PROJECT REPORT ON

Data Scraping and Visualisation with Python

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

Batch 2021 – 24

Under the Guidance of	**Submitted by**
Dr. Santosh Kumar Singh	Anmol Sahni
(Associate professor)	01528402021

Bosco Technical Training Society

Don Bosco Technical School, Okhla Road, New Delhi-110025

9643868820, 8527787221, 011-41033889

Affiliated to

Guru Govind Singh Indraprastha University, Delhi

Sector – 16C, Dwarka, New Delhi – 110078 (India)

Phone: +91 11-25302170, Fax: +91-11-25302111

ACKNOWLEDGEMENT

The note starts with thanks to Almighty who actually created this piece of work and helped us when things were not easy for us.

I am very grateful and indebted to our Guide Dr. Santosh Kumar Singh who immensely helped and rendered her/his valuable advice, precious time, knowledge, and relevant information regarding the collection of material. He has been a major source of inspiration throughout the project as he not only guided me throughout this Project Report Data Scraping and Visualization with Python but also encouraged me to solve problems that arose during this report.

His guidance and suggestions about this Project report have really enlightened me. It has been a great help to support to have him around.

And finally, I would like to mention appreciation to our parents and friends who have been instrumental throughout this period by providing unrelenting encouragement.

Name of Students	Enrolment No.	Signature
Parth Verma	01428402021	
Anmol Sahni	01428402021	

SELF CERTIFICATE

This is to certify that the dissertation/project report "Data Scraping and Visualisation with Python" work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelor of Computer Applications under the guidance of. Dr. Santosh Kumar. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Signature of the student

Name of the Student

Roll No.

CERTIFICATE

This is to certify that this project entitled "Data Scraping and Visualisation with Python" submitted in partial fulfilment of the degree of Bachelor of Computer Applications to Dr. Santosh Kumar Singh through Guru Gobind Singh Indraprastha University, done by Mr. Anmol Sahni, Roll No. 01528402021 is an authentic work carried out by him at Bosco Technical Training Society, Don Bosco Technical School under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Signature of the Student

Signature of the Guide

SYNOPSIS

INTRODUCTION

The "Data Scraping and Visualization with Python" project aims to leverage the power of Python programming to extract valuable insights from online job listings website. In an era where information is abundant on the internet, this project focuses on scraping relevant data from the hirist.com website, a platform widely used for job postings. By employing web scraping techniques and utilizing popular Python libraries such as BeautifulSoup, requests, and Pandas, the project automates the retrieval of crucial information such as company names, position titles, skills, locations, experience requirements, and posting dates.

PROBLEM STATEMENT

Lack of Analysis and visualisation when dealing with online social networking sites for working professionals seeking job opportunities. Data scraping and visualization project can solve various problems across different domains such as Market Research and Competitor Analysis.

OBJECTIVES AND SCOPE

The primary objective of a data scraping and visualization project is to extract, process, and analyse data from concerned websites or other data repositories, and then present the insights gained from that data in a visual and meaningful way.

To Develop a python program to take user inputs from user and perform data scraping based on data inputted by the user. Scraping data with optimum speed and visualise the data scraped in the form of insights and visualisation which convey meaningful data.

METHODOLOGY

Data Collection: Gather relevant data from one or more sources to support analysis and decision-making.

Data Processing: Clean, transform, and structure the raw data for analysis.

Data Analysis: Extract meaningful insights, trends, patterns, and correlations from the data.

Data Visualization: Present the analysed data in a visual format that is easy to understand and interpret

Decision Support: Enable users to make informed decisions based on the visualized data and insights.

Automation: Streamline the process of data collection, analysis, and visualization through automation, where applicable.

HARDWARE AND SOFTWARE USED

Hardware Requirements:

- RAM: 8GB
- Storage: 200GB
- Internet Speed: over 50mbps (200mbps or more preferred)

Software Requirements:

- Operating system platform, Windows 10 and above
- Python 3.12.0

- Scraping libraries - Beautiful Soup, Requests
- Python Libraries: Plotly, RE, Datetime, Folium, OS
- Data analysis tools: Matplotlib, Seaborn, Pandas, NumPy,
- Framework Support: Django
- IDEs: Python IDE, Microsoft Visual Studio code
- Chromium based browser

TESTING METHODOLOGY

1. **Web Scraping:**

   Input Validation:

   - Enter various primary skills in the input field (e.g., "python developer," "data analyst," etc.). Verify that the application handles invalid input gracefully (e.g., non-alphabetic characters, empty input).

   Data Retrieval:

   - Ensure that the application successfully retrieves data from the hirist.com website. Check if the application handles network errors or invalid URLs appropriately.

2. **Data Processing:**

   CSV File Creation:

   - Confirm that the CSV file is created with the expected name ("file.csv"). Verify the correctness of the CSV file by manually inspecting its contents. Check if the application gracefully handles errors during file creation.

3. **Data Visualization:**

   Chart Generation:

   - Confirm that the application generates all types of Plotly Express charts. Check if the charts are displayed correctly on the webpage. Ensure that the charts update dynamically based on the selected primary skill.

   Chart Content:

   - Manually inspect the generated charts to verify the accuracy of the displayed information. Confirm that the charts exclude the primary skill being analyzed (as specified in the code). Verify that the charts have appropriate labels and legends.

4. **User Interface:**

   Navigation:

   - Ensure that the navigation between different pages and functionalities (e.g., home, chart) works seamlessly.

   User Feedback:

   - Check if the application provides clear feedback to users during and after data scraping and chart generation. Verify that error messages, if any, are informative and user-friendly.

5. **Edge Cases:**

   Empty Data Handling:

- Test the application with primary skills that yield no job listings. Verify that the application handles empty datasets gracefully and provides appropriate feedback.

Performance Testing:

- Test the application with a primary skill that has a large number of job listings to assess performance.

6. **Compatibility:**

   Browser Compatibility:

   - Test the application on different web browsers (e.g., Chrome, Firefox, Safari) to ensure compatibility.

   Operating System:

   - Verify that the application functions correctly on different operating systems (e.g., Windows, macOS, Linux).

7. **Error Handling:**

   Input Errors:

   - Intentionally introduce errors in the code or data files and verify that the application handles these errors gracefully.

8. **Security:**

   Input Sanitization:

   - Attempt to inject malicious input or special characters and ensure that the application sanitizes input to prevent security vulnerabilities.

Completion Criteria: The testing process is considered complete when all identified test cases have been executed, and no further defects or bugs are found.

CONCLUSIONS

In conclusion, the data scraping and visualization project aims to empower users with valuable insights from diverse data sources. By following systematic manual testing methodologies, we can ensure the accuracy, functionality, and user-friendliness of the application and visualizations. With rigorous testing and continuous improvement, the project not only meets its objectives but also provides a powerful tool for informed decision-making, efficient data analysis, and enhanced user experiences.

INDEX

CHAPTER 1 - OBJECTIVE &

SCOPE OF THE PROJECT

1.1 OBJECTIVE

The primary objective of the Data Scraping and Visualisation with Python is to automate the extraction and organization of key information from job postings available on the "https://www.hirist.com" website. By employing web scraping techniques, the script aims to systematically retrieve, process, and present relevant details from the HTML source code of the specified URL. The extracted data includes essential elements such as company names, job position titles, locations, programming languages, required experience levels, mandatory skills, and the creation dates of the job postings.

1.2 SCOPE OF THE PROJECT

Web Scraping:

Utilizes the requests library to fetch the HTML content from the specified URL. Applies the BeautifulSoup library for HTML parsing, creating a navigable tree structure for data extraction.

Data Extraction:

Extracts specific information from the HTML content, including:

- Company names
- Job position titles
- Job locations

- Required programming languages
- Required experience levels
- Mandatory skills for each job posting
- Posting creation dates

Data Transformation:

Processes and refines the extracted data, such as splitting job titles to isolate the actual position titles and formatting dates.

Data Presentation:

Organizes the extracted information into a structured format. Prints the information in tabular form, presenting each job posting's details in a clear and readable manner.

File Operations:

Reads the HTML content from a local file (page_source.txt) instead of making a new web request each time.

Example Usage:

The script showcases the use of regular expressions to extract specific details from the HTML content. This includes parsing company names, job titles, and creation dates, demonstrating the flexibility of the script in adapting to variations in the HTML structure.

Integration Flexibility:

While serving as a standalone tool for job posting data extraction, the script is designed to be adaptable and integrable into larger projects or workflows. Its modular structure allows users to incorporate the script into diverse applications for further analysis or processing of job-related information.

CHAPTER 2 - THEORETICAL BACKGROUND

DEFINITION OF PROBLEM

The "Data Scraping and Visualization with Python" project addresses the intersection of web scraping, data manipulation, and visualization within the context of job market analysis. Theoretical foundations for understanding the problem include:

2.1 Web Scraping:

Web scraping involves the automated extraction of data from websites, typically for the purpose of collecting information or performing data analysis. The project employs libraries such as BeautifulSoup and requests to navigate and extract relevant content from the hirist.com website. Theoretical concepts include HTML parsing, HTTP requests, and DOM (Document Object Model) traversal.

2.2 Data Extraction and Processing:

The project focuses on extracting specific information from job listings, such as company names, position titles, skills, locations, experience requirements, and posting dates. Theoretical understanding of regular expressions is crucial for efficiently parsing and extracting structured data from the raw HTML content. Additionally, Pandas is used for organizing and processing the extracted data in tabular form.

2.3 Data Visualization:

Data visualization is the graphical representation of data to facilitate the interpretation of patterns, trends, and insights. In this project, Plotly Express is employed for creating interactive and visually engaging charts. Theoretical concepts include bar charts for categorical data analysis and line charts for temporal trends. Visualization is a powerful tool for conveying complex information in a comprehensible manner.

Primary and Secondary Skills Analysis:

The project's objective is to analyse job listings based on primary skills (e.g., "python developer") and explore associated secondary skills. Theoretical understanding of data categorization and counting is applied to create a bar chart illustrating the frequency of secondary skills. This analysis contributes to understanding the skill sets demanded by the job market for a given primary skill.

Temporal Analysis of Job Postings:

The project includes a temporal analysis of job postings over different months, represented through a line chart. Theoretical concepts involve date manipulation and visualization to uncover trends in job postings, allowing users to discern patterns related to hiring demand and seasonal variations.

