Documentation

for

Scrapper App

Version 1.0

Supervisor: Prof. Saravana Jaikumar, Mr. Bibek Guha Sarkar

Indian Institute of Management Calcutta

Team Members: Arshit, Harsh Agrawal, Naman K Jaiswal, Prem Kansagra, Raj Vinayak

Indian Institute of Technology Kanpur

CONTENTS II
REVISIONS II

1 INTRODUCTION 1

2 IMPLEMENTATION DETAILS 9

3 CODEBASE 12

4 USER MANUAL 13

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Arshit, Harsh Agrawal, Naman K Jaiswal, Prem Kansagra, Raj Vinayak	Initial Version	10/05/2024

1. Introduction

1.1 Purpose

The purpose of the website scraper focused on extracting data from TripAdvisor could serve various stakeholders and purposes:

1. **Travel Enthusiasts and Tourists**: This tool could provide valuable insights for travelers looking to plan their trips more efficiently. By scraping data such as the names, locations, number of reviews, ratings, and more from major attractions in any city, users can easily compare attractions, assess their popularity and quality, and make informed decisions about their travel itineraries.
2. **Travel Agencies and Tour Operators**: For businesses in the travel industry, having access to comprehensive data about attractions can be crucial for creating appealing travel packages and itineraries. This scraper could help travel agencies gather up-to-date information about popular attractions, enabling them to tailor their offerings to the preferences of their target audience.
3. **City Planners and Tourism Boards**: City planners and tourism boards could benefit from this tool by using the scraped data to analyze trends in tourist preferences and behavior. Understanding which attractions receive the most attention and positive reviews can inform decisions about infrastructure development, marketing strategies, and allocation of resources to enhance the overall tourism experience in a city.
4. **Market Researchers**: Researchers studying travel trends, consumer behavior, or the tourism industry as a whole could utilize the scraped data to conduct analyses and extract valuable insights. This information could be used to identify emerging tourism hotspots, predict future travel trends, or assess the impact of various factors on the popularity of attractions.
5. **Data Analysts and Developers**: Data analysts and developers could use the scraped data to create visualizations, develop predictive models, or build recommendation systems aimed at improving the travel planning experience for users. By extracting and analyzing data from TripAdvisor, they can uncover patterns and correlations that can drive innovation in the travel technology space.

Overall, the purpose of this website name scraper would be to provide valuable data and insights related to tourist attractions, catering to the needs of travelers, businesses, researchers, and other stakeholders involved in the tourism ecosystem.

1.2 Product Scope

The product scope for the website scraper would encompass several key features and functionalities to fulfill its purpose effectively:

1. **Features and Functionalities**:
    a. Web Scraping: Ability to extract data from TripAdvisor's website, including attraction names, locations, number of reviews, ratings, and other relevant information.
    b. Multiple Attractions Input: Support for inputting multiple attractions or a list of attractions from different cities simultaneously.
    c. Customizable Data Selection: Flexibility for users to choose specific data attributes to scrape for each attraction.
    d. Review Scraping: Capability to extract reviews for each attraction to provide deeper insights into visitor experiences.
    e. Data Export: Option to export scraped data in formats such as CSV, JSON, or Excel for further analysis and integration with other tools.
    f. Error Handling: Mechanisms to handle potential errors or issues during the scraping process to minimize data loss and ensure reliability.
2. **User Requirements**:
    a. Target Audience: Travel enthusiasts, tourists, travel agencies, city planners, tourism boards, market researchers, data analysts, and developers seeking comprehensive data about tourist attractions.
    b. User Experience: Intuitive interface with clear instructions and controls to facilitate efficient data extraction and analysis.
3. **Technical Requirements**:
    a. Web Scraping Technologies: Utilization of web scraping libraries/tools such as BeautifulSoup and Selenium for parsing HTML and extracting data.
    b. Programming Language: Development in Python for its versatility and extensive support for web scraping and data processing.
    c. Compatibility: Ensure compatibility with major web browsers and operating systems for broader accessibility.
4. **Boundaries and Limitations**:
    a. Scope Limitation: Focus primarily on extracting data from TripAdvisor's website and not from other sources or platforms.
    b. Legal and Ethical Considerations: Compliance with TripAdvisor's terms of service and legal regulations governing web scraping and data usage.
5. **Timeline and Milestones**:
    a. Development Timeline: 1-1.5 month timeframe for development, testing, and of the scraper.
    b. Able to Scrape Data Without IP Blocking

i. Description: Develop and implement techniques to scrape data from TripAdvisor's website without triggering IP blocking mechanisms due to frequent requests. This milestone involves configuring appropriate request headers and implementing rate-limiting strategies to ensure that scraping activities are conducted in a manner that mimics human behavior and avoids detection.
ii. