**The Superior University**

**Project Title:**  
Car Data Scraper from PakWheels

**Name:**  
Muhammad Ahmad

**Roll No:**  
SU92-BSDSM-F23-031

**Project Details**

**Course:** Data Structures and Algorithms  
**Instructor:** Rasikh Ali   
**Semester:** 3rd  
**Section:** BSDSM-3A  
**Submission Date:** 09/12/2024

**Abstract**

The "Car Data Scraper from PakWheels" is a Python-based project that automates data collection from the PakWheels website. The project leverages web scraping techniques using libraries such as BeautifulSoup and pandas to extract and structure car listings. The resulting dataset includes attributes such as car titles, locations, models, mileage, fuel types, engine capacity, specifications, and prices. This project showcases the practical application of automation and data handling in Python, making it a valuable tool for market analysis and consumer research.

**Table of Contents**

1. **Introduction**
2. **Objectives**
3. **System Requirements**
4. **Methodology**
5. **Implementation**
6. **Challenges and Solutions**
7. **Conclusion**

**1. Introduction**

The "Car Data Scraper from PakWheels" aims to simplify the process of gathering car listing data for analysis. By automating the extraction of information from the PakWheels website, this project demonstrates the power of web scraping in real-world applications, such as market analysis for the automotive industry.

**2. Objectives**

* To design and implement a web scraper for extracting car data.
* To create a structured dataset from unorganized web content.
* To understand the practical use of libraries like BeautifulSoup and pandas.
* To facilitate market analysis and trend prediction for automotive data.

**3. System Requirements**

**Hardware Requirements:**

* **Processor:** Intel Core i3 or higher
* **RAM:** 4 GB or more
* **Storage:** 1 GB free space

**Software Requirements:**

* **Programming Language:** Python 3.x
* **Libraries:** requests, BeautifulSoup, pandas
* **Operating System:** Windows/Linux/MacOS

**4. Methodology**

**Design Approach:**

* Utilize Python's requests library to fetch HTML content from the website.
* Parse the HTML using BeautifulSoup to extract relevant car data.
* Structure the data into a CSV file using pandas for further analysis.

**Workflow:**

1. Fetch HTML content for multiple pages of car listings.
2. Parse car details such as title, location, price, and specifications.
3. Save the structured data in a CSV format.

**Algorithm:**

1. Loop through the pages of the PakWheels website.
2. Use BeautifulSoup to extract specific details.
3. Append the extracted data to lists and store them in a pandas DataFrame.
4. Export the DataFrame to a CSV file.

**5. Implementation**

**Core Components:**

* **requests.get(url)**: Fetches the webpage HTML content.
* **BeautifulSoup**: Parses and extracts relevant data.
* **pandas DataFrame**: Structures the extracted data.
* **CSV Export**: Saves the data in a usable format.

**Sample Code Snippet:**

data = pd.DataFrame({

'Title': titlelist,

'Location': locationList,

'Model': modelList,

'KM': kmList,

'Fuel Type': fuelTypelist,

'CC': ccList,

'Specs': spexlist,

'Price': pricelist

})

data.to\_csv('CarData\_From\_PakWheels.csv', index=False)

print("Data saved successfully!")

**6. Challenges and Solutions**

**Challenge:** Parsing inconsistent or missing data on the website.  
**Solution:** Implemented conditional checks to handle null or incomplete values.

**Challenge:** Handling large volumes of data over multiple pages.  
**Solution:** Used a loop-based approach with pagination to ensure comprehensive scraping.

**7. Conclusion**

The "Car Data Scraper from PakWheels" project demonstrates the effective use of Python for automating web data extraction. It highlights the importance of data handling in building robust and scalable solutions for real-world problems. Future enhancements may include database integration, advanced data visualization, and support for scraping additional automotive websites.

Let me know if you want to make adjustments to this draft!