2.4 User Input Handling and Error Management:

Theoretical principles of user input validation and error handling are applied to ensure the robustness of the application. This includes checking for valid primary skills, handling network errors during web scraping, and providing informative feedback to users.

Ethical Considerations:

Web scraping raises ethical considerations, and theoretical awareness of responsible scraping practices is crucial. Adherence to website terms of service, respecting robots.txt files, and avoiding excessive requests are among the ethical considerations explored in this project.

CHAPTER 3 - SYSTEM ANALYSIS & DESIGN VIS-A-VIS USER REQUIREMENTS

3.1 INTRODUCTION

The "Job Market Analysis Tool" is designed to fulfill the user requirements outlined in the Software Requirements Specification (SRS). This section provides an overview of how the system analysis and design align with the user requirements.

3.2 SYSTEM ANALYSIS

3.2.1 Overview

The system analysis phase focused on understanding the user requirements and translating them into functional specifications. The analysis identified key components, such as web scraping, data processing, and visualization, to meet the primary objectives of the tool.

3.2.2 Functional Components

1. **Web Scraping Module:**
    - Extracts job listings from hirist.com.
    - Handles network errors and invalid URLs using the **requests** library.
    - Utilizes BeautifulSoup for parsing HTML content.
2. **Data Processing Module:**
    - Extracts relevant information from job listings, including company names, positions, skills, locations, experience requirements, and posting dates.

- Organizes extracted data into a CSV file ("file.csv").
- Categorizes primary and secondary skills for analysis.

3. **Data Visualization Module:**
    - Generates a bar chart illustrating the frequency of secondary skills associated with a given primary skill using Plotly Express (**px**).
    - Produces a line chart depicting temporal trends in job postings over different months.
4. **User Interface:**
    - Provides a user-friendly interface for users to input primary skills.
    - Displays visualizations and relevant information.
    - Offers navigation between different functionalities (home, chart).

3.2.3 Non-Functional Components

1. **Performance:**
    - Efficiently handles variations in job listing data volume.
    - Provides a responsive user experience.
2. **Security:**
    - Adheres to website terms of service during web scraping.
    - Ensures secure handling of user data and input.
3. **Usability:**
    - Ensures the user interface is intuitive and user-friendly.
    - Compatible with major web browsers (Chrome, Firefox, Safari) and different operating systems (Windows, macOS, Linux).
4. **Documentation:**
    - Provides a user manual for guidance.

- Offers technical documentation for system architecture, algorithms, and libraries used.

5. **Error Handling:**
    - Provides clear feedback to users during and after data scraping and chart generation.
    - Presents informative and user-friendly error messages.

3.3 SYSTEM DESIGN

3.3.1 Architecture

The system follows a client-server architecture where the client (user interface) interacts with the server-side components responsible for web scraping, data processing, and visualization.

3.3.2 Technologies Used

1. **Web Scraping:**
    - **requests** library for fetching job listings.
    - **BeautifulSoup** for parsing HTML content.
2. **Data Processing:**
    - **Pandas** library for organizing extracted data into CSV files.
3. **Data Visualization:**
    - **plotly.express** for generating interactive charts.
4. **User Interface:**
    - Developed using Django templates for rendering HTML.

3.3.3 Database

The system does not employ a traditional database, as it relies on CSV files for data storage. This decision simplifies the architecture and allows for easy data retrieval and manipulation using pandas.

3.4 USER REQUIREMENTS FULFILLMENT

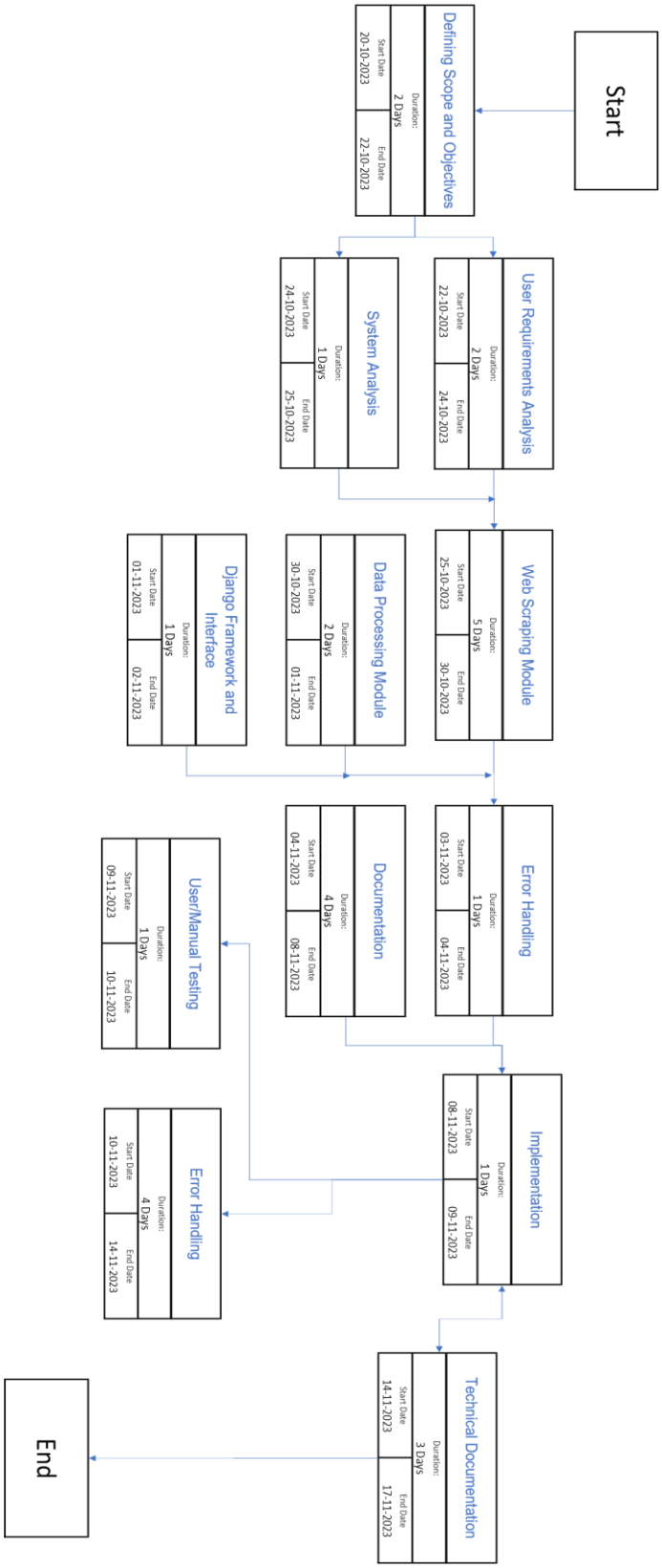
The system has been designed to address each user requirement outlined in the Software Requirements Specification. Key features, such as web scraping, data processing, and visualization, have been implemented to meet the objectives of analyzing job market trends based on primary skills.

3.5 CONCLUSION

The system analysis and design have successfully translated user requirements into functional and non-functional components. The architecture, technologies used, and data handling strategies have been carefully considered to ensure the development of a robust and user-friendly "Job Market Analysis Tool." The implementation phase will involve coding the outlined design and deploying the system for user access.

This document provides a comprehensive overview of how the system analysis and design align with the specified user requirements, ensuring the successful development of the intended tool.

CHAPTER 4 - SYSTEM PLANNING (PERT CHART)



CHAPTER 5 - METHODOLOGY ADOPTED, SYSTEM IMPLEMENTATION & DETAILS OF HARDWARE & SOFTWARE USED SYSTEM MAINTENANCE & EVALUATION

5.1 METHODOLOGY

The development of the "Job Market Analysis Tool" follows an iterative and incremental methodology, blending elements of the Agile and Waterfall methodologies to ensure a flexible yet structured approach to project execution. The chosen methodology aims to deliver a high-quality and well-tested software product that meets user requirements effectively.

5.1.1 Agile Principles

The project incorporates Agile principles to enhance collaboration, adaptability, and responsiveness to change. Key Agile practices include:

Scrum Framework: The project is organized into sprints, each typically lasting two weeks. Regular sprint planning, daily stand-ups, and sprint reviews facilitate continuous improvement and adaptability.

User Involvement: Close collaboration with end-users and stakeholders ensures that their feedback is incorporated throughout the development process, leading to a solution that aligns closely with user expectations.

Iterative Development: The project follows an iterative development approach, allowing for continuous refinement of features and functionalities based on ongoing feedback and testing.

5.1.2 Waterfall Phases

While Agile principles guide the development process, the project adheres to certain aspects of the Waterfall model to maintain a structured and well-defined framework:

Requirements Analysis: A comprehensive analysis of user requirements is conducted at the beginning of the project to establish a clear understanding of the scope and objectives.

Design: Following the requirements analysis, a detailed system analysis and design phase is undertaken to define the architecture, components, and technologies to be used in the system.

Implementation: The development phase involves the implementation of distinct modules, including web scraping, data processing, data visualization, and user interface components, following the principles of modular and maintainable coding practices.

Testing: Rigorous testing is conducted at various levels, including unit testing, integration testing, and user acceptance testing (UAT), to ensure the reliability and effectiveness of the developed system.

Deployment: Once all modules have been thoroughly tested and approved, the deployment phase is initiated, making the system available for end-users.

5.1.3 Continuous Integration and Continuous Deployment (CI/CD)

To streamline the development and deployment process, the project incorporates CI/CD practices. Automated testing, version control, and continuous integration help maintain a stable

codebase, while continuous deployment ensures that updates and improvements are efficiently delivered to end-users.

By combining Agile flexibility with Waterfall structure, the methodology adopted for this project aims to deliver a robust, user-friendly, and scalable solution for analyzing job market trends.

5.2 Hardware Requirements

The implementation of the "Job Market Analysis Tool" necessitates the use of hardware that supports the efficient execution of the developed software components. The key hardware requirements include:

5.2.1 Server Infrastructure:

- Operating System: Linux-based/ Windows server environment.
- Processor: Multi-core processors to handle concurrent tasks.
- RAM: (8gb) Sufficient RAM for handling large datasets and simultaneous user requests.
- Storage: (200gb) Adequate storage capacity for storing scraped data, processed data, and system logs.

