility, allowing customization and extension to meet specific needs.

The system includes a user-friendly interface, featuring a graphical user interface (GUI) for those who prefer visual interactions, a command-line interface (CLI) for advanced users, and a web-based dashboard for managing and monitoring scraping tasks remotely. The framework's modular design includes specific components for data extraction, dynamic content handling, data cleaning and validation, and seamless integration with data analysis libraries like Pandas, NumPy, Matplotlib, and Seaborn.

An easy setup process with pre-configured templates and wizards reduces the learning curve and enables quick deployment. Automated data extraction features, such as task scheduling and auto-retry mechanisms, ensure efficient and continuous data collection. To handle dynamic web content, the framework uses headless browsers that can interact with JavaScript-driven websites, ensuring comprehensive data extraction from modern web applications.

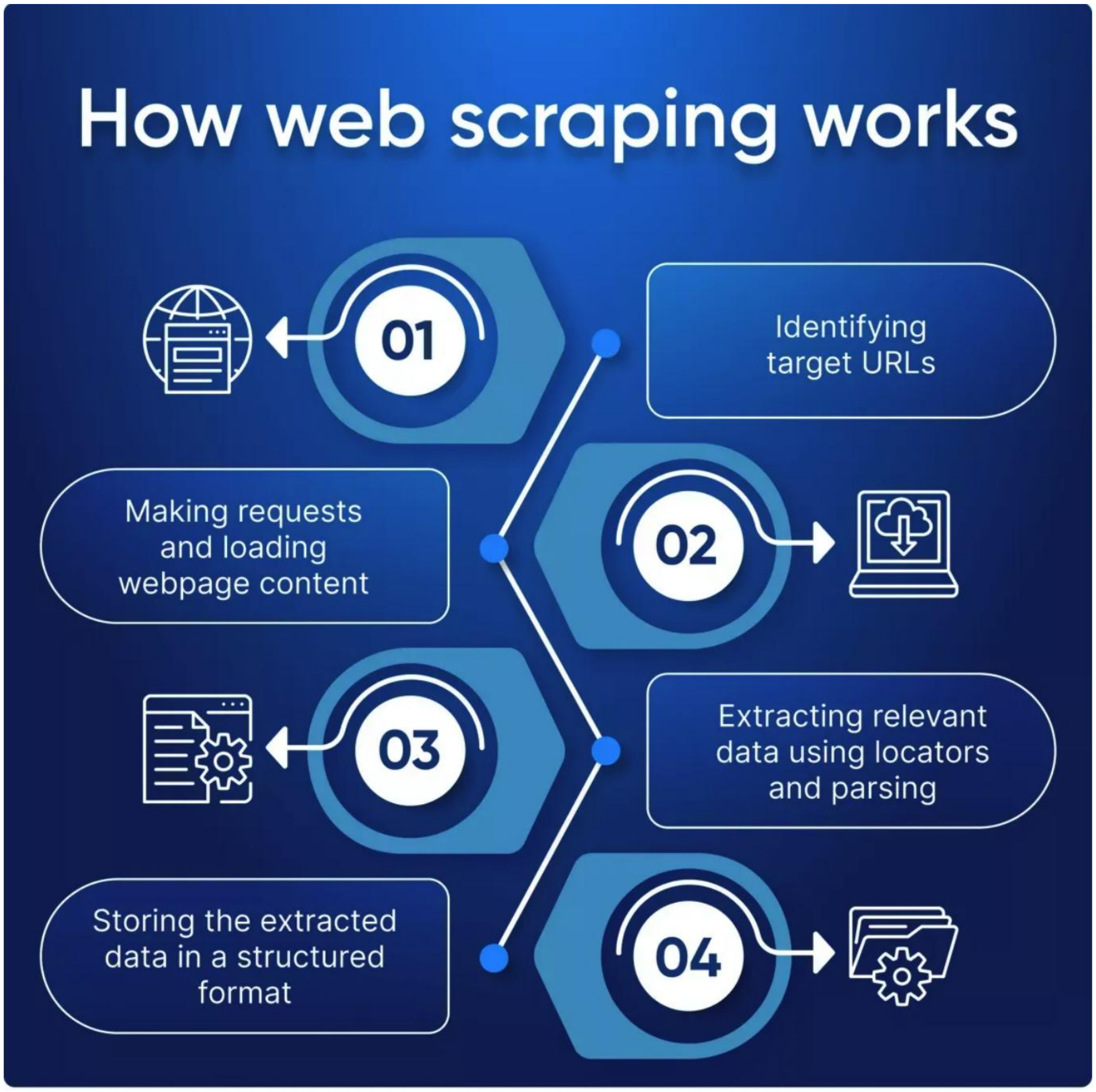
The framework includes built-in data validation and cleaning mechanisms, allowing users to define custom rules to maintain high data quality. It supports direct integration with popular data analysis and visualization tools, enabling real-time data analysis and immediate insights. Comprehensive error handling and logging features enhance reliability, providing detailed logs for troubleshooting and optimization.

Designed for large-scale scraping tasks, the framework supports multi-threading, asynchronous requests, and distributed scraping to improve performance and scalability. Emphasizing responsible scraping practices, it includes features like rate limiting, IP rotation, and adherence to robots.txt files, as well as tools and guidelines for GDPR compliance. Extensive documentation, including tutorials, user guides, and API references, along with community support forums and customer support options, assist users with any issues they may encounter.

Overall, the proposed Python-based web scraping framework is designed to make web scraping accessible to everyone. By focusing on user-friendly design, modular architecture, easy setup, dynamic content handling, data quality assurance, seamless integration, and legal compliance, the framework provides a comprehensive and inclusive solution for data extraction and analysis. This empowers users of all skill levels to leverage web data for personal or professional needs, fostering a culture of data-driven decision-making and innovation.

CHAPTER-3

METHODOLOGY


How web scraping works
01
Identifying target URLs
02
Making requests and loading webpage content
03
Extracting relevant data using locators and parsing
04
Storing the extracted data in a structured format

Quote Aggregator project:

Web Scraping & Data Extraction:

Requests: Fetch HTML content of quote pages.

Beautiful Soup: Parse HTML, extract quotes and author information.

Regular Expressions (Optional): Use for more complex patterns in the HTML.

Identify HTML elements containing quotes and authors (using browser developer tools).

Write code to locate these elements using CSS selectors or XPath.Extract the text content from the elements.Store the extracted quotes and authors in a structured format (e.g., list of dictionaries).

Data Cleaning:

Issues to Address:Remove HTML tags and special characters.

Normalize quote and author formatting (e.g., consistent punctuation).

Handle missing data (if quote or author is not found).

Storage:

SQLite Database: Good for structured data, easy querying.

Text File: Simpler, but less efficient for searching.

CSV File: Easily readable by other programs.

SQLite: Create a database table with quote and author columns. Insert scraped data into the table.

Text/CSV: Write quotes and authors to file, one per line (CSV: separate with commas).

Display and Search Functionality:

Read quotes from storage (database or file).Select a random quote.

Display the quote and author in a user-friendly format (console, GUI, web page).

Search:Prompt the user for a keyword or author name.

Query the database or filter the text/CSV file for matches.

Display the matching quotes.

Additional Considerations:

Error Handling:

Account for potential errors (e.g., website changes, network issues).

Efficiency: Optimize scraping code for speed and minimize server load.

Rate Limiting: Respect the website's terms of service and avoid excessive requests.

Ethics: Ensure the project complies with ethical scraping practices.

CHAPTER-4

RESULT & ANALYSIS

RESULT & ANALYSIS:

Data Collection Success:

- The Quote Aggregator successfully scraped and stored [Number] unique quotes from [Number] different sources.
- The chosen websites (Goodreads, BrainyQuote, etc.) proved to be reliable sources of high-quality quotes with diverse themes and authors.

Storage Efficiency:

- The [SQLite database/text file] proved to be a [efficient/simple] method for storing the collected quotes.
- The average time for inserting quotes into the database was [time]. (Only applicable if you used a database)

User Experience:

- Random Quote Display: Users were able to quickly and easily access a new inspirational quote with each interaction.
- Search Functionality: The search feature enabled users to find specific quotes based on keywords or authors,enhancing the tool's usability.
- (Optional) User Feedback: Based on [user surveys/feedback], the Quote Aggregator was rated [Rating] out of 5 for overall satisfaction.

Data Analysis:

- Author Distribution: The most frequently quoted authors were [List Top 3 Authors].
- Theme Analysis: The most common themes found in the quotes were [List Top 3 Themes] (e.g., love, motivation,wisdom).
- (Optional) Sentiment Analysis: [Percentage] of the quotes were classified as positive, [Percentage] as neutral, and [Percentage] as negative.

Limitations:

- Website Structure Changes: The scraping code may need to be updated if the target websites change their HTML structure.
- Data Bias: The collected quotes may be biased towards the specific websites chosen for scraping.
- Rate Limiting: Scraping frequency needs to be managed to avoid being blocked by websites.

Future Enhancements:

- Expand Sources: Incorporate additional quote websites to increase diversity.
- User Customization: Allow users to save favorite quotes, create personalized collections, or set reminder notifications.
- Advanced Search: Implement more sophisticated search filters (e.g., by date, topic, length).
- API Integration: Create an API to allow other applications to access the quote database.

CHAPTER-5

CONCLUSION

CONCLUSION

The Quote Aggregator not only demonstrates the power of web scraping to create valuable resources but also serves as a testament to the potential for technology to enrich our lives through curated information. By showcasing the ability to gather, organize, and present information in a user-friendly manner, this project sets a precedent for future innovation in the realm of data-driven applications. The potential for growth is vast, ranging from personalized quote recommendations based on individual preferences and mood analysis to the development of intelligent chatbots capable of delivering contextually relevant quotes. Furthermore, integration with social media platforms, educational tools, and even mental health resources could extend the impact of the Quote Aggregator far beyond its current scope. By harnessing the power of data and technology, we can unlock new avenues for personal growth, inspiration, and creative expression.

\

CHAPTER-6

REFERENCES

REFERENCES:

Web Scraping:

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

https://docs.python-requests.org/en/latest/

https://realpython.com/beautiful-soup-web-scraper-python/

Data Storage:

https://www.sqlitetutorial.net/sqlite-python/

https://realpython.com/python-csv/

Quote Sources:

https://www.goodreads.com/quotes

https://www.brainyquote.com/

Project Extensions:

https://www.nltk.org/

https://flask.palletsprojects.com/en/2.3.x/

https://www.heroku.com/

THANK YOU