5.2.2 Client Devices:

- Desktops/Laptops: Standard configurations to support web-based user interfaces.
- Web Browsers: Compatible web browsers such as Chrome, Firefox, or Safari

5.3 Software Requirements

The system relies on various software components to facilitate web scraping, data processing, data visualization, and user interface development. The key software requirements include:

Web Scraping:

- Python: Core programming language for web scraping scripts.
- Requests: Library for making HTTP requests to retrieve web pages.
- Beautiful Soup: Library for parsing HTML and extracting relevant information.

Data Processing:

- Python: Continues to be the primary language for data processing.
- Pandas: Data manipulation and analysis library.
- NumPy: Scientific computing library for numerical operations.
- Scikit-Learn: Machine learning library for data analysis.

Data Visualization:

- Plotly Express: Library for creating interactive and visually appealing charts.
- Matplotlib: Comprehensive library for static, animated, and interactive visualizations.

User Interface:

- Django Framework: High-level Python web framework for rapid development.
- HTML, CSS, JavaScript: Standard web development technologies for creating user interfaces.

- Bootstrap: Front-end framework for designing responsive and modern UIs.

Version Control:

- Git: Distributed version control system for tracking changes in the codebase.

5.4 SYSTEM MAINTENANCE

Codebase Management

The project employs version control with Git to manage the codebase. The code is hosted on a Git repository (e.g., GitHub) to facilitate collaboration and ensure a consistent and stable codebase. Regular code reviews and pull requests are conducted to maintain code quality.

Continuous Integration and Continuous Deployment (CI/CD)

CI/CD practices are implemented to automate testing and deployment processes. Automated testing ensures that changes do not introduce regressions, and continuous deployment streamlines the release of new features and improvements.

5.5 DATA MAINTENANCE

The system periodically updates its dataset through scheduled web scraping tasks. Data cleaning and processing routines are applied to maintain data accuracy and integrity. Logs and error reports are monitored to address any issues promptly.

5.6 SYSTEM EVALUATION

System evaluation involves assessing the effectiveness, performance, and user satisfaction with the deployed "Job Market Analysis Tool." Key evaluation criteria include:

1. Performance Metrics:

- Response Time: Measure the time taken for the system to respond to user requests.
- Scalability: Assess the system's ability to handle increasing user loads.

2. User Satisfaction:

- Surveys and Feedback: Collect user feedback through surveys to understand user satisfaction and identify areas for improvement.

Bug Tracking:

- Issue Tracking System: Utilize an issue tracking system (e.g., Jira) to log and address reported bugs and errors.

Data Accuracy:

- Data Validation: Implement validation checks to ensure the accuracy of scraped and processed data.

Security:

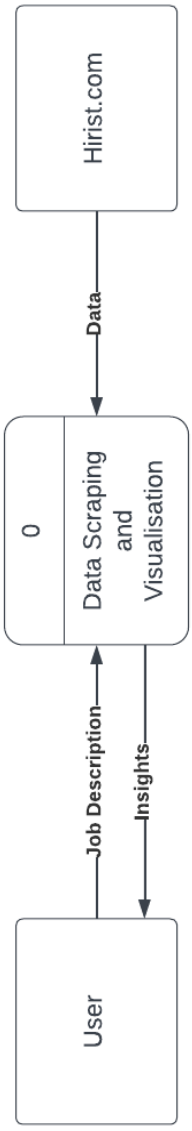
- Security Audits: Regularly conduct security audits to identify and address potential vulnerabilities.

The system evaluation process is ongoing, with feedback and performance data informing iterative improvements and updates to ensure the tool remains effective and meets user expectations.

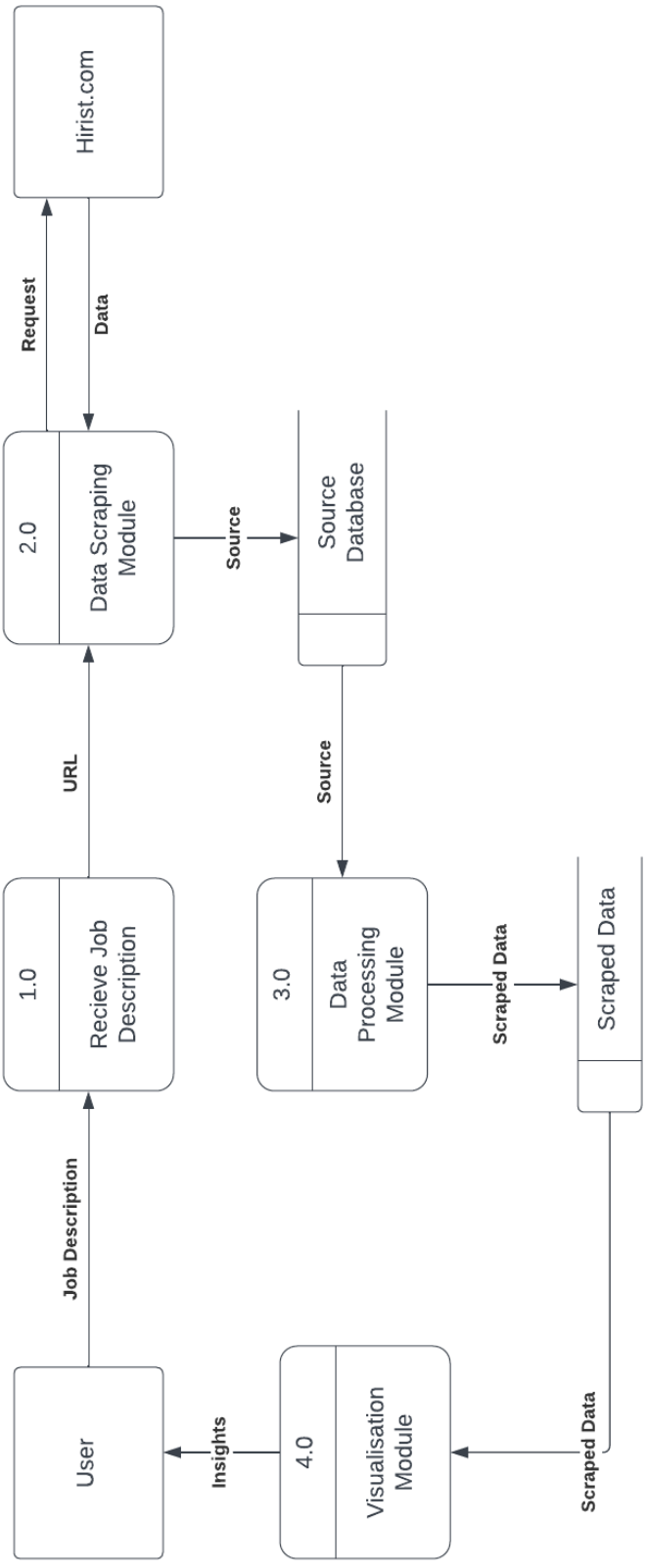
CHAPTER 6 - DETAILED LIFE CYCLE OF THE PROJECT

a. ERD, DFD

0 Level DFD

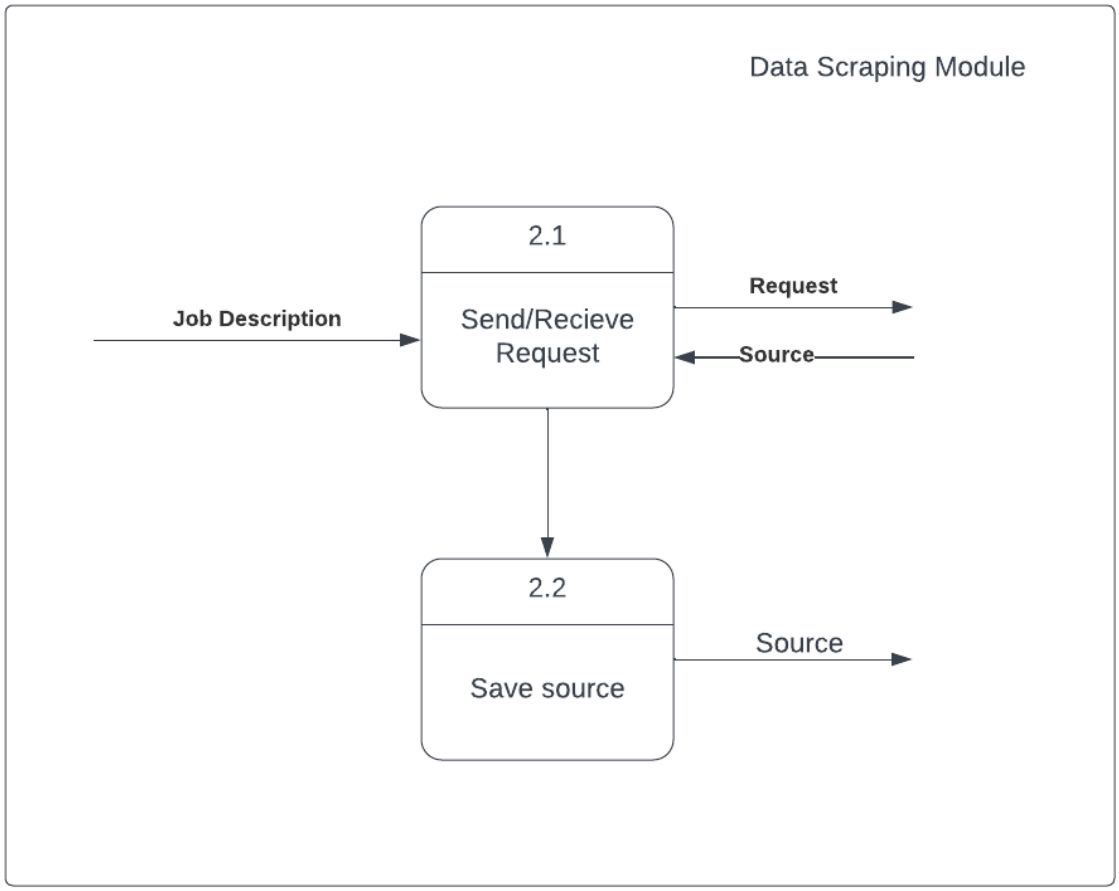


1 Level DFD

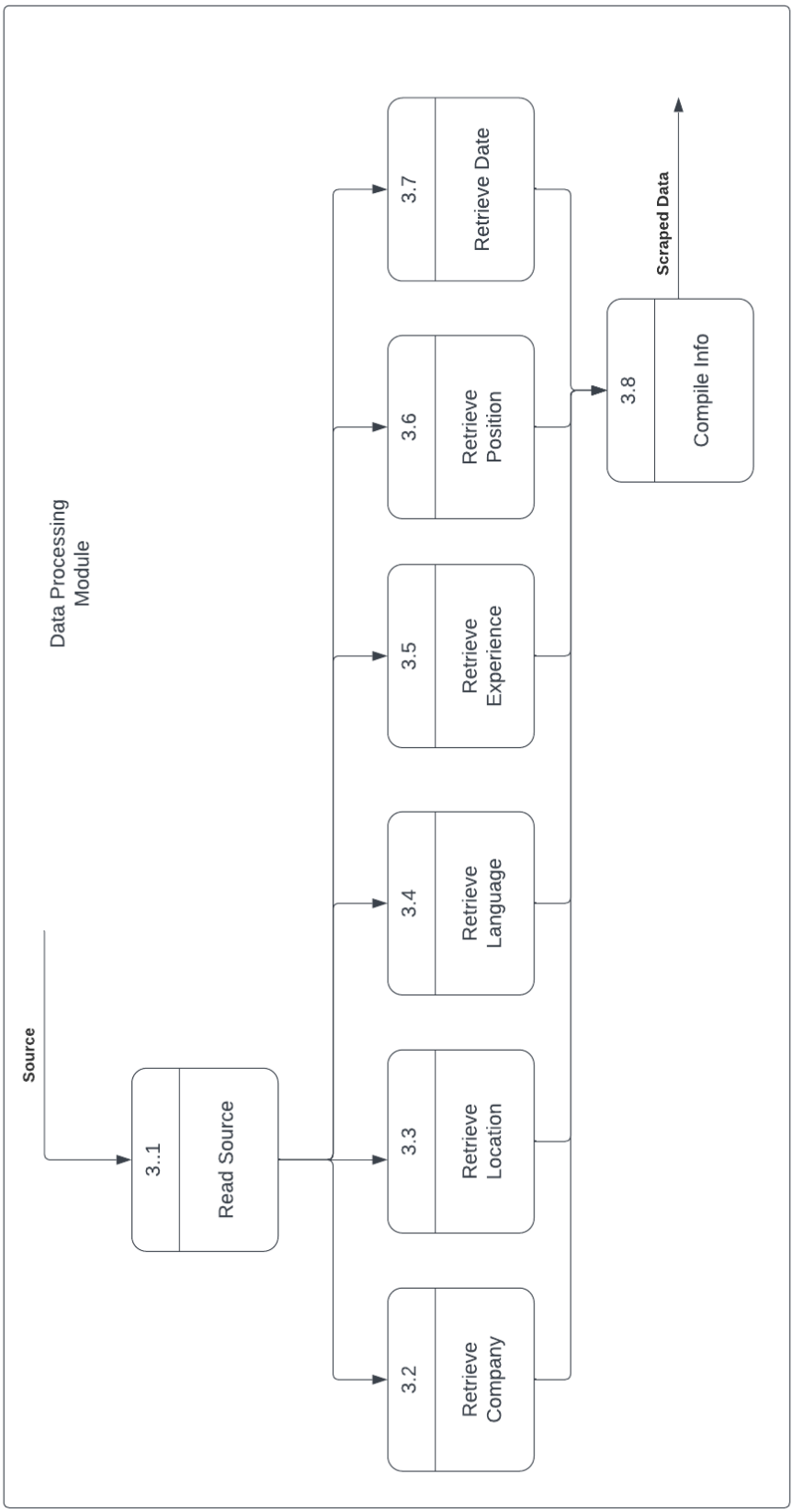

Hirist.com
Request
Data
2.0
Data Scraping Module
Source
Source Database
URL
Source
1.0
Recieve Job Description
3.0
Data Processing Module
Scraped Data
Scraped Data
Job Description
User
Insights
4.0
Visualisation Module
Scraped Data

2 Level DFD

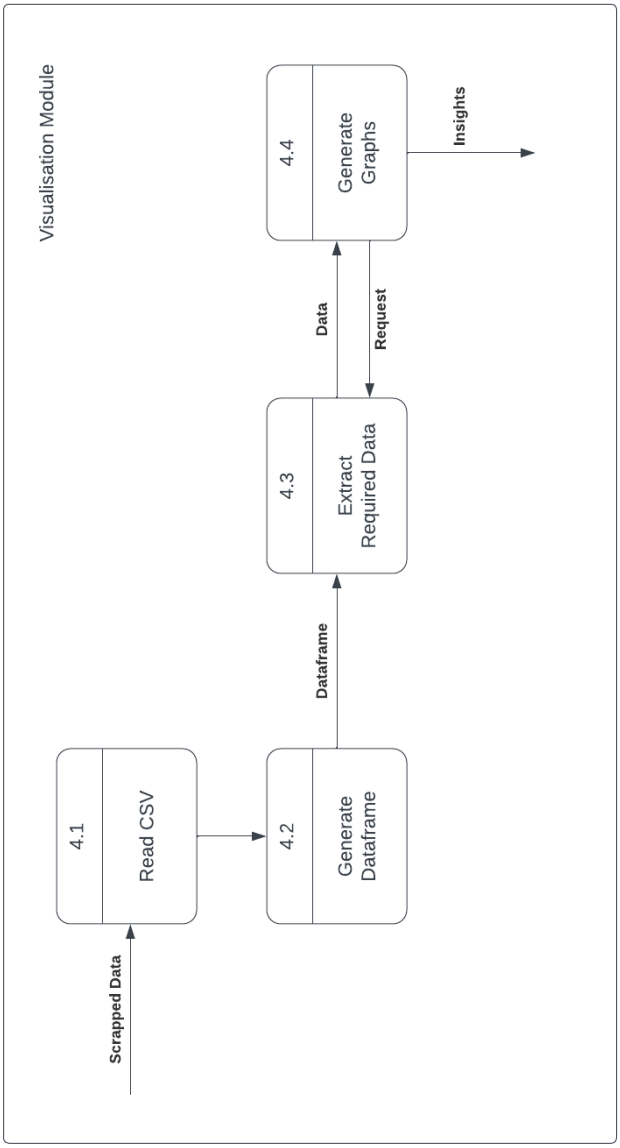
Data Scraping Module:



Data Processing Module



Visualisation Module



b. Input and output screen design

User Interface Overview

The user interface of the "Job Market Analysis Tool" is designed to provide an intuitive and interactive experience for users. The tool primarily consists of input screens for initiating job searches and output screens for presenting visualizations and insights based on the scraped data. Below are detailed descriptions of the key input and output screens.

Input Screens

1. Search Input Form

Description:

- **Purpose:** Allows users to specify search parameters for job market analysis.
- **Components:**
    - **Search Bar:** Input field for entering job-related keywords.
    - **Additional Filters:** Options to filter by location, experience level, or specific skills.
    - **Submit Button:** Initiates the search process based on the provided parameters.

2. Data Source Configuration

Description:

- **Purpose:** Enables users to configure the data sources for the job market analysis.
- **Components:**
    - **Source Selection:** Dropdown menu for selecting job platforms or websites.

- **Advanced Options:** Additional settings for specifying the depth of the search or specific sources.

Output Screens

1. Job Market Overview

Description:

- **Purpose:** Provides an overview of the job market based on the user's search parameters.
- **Components:**
  - **Graphs and Charts:** Visual representations of job distribution by location, skills demand, and experience levels.
  - **Summary Statistics:** Key metrics such as the total number of jobs, top hiring companies, and popular skills.

2. Skills Analysis Dashboard

Description:

- **Purpose:** Offers a detailed analysis of the skills required in the job market.
- **Components:**
  - **Skill Cloud:** A visual representation of the most in-demand skills.
  - **Skill Distribution Chart:** Bar chart showcasing the frequency of each skill in job listings.

3. Temporal Analysis

Description:

- **Purpose:** Presents a temporal analysis of job postings over time.
- **Components:**
    - **Line Chart:** Displays the number of job postings per month.
    - **Date Range Selector:** Allows users to focus on a specific time frame.

Navigation and Interaction

- **Navigation Bar:** Consistent navigation bar across screens for easy access to different sections.
- **Interactive Elements:** Users can hover over charts for additional information and click on data points for drill-down details.
- **Export and Share:** Options for users to export visualizations or share insights through various channels.

These input and output screen designs aim to streamline the user experience, making it simple for users to input their preferences and interpret the generated insights effectively. The visualizations are designed to be clear, informative, and interactive, allowing users to gain valuable insights into the job market based on their queries.

c. PROCESS INVOLVED

The "Job Market Analysis Tool" involves a series of processes that collectively enable users to search, scrape, process, and visualize job market data. The following sections provide an in-depth overview of the key processes involved in the system.

1. Search Process

Purpose:

Initiate a job market analysis based on user-defined search parameters.

Steps:

1. **User Input:**
    - Users enter job-related keywords, location preferences, experience level, and additional filters.
2. **Form Submission:**
    - Users submit the search form, triggering the system to initiate the data collection process.

Output:

- The search process outputs a set of parameters that serve as input for the web scraping process.

2. Web Scraping Process

Purpose:

Retrieve relevant job data from specified sources on the internet.

Steps:

1. **Source Configuration:**
    - The system uses the user-defined parameters to configure the sources for web scraping.
2. **Scraping Execution:**
    - Automated scripts make HTTP requests to job platforms, extracting HTML content.
    - BeautifulSoup is employed to parse the HTML and extract specific job details.

Output:

- The web scraping process generates raw data in the form of HTML content from job listings.

3. Data Processing

Purpose:

Cleanse and structure the scraped data for further analysis.

Steps:

1. **Data Cleaning:**
    - Remove irrelevant or duplicate entries from the scraped data.
    - Address inconsistencies in job titles, company names, and other attributes.
2. **Data Transformation:**

- Extract relevant information such as company names, job titles, skills, and locations.
- Format dates and other temporal data.

Output:

- Processed data in a structured format, suitable for analysis and visualization.

4. Data Analysis and Visualization

Purpose:

Present insights derived from the processed data through interactive visualizations.

Steps:

1. **Data Loading:**
    - Load the processed data into a Pandas DataFrame for analysis.
2. **Visualization Creation:**
    - Utilize Plotly Express and Matplotlib to create interactive charts and graphs.
    - Generate visualizations such as bar charts, line charts, and skill clouds.

Output:

- Visualizations depicting job market trends, skill distributions, and temporal analysis.

5. User Interaction

Purpose:

Facilitate user interaction with the visualizations for deeper insights.

Steps:

1. **Interactive Elements:**
    - Implement hover-over tooltips and click events for interactive charts.
    - Provide date range selectors for temporal analysis.
2. **Export Options:**
    - Allow users to export visualizations in various formats (e.g., PNG, PDF).
    - Enable sharing options for collaborative analysis.

Output:

- Enhanced user experience with interactive features and export functionalities.

6. System Maintenance and Updates

Purpose:

Ensure the system remains robust and up-to-date.

Steps:

1. **Periodic Data Updates:**
    - Schedule regular web scraping tasks to update the dataset.
    - Apply data cleaning and processing routines to maintain data accuracy.
2. **Codebase Management:**
    - Conduct regular code reviews and implement improvements.
    - Utilize version control (Git) for codebase management.

Output:

- A well-maintained system with up-to-date job market data and improved functionalities.

The collective execution of these processes empowers the "Job Market Analysis Tool" to provide users with valuable insights into the dynamic landscape of job opportunities, skill requirements, and temporal trends in the market.

d. METHODOLOGY USED TESTING

Methodology Adopted

The development and testing of the "Job Market Analysis Tool" followed an agile methodology with a focus on iterative development, collaboration, and continuous feedback. The agile methodology allowed for flexibility in responding to changing requirements and incorporating user feedback throughout the development lifecycle.

Agile Development Process

1. **Requirements Gathering:**
    - Collaborative sessions with stakeholders to gather and refine user requirements.
    - Iterative discussions to prioritize features and functionalities.
2. **Sprint Planning:**
    - Divided development into sprints, each focusing on specific features or enhancements.
    - Regular sprint planning meetings to define goals for each iteration.
3. **Continuous Integration and Deployment:**
    - Utilized continuous integration practices to ensure early detection of code issues.
    - Automated deployment pipelines for seamless delivery of new features.
4. **User Feedback Loops:**
    - Regularly incorporated user feedback obtained through testing and usage.
    - Adjustments made based on feedback to enhance user experience and address issues.

Testing Methodology

The testing approach for the "Job Market Analysis Tool" primarily involves user testing and manual testing by developers. The emphasis is on ensuring the tool's usability, functionality, and reliability in real-world scenarios.

User Testing

1. **Alpha Testing:**
    - Internal testing by the development team to identify and rectify issues before releasing to a wider audience.
    - Emphasis on uncovering any bugs, glitches, or usability challenges.
2. **Beta Testing:**
    - Limited release to a select group of external users to gather feedback.
    - Users explore the tool independently, providing insights into usability and potential enhancements.
3. **User Feedback Analysis:**
    - Systematic analysis of user feedback to identify common issues or areas of improvement.
    - Prioritization of feedback based on impact and relevance.

Manual Testing by Developers

1. **Unit Testing:**
    - Developers conducted unit tests to ensure individual functions and modules work as intended.
    - Addressed any coding errors or logic issues during this phase.
2. **Integration Testing:**
    - Ensured seamless integration between different components of the system.

- Checked for data consistency and proper communication between modules.

3. **System Testing:**
    - Comprehensive testing of the entire system to verify that it meets the specified requirements.
    - Identified and resolved any compatibility issues or performance bottlenecks.
4. **User Acceptance Testing (UAT):**
    - Developers performed UAT to simulate user scenarios and validate the tool's overall functionality.
    - Adjustments made based on UAT findings to align with user expectations.

Conclusion

The agile development methodology, coupled with user testing and manual testing by developers, has been integral to the success of the "Job Market Analysis Tool." Continuous collaboration, frequent testing cycles, and an iterative approach have contributed to the creation of a robust, user-friendly tool that aligns with user needs and expectations. Ongoing testing and feedback mechanisms will be maintained to address future enhancements and ensure the tool's reliability.

CHAPTER 7 - CODING

AND SCREENSHOTS OF THE PROJECT

Views.py:

```
from django.shortcuts import render

import plotly.express as px

import requests

from bs4 import BeautifulSoup

import re

from datetime import datetime

import csv

import pandas as pd

from collections import Counter

import folium

import os



def get_page_source(url):

    try:

        response = requests.get(url)

        response.raise_for_status()
```

```
        soup = BeautifulSoup(response.content, 'html.parser')
        page_source = str(soup)
        return page_source

    except requests.exceptions.RequestException as e:
        print(f"Error: {e}")
        return None

def save_to_file(page_source, filename):
    try:
        with open(filename, 'w', encoding='utf-8') as file:
            file.write(page_source)

        print(f"Page source saved to {filename}")

    except Exception as e:
        print(f"Error saving to file: {e}")

def get_companyName(file_content):

    pattern = r'"companyName":"([^"]+)"'
```

```
    matches = re.findall(pattern, file_content)
    return matches

def get_Location(file_content):

    pattern = r'"location":\[\{"id":\d+,"name":"([^"]+)"\}\]'

    matches = re.findall(pattern, file_content)
    return matches

def get_language(file_content):

    pattern = r'"mandatoryTags":\[\{"id":\d+,"name":"([^"]+)"'

    matches = re.findall(pattern, file_content)

    return matches

def get_positionTitle(file_content):
    pattern = r'"title":"(.*?)"'
    new_match = []
```

```
    matches = re.findall(pattern, file_content)
    for match in matches:
        print(match)
        nm = match.split(" - ")
        if len(nm)>1:
            # print(nm[-2])
            new_match.append(nm[-2])
        else:
            new_match.append("null")
    return new_match


def get_experience(file_content):
    pattern = r'\((\d+-\d+) yrs\)'
    matches = re.findall(pattern, file_content)
    return matches


def get_allMandatorySkills(file_content):
    pattern = r'"title":"(.*?)".*?"mandatoryTags":\[(.*?)\]'

    skill_list = []

    matches = re.findall(pattern, file_content, re.DOTALL)
```

```
    for name in matches:
        mandatory_tags_part = name[1]
        skills = re.findall(r'"name":"(.*?)"', mandatory_tags_part)
        skill_list.append(skills)

    return skill_list

def get_Date(file_content):
    pattern = r'"createdTimeMs":(\d+)'
    timestamps = re.findall(pattern, file_content)
    matches = timestamps

    final_date = []
    # matches = [datetime.utcfromtimestamp(int(ts) / 1000.0) for ts in
timestamps]
    for timestamp in matches:
        full_date = datetime.utcfromtimestamp(int(timestamp) / 1000.0)
        date = full_date.date()
        formatted_date = date.strftime("%d-%m-%Y")
        # print(formatted_date)
        final_date.append(formatted_date)
```

```
    return final_date


def write_CSV(list,csv_path):

    header_row = ["Comapay","Position", "Primary Skill", "Location",
"Experience","Other Skills","Date"]

    with open(csv_path, 'w', newline='') as csv_file:

        csv_writer = csv.writer(csv_file)

        csv_writer.writerow(header_row)



        for row in list:

            csv_writer.writerow(row)

    print("Data has been written to"+str(csv_path))


#-------------------------------------------------------------------------
-----------------


def scrapTo_csv(page_source):




    main_list = list(zip(

        get_companyName(page_source),

        get_positionTitle(page_source),

        get_language(page_source),
```

```
        get_Location(page_source),
        get_experience(page_source),
        get_allMandatorySkills(page_source),
        get_Date(page_source)
        ))

    write_CSV(main_list,"file.csv")



# with open(file_path, 'r') as file:
#     page_source = file.read()

#_____________________________________________________________________________
_________________________

def home(request):

    # os.remove("file.csv")
    # os.remove("page_source.txt")
    # try:
    #     df.drop(axis=0, inplace=True)
    # except:
```

```
    #     pass


    return render(request, 'base.html')


def about(request):

    return render(request,'about.html')


def inst(request):

    return render(request,'instructions.html')


def dev(request):

    return render(request,'dev.html')


def add(request):

    #Getting the variable

    a = request.GET['n1']

    url       =       (get_page_source("https://www.hirist.com/search/"+a+"-
p"+str(1)+".html"))

    save_to_file(url, "pagesource.txt")



    for i in range(2,20):
```

```
        source    =    (get_page_source("https://www.hirist.com/search/"+a+"-
p"+str(i)+".html"))



        if "title" in source:

            try:

                with open("pagesource.txt", 'a', encoding='utf-8') as file:

                    file.write(source)



                print("Page source saved to pagesource.txt again")



            except Exception as e:

                print(f"Error saving to file: {e}")



    with open("pagesource.txt", 'r', encoding="utf8") as file:

        page_source = file.read()




    # url = "https://www.hirist.com/search/"+str(a)+".html"

    scrapTo_csv(page_source)



#Graph1-------------------------------------------------------------
```

```
x_axis = []

y_axis = []

df = pd.read_csv("file.csv")



column_list = df["Primary Skill"].tolist()

counted_elements = Counter(column_list)



for element,count in counted_elements.items():

    print(element," : ",count)

    x_axis.append(element)

    y_axis.append(count)



if a in x_axis:

    pos = x_axis.index(a)

    print(a," is on ", pos)

    x_axis.pop(pos)

    y_axis.pop(pos)

else:

    pass

    print(a," is not")
```

```
    print(x_axis)
    print(y_axis)



    fig = px.bar(
        x=x_axis,
        y=y_axis,
        title="Secondary Skills",
        labels={'x': 'Skills', 'y': 'Number of Jobs'}
    )
#Graph2-----------------------------------------------------------------
    dates = []
    x_axis = []
    y_axis = []



    column_list = df["Date"].tolist()


    for i in column_list:
        print(i)
```

```
    i = datetime.strptime(i, "%d-%m-%Y")
    i = i.strftime("%m-%Y")

    dates.append(i)

date_counts = Counter(dates)
for date, count in date_counts.items():
    print(f"{date}: {count} occurrences")
    x_axis.append(date)
    y_axis.append(count)

print(x_axis)
print(y_axis)


fig2 = px.line(
    x=x_axis,
    y=y_axis,
    title="Jobs Posted by Month",
    labels={'x': 'Jobs', 'y': 'No. of postings'}
)
```

```
fig.update_layout(
    title={
        'font_size': 24,
        'xanchor': 'center',
        'x': 0.5

})

chart = fig.to_html()
chart2 = fig2.to_html()
context = {'chart': chart,'chart2':chart2,'result':a}
return render(request, 'chart.html', context)
```

URLS.py

```
from django.contrib import admin

from django.urls import path

from theapp import views



urlpatterns = [

    path('admin/', admin.site.urls),

    path('home',views.home),

    path('add',views.add),

    path('about',views.about),

    path('instructions',views.inst),

    path('developers',views.dev)




]
```

Settings.py

```
"""
Django settings for oner project.

Generated by 'django-admin startproject' using Django 4.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY    =    'django-insecure-lk(2tsr-&r@!esttexb&)z2q7dvdq-@8)0835bib-
j+4v+91+^'



# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True



ALLOWED_HOSTS = []




# Application definition



INSTALLED_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'theapp'

]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'oner.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
```

```
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'oner.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
```

```
#         https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-
validators



AUTH_PASSWORD_VALIDATORS = [

    {

        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',

    },

    {

        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',

    },

]




# Internationalization
```

```
# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [
    'static/'
]

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Index.html

```
<html>

<head>

    {% load static %}



    <link                                                    rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">

    <script
src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></scrip
t>

    <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <link rel="stylesheet" href="{% static 'style.css'%}">

</head>




<nav    class="navbar    navbar-icon-top    navbar-expand-lg    navbar-dark"
style="background-color: #122952">

    <a class="navbar-brand" href="home">Job Market Analysis Tool</a>

    <button   class="navbar-toggler"   type="button"   data-toggle="collapse"
data-target="#navbarSupportedContent"                                    aria-
controls="navbarSupportedContent"  aria-expanded="false"  aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

    </button>
```

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
      <a class="nav-link" href="home">
        <i class="fa fa-home"></i>
        Home
        <span class="sr-only">(current)</span>
        </a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="about">
        <i class="fa fa-info-circle">
          <span class="badge badge-danger"></span>
        </i>
        About
      </a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="instructions">
        <i class="fa fa-envelope-o">
          <span class="badge badge-danger"></span>
```

```
            </i>

            Instructions

          </a>

        </li>

        <!-- <li class="nav-item dropdown">

          <a        class="nav-link        dropdown-toggle"        href="#"
id="navbarDarkDropdownMenuLink" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">

            <i class="fa fa-envelope-o">

              <span class="badge badge-primary"></span>

            </i>

            Frequent Searches

          </a>

          <div class="dropdown-menu" aria-labelledby="navbarDropdown">

            <a class="dropdown-item" href="#">Action</a>

            <a class="dropdown-item" href="#">Another action</a>

            <div class="dropdown-divider"></div>

            <a class="dropdown-item" href="#">Something else here</a>

          </div>

        </li> -->

      </ul>

      <ul class="navbar-nav ">

        <li class="nav-item">
```

```
            <a class="nav-link" href="https://www.hirist.com/">
              <i class="fa fa-globe">
                <span class="badge badge-success"></span>
              </i>
              hirist.com
            </a>
          </li>
        <li class="nav-item">
          <a class="nav-link" href="developers">
            <i class="fa fa-desktop">
              <span class="badge badge-info"></span>
            </i>
            Developer
          </a>
        </li>


      </ul>
      <!-- <form class="form-inline my-2 my-lg-0"  action="add">

        <input type="text" class="search-bar" placeholder="Enter your search
term">
        <button class="search-button">Search</button>
```

```
      </form> -->

    </div>

  </nav>

<body>

    {% block content %}

    {% endblock %}

</body>

</html>
```

Chart.html

```
<head>

    {% load static %}



    <link                                                    rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">

    <script
src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></scrip
t>

    <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

    <link rel="stylesheet" href="{% static 'style.css'%}">

  </head>



  <style>

    .search-bar {

      width: 300px;

      padding: 10px;

      border: 2px solid #122952;

      border-radius: 5px;

      outline: none;

  }
```

```
.search-button {
    background-color: #ff2175;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}
  body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f4f4f4;
  }

  header {
      background-color: #e8ecf4;
      color: #636efa;
      padding: 10px;
      text-align: center;
  }
```

```
  </style>

  <nav class="navbar navbar-icon-top navbar-expand-lg navbar-dark" style="background-color: #122952">

    <a class="navbar-brand" href="home">Job Market Analysis Tool</a>

    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

      <span class="navbar-toggler-icon"></span>

    </button>



    <div class="collapse navbar-collapse" id="navbarSupportedContent">

      <ul class="navbar-nav mr-auto">

        <li class="nav-item">

          <a class="nav-link" href="home">

            <i class="fa fa-home"></i>

            Home

            <span class="sr-only">(current)</span>

            </a>

        </li>

        <li class="nav-item">

          <a class="nav-link" href="about">

            <i class="fa fa-info-circle">

              <span class="badge badge-danger"></span>
```

```
    </i>
    About
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="instructions">
    <i class="fa fa-envelope-o">
      <span class="badge badge-danger"></span>
    </i>
    Instructions
  </a>
</li>
<li class="nav-item active">
    <a class="nav-link" href="instructions">
      <i class="fa fa-line-chart">
        <span class="badge badge-danger"></span>
      </i>
      Insights
    </a>
  </li>
<!-- <li class="nav-item dropdown">
```

```
          <a        class="nav-link        dropdown-toggle"        href="#"
id="navbarDarkDropdownMenuLink" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">

            <i class="fa fa-envelope-o">

              <span class="badge badge-primary"></span>

            </i>

            Frequent Searches

          </a>

          <div class="dropdown-menu" aria-labelledby="navbarDropdown">

            <a class="dropdown-item" href="#">Action</a>

            <a class="dropdown-item" href="#">Another action</a>

            <div class="dropdown-divider"></div>

            <a class="dropdown-item" href="#">Something else here</a>

          </div>

        </li> -->

      </ul>

      <ul class="navbar-nav ">

        <li class="nav-item">

          <a class="nav-link" href="https://www.hirist.com/">

            <i class="fa fa-globe">

              <span class="badge badge-success"></span>

            </i>

            hirist.com
```

```
            </a>
          </li>
        <li class="nav-item">
          <a class="nav-link" href="developers">
            <i class="fa fa-desktop">
              <span class="badge badge-info"></span>
            </i>
            Developer
          </a>
        </li>

      </ul>
      <!-- <form class="form-inline my-2 my-lg-0"  action="add">

        <input type="text" class="search-bar" placeholder="Enter your search
term">
        <button class="search-button">Search</button>
      </form> -->
    </div>
  </nav>

<header>
```

```
    <h1> Results for {{ result }}</h1>
</header>

{% comment %} <span>
    {{ chart|safe }}
</span>

</span>
    {{ chart2|safe }}
</span> {% endcomment %}

<div style="width: 100%; height: 650px;">

  <div style="width: 50%; height: 550px; float:left;">
  {{ chart|safe }}
  </div>

  <div style="width:50%; height: 550px; float:left;">

  {{ chart2|safe }}
  </div>
```

```
</div>
```

About.html

```
<head>

  {% load static %}



  <link                                                    rel="stylesheet"
href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
rel="stylesheet" id="bootstrap-css">

  <script
src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></scrip
t>

  <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

  <link rel="stylesheet" href="{% static 'style.css'%}">

</head>



<style>

  .search-bar {

    width: 300px;

    padding: 10px;

    border: 2px solid #122952;

    border-radius: 5px;

    outline: none;

}
```

```
.search-button {

    background-color: #ff2175;

    color: #fff;

    border: none;

    padding: 10px 20px;

    border-radius: 5px;

    cursor: pointer;

}



</style>

<nav class="navbar navbar-icon-top navbar-expand-lg navbar-dark" style="background-color: #122952">

  <a class="navbar-brand" href="home">Job Market Analysis Tool</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

    <span class="navbar-toggler-icon"></span>

  </button>



  <div class="collapse navbar-collapse" id="navbarSupportedContent">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item">

        <a class="nav-link" href="home">
```

```
    <i class="fa fa-home"></i>
    Home
    <span class="sr-only">(current)</span>
    </a>
</li>
<li class="nav-item active">
  <a class="nav-link" href="about">
    <i class="fa fa-info-circle">
      <span class="badge badge-danger"></span>
    </i>
    About
  </a>
</li>
<li class="nav-item">
  <a class="nav-link" href="instructions">
    <i class="fa fa-envelope-o">
      <span class="badge badge-danger"></span>
    </i>
    Instructions
  </a>
</li>
<!-- <li class="nav-item dropdown">
```

```
        <a         class="nav-link         dropdown-toggle"         href="#"
id="navbarDarkDropdownMenuLink" role="button" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
          <i class="fa fa-envelope-o">
            <span class="badge badge-primary"></span>
          </i>
          Frequent Searches
        </a>
        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
          <a class="dropdown-item" href="#">Action</a>
          <a class="dropdown-item" href="#">Another action</a>
          <div class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Something else here</a>
        </div>
      </li> -->
    </ul>
    <ul class="navbar-nav ">
      <li class="nav-item">
        <a class="nav-link" href="https://www.hirist.com/">
          <i class="fa fa-globe">
            <span class="badge badge-success"></span>
          </i>
          hirist.com
```

```
          </a>
        </li>
      <li class="nav-item">
        <a class="nav-link" href="developers">
          <i class="fa fa-desktop">
            <span class="badge badge-info"></span>
          </i>
          Developer
        </a>
      </li>

    </ul>
    <!-- <form class="form-inline my-2 my-lg-0"  action="add">

      <input type="text" class="search-bar" placeholder="Enter your search term">
      <button class="search-button">Search</button>
    </form> -->
  </div>
</nav>

<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About the Project</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
        }

        header {
            background-color: #333;
            color: white;
            padding: 20px;
            text-align: center;
        }

        section {
            margin: 20px;
```

```
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h2 {
    color: #333;
}

p {
    line-height: 1.6;
}

ul {
    list-style-type: disc;
    margin-left: 20px;
}

footer {
    background-color: #333;
    color: white;
```

```
            text-align: center;
            padding: 10px;
            position: fixed;
            bottom: 0;
            width: 100%;
        }
    </style>
</head>
<body>

    <header>
        <h1>About the Project</h1>
    </header>

    <section>
        <h2>Overview</h2>
        <p>
            The "Job Market Analysis Tool" is an innovative web application designed to empower individuals and organizations
            with valuable insights into the dynamic job market. Whether you are a job seeker exploring career opportunities
            or an employer seeking to understand market trends, this tool provides a comprehensive solution.
```

```
        </p>
    </section>

    <section>
        <h2>Key Features</h2>
        <ul>
            <li><strong>Data Accuracy and Relevance:</strong> The tool ensures the accuracy and relevance of information by scraping
                data from various job platforms and processing it to provide up-to-date insights.</li>
            <li><strong>Interactive Visualizations:</strong> Dynamic charts and graphs allow users to explore and analyze job market trends
                in an engaging and interactive manner.</li>
            <li><strong>User-Friendly Interface:</strong> The intuitive interface is designed for easy navigation, ensuring that users
                can access valuable information effortlessly.</li>
            <li><strong>Agile Development Practices:</strong> The project embraced agile methodologies, fostering adaptability, responsiveness,
                and continuous improvement throughout the development lifecycle.</li>
        </ul>
    </section>

    <section>
        <h2>Future Scope</h2>
```

While the "Job Market Analysis Tool" has achieved its primary objectives, the journey doesn't end here. There are exciting

possibilities for future enhancements and expansions to ensure the tool remains a valuable resource:

Enhanced Data Sources: Expand the tool's capability to extract data from a wider range of job platforms,

providing users with an even more comprehensive view of the job market.

Advanced Analytics Features: Integrate advanced analytics features, such as predictive modeling, sentiment

analysis, and skill forecasting, offering deeper insights for users.

Customization and Personalization: Implement features that allow users to customize their dashboard, set preferences,

and receive personalized recommendations based on their career goals.

Integration with Learning Platforms: Collaborate with online learning platforms to integrate educational

recommendations based on emerging job trends and required skill sets.

Mobile Application Development: Develop a mobile application version of the tool to cater to users who prefer

accessing job market insights on their mobile devices.

```
            <li><strong>Collaboration with Industry Partners:</strong> Establish collaborations with industry partners, recruitment agencies, and
                educational institutions to enrich the tool's dataset and provide additional value to users.</li>
        </ul>
    </section>

    <footer>
        &copy; 2023 Job Market Analysis Tool
    </footer>

</body>
</html>
```

Instructions.html

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Instructions</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f4f4f4;
            color: #333;
        }

        header {
            background-color: #333;
            color: white;
```

```
    padding: 20px;
    text-align: center;
}

section {
    margin: 20px;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h2 {
    color: #333;
}

h3 {
    color: #333;
    margin-top: 15px;
}

p {
```

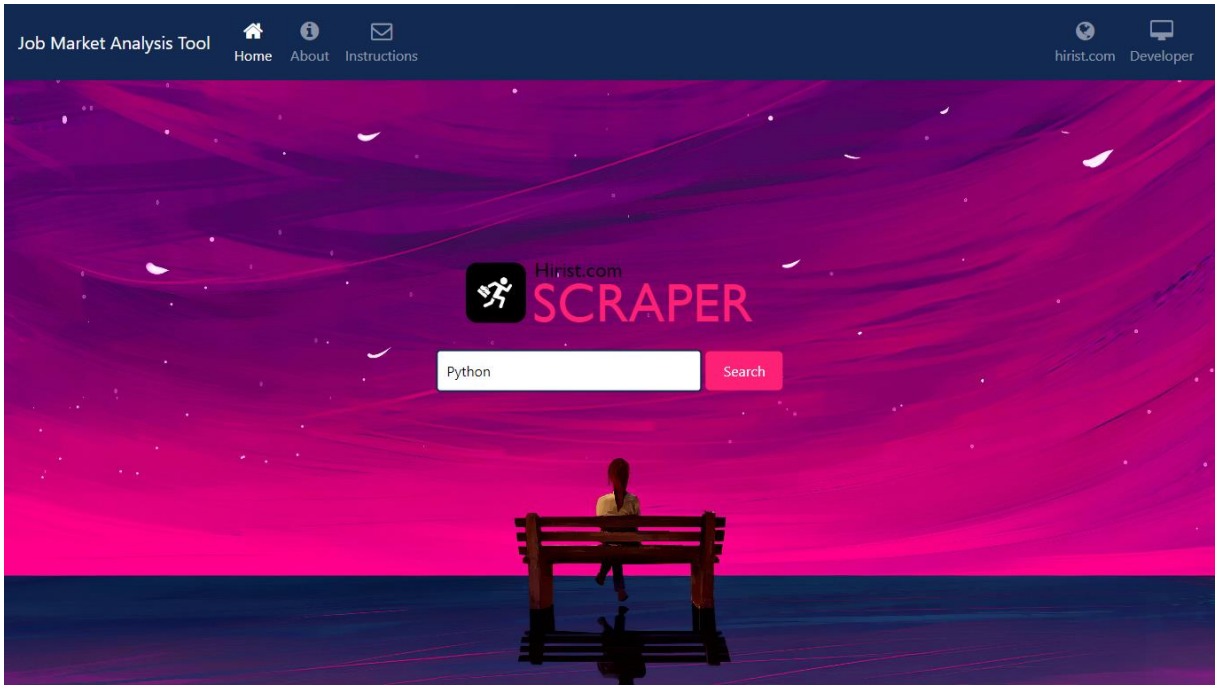
```
    line-height: 1.6;
}

ol {
    list-style-type: decimal;
    margin-left: 20px;
}

ul {
    list-style-type: disc;
    margin-left: 20px;
}

footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

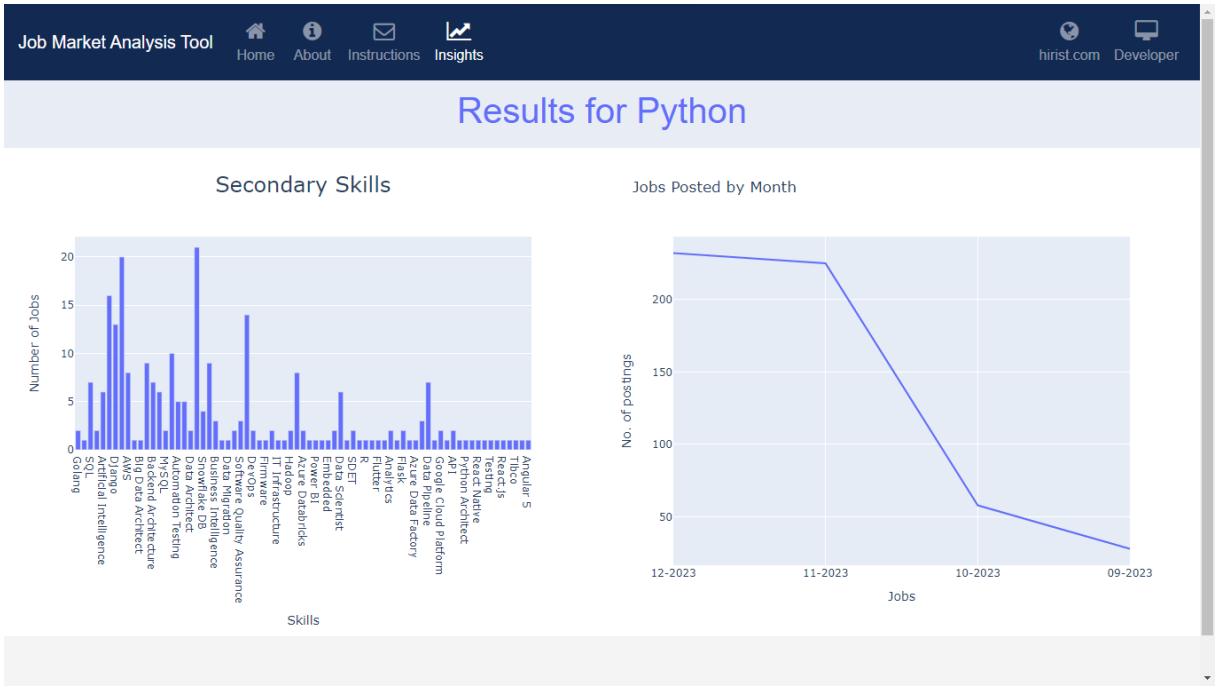
```
    </style>
</head>

<body>

    <header>
        <h1>Instructions</h1>
    </header>

    <section>
        <h2>Welcome to the Job Market Analysis Tool!</h2>
        <p>
            This tool is designed to provide you with valuable insights into the job market. Here's a guide on how to use its features and what to expect:
        </p>

        <h3>1. Search for Job Insights</h3>
        <p>
            Start by entering a job title or keyword in the search bar and click the "Search" button. The tool will scrape relevant data from job platforms to provide insights.
        </p>
```

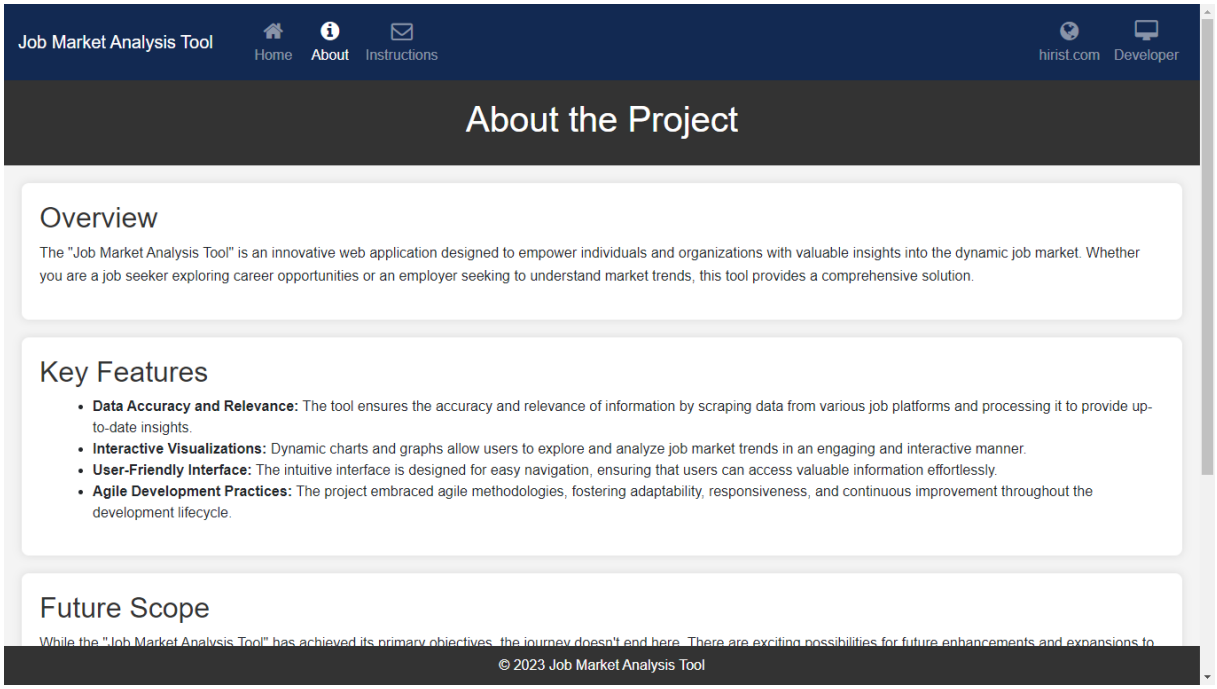
```
        <h3>2. Explore Visualizations</h3>

        <p>

            Once the data is retrieved, explore the interactive visualizations on the "Secondary Skills" and "Jobs Posted by Month" graphs. Click on data points for more details.

        </p>



        <h3>3. Key Features</h3>

        <ul>

            <li><strong>Data Accuracy:</strong> The tool ensures accurate and up-to-date information by scraping multiple job platforms.</li>

            <li><strong>User-Friendly Interface:</strong> Navigate through the tool effortlessly with its intuitive design.</li>

            <li><strong>Insightful Visualizations:</strong> Gain insights into secondary skills and job posting trends through dynamic graphs.</li>

        </ul>



        <h3>4. Future Scope</h3>

        <p>

            The tool is continuously evolving. Expect future updates with enhanced features, additional data sources, and advanced analytics capabilities.

        </p>

    </section>
```

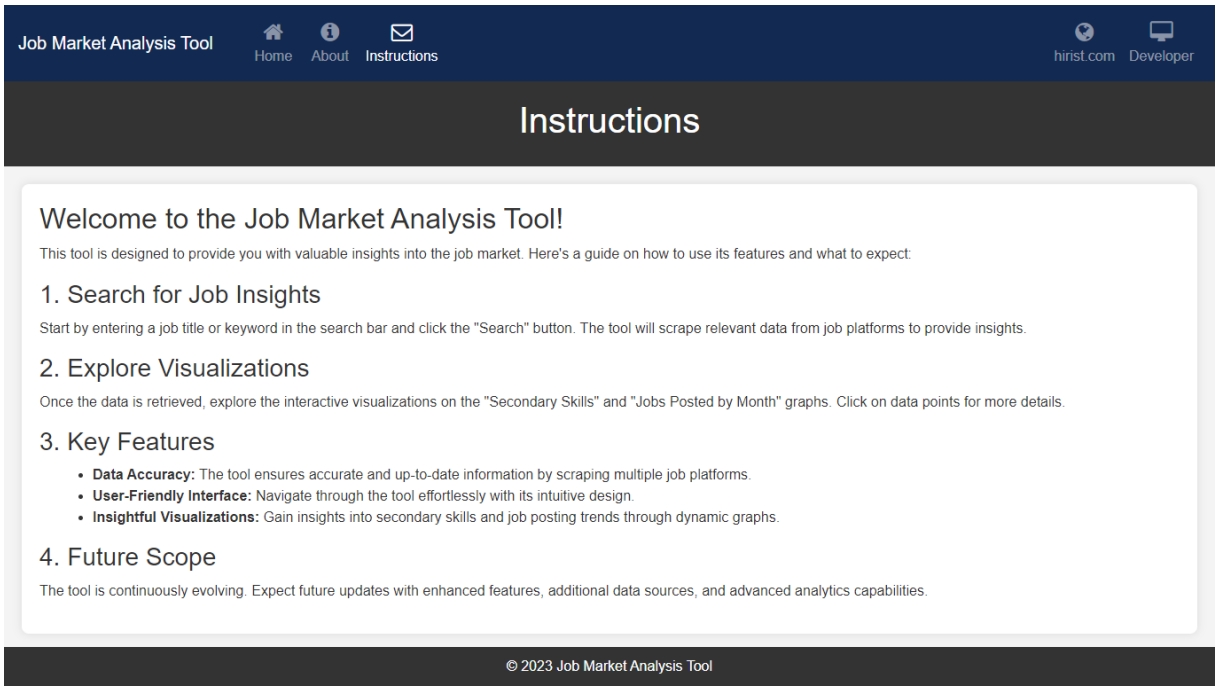
```
    <footer>
        &copy; 2023 Job Market Analysis Tool
    </footer>

</body>
</html>
```

SCREENSHOTS






Job Market Analysis Tool
Home
About
Instructions
hirist.com
Developer
About the Project
Overview
The "Job Market Analysis Tool" is an innovative web application designed to empower individuals and organizations with valuable insights into the dynamic job market. Whether you are a job seeker exploring career opportunities or an employer seeking to understand market trends, this tool provides a comprehensive solution.
Key Features
Data Accuracy and Relevance: The tool ensures the accuracy and relevance of information by scraping data from various job platforms and processing it to provide up-to-date insights.
Interactive Visualizations: Dynamic charts and graphs allow users to explore and analyze job market trends in an engaging and interactive manner.
User-Friendly Interface: The intuitive interface is designed for easy navigation, ensuring that users can access valuable information effortlessly.
Agile Development Practices: The project embraced agile methodologies, fostering adaptability, responsiveness, and continuous improvement throughout the development lifecycle.
Future Scope
© 2023 Job Market Analysis Tool


Job Market Analysis Tool
Home
About
Instructions
hirist.com
Developer
Instructions
Welcome to the Job Market Analysis Tool!
This tool is designed to provide you with valuable insights into the job market. Here's a guide on how to use its features and what to expect:
1. Search for Job Insights
Start by entering a job title or keyword in the search bar and click the "Search" button. The tool will scrape relevant data from job platforms to provide insights.
2. Explore Visualizations
Once the data is retrieved, explore the interactive visualizations on the "Secondary Skills" and "Jobs Posted by Month" graphs. Click on data points for more details.
3. Key Features
Data Accuracy: The tool ensures accurate and up-to-date information by scraping multiple job platforms.
User-Friendly Interface: Navigate through the tool effortlessly with its intuitive design.
Insightful Visualizations: Gain insights into secondary skills and job posting trends through dynamic graphs.
4. Future Scope
The tool is continuously evolving. Expect future updates with enhanced features, additional data sources, and advanced analytics capabilities.
© 2023 Job Market Analysis Tool

CHAPTER 8 - CONCLUSION

AND FUTURE SCOPE

8.1 Conclusion

The development and deployment of the "Job Market Analysis Tool" mark a significant milestone in addressing the dynamic challenges faced by job seekers and employers in the contemporary employment landscape. This project aimed to provide users with valuable insights into job market trends, skill requirements, and temporal patterns through an intuitive and interactive platform.

Throughout the development lifecycle, an agile methodology facilitated iterative improvements, collaboration, and the incorporation of user feedback. The tool underwent extensive testing, including user testing and manual testing by developers, ensuring its robustness and reliability. The seamless integration of web scraping, data processing, and visualization techniques empowered users to make informed decisions regarding their career paths and hiring strategies.

8.2 Key Achievements

1. **Data Accuracy and Relevance:**
    - The tool successfully scraped and processed data from various job platforms, ensuring the accuracy and relevance of information presented to users.
2. **Interactive Visualizations:**
    - Implemented interactive charts and graphs, enabling users to explore job market insights in a dynamic and engaging manner.

3. **User-Friendly Interface:**
    - The user interface was designed with a focus on simplicity and accessibility, ensuring that users could navigate the tool effortlessly.
4. **Agile Development Practices:**
    - The adoption of agile practices facilitated adaptability, responsiveness to user needs, and a continuous improvement mindset.

8.3 Future Scope

While the "Job Market Analysis Tool" has achieved its primary objectives, there are avenues for further enhancement and expansion to ensure its continued relevance and effectiveness. The following areas represent potential avenues for future development:

1. **Enhanced Data Sources:**
    - Expand the tool's capability to extract data from a wider range of job platforms and sources, providing users with a more comprehensive view of the job market.
2. **Advanced Analytics Features:**
    - Integrate advanced analytics features, such as predictive modeling, sentiment analysis, and skill forecasting, to offer deeper insights for both job seekers and employers.
3. **Customization and Personalization:**
    - Implement features that allow users to customize their dashboard, set preferences, and receive personalized recommendations based on their career goals and preferences.
4. **Integration with Learning Platforms:**

- Collaborate with online learning platforms to integrate educational recommendations based on emerging job trends and required skill sets.

5. **Mobile Application Development:**
    - Develop a mobile application version of the tool to cater to users who prefer accessing job market insights on their mobile devices.
6. **Collaboration with Industry Partners:**
    - Establish collaborations with industry partners, recruitment agencies, and educational institutions to enrich the tool's dataset and provide additional value to users.

The "Job Market Analysis Tool" stands as a foundation for ongoing innovation and evolution. By embracing emerging technologies, user feedback, and industry collaborations, the tool can continue to empower individuals and organizations in navigating the ever-evolving job market landscape. The commitment to continuous improvement ensures that the tool remains a valuable resource for years to come.

CHAPTER 8 – REFERENCES

1. Beautiful Soup Documentation

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

2. Plotly Express Documentation

URL: https://plotly.com/python/plotly-express/

3. Django Documentation

URL: https://docs.djangoproject.com/

4. Requests Library Documentation

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

5. HTML and CSS Tutorials - W3Schools

URL: https://www.w3schools.com/

6. Stack Overflow - Community Q&A

URL: https://stackoverflow.com/

7. Python Documentation

URL: https://docs.python.org/3/

8. Folium Documentation

URL: https://python-visualization.github.io/folium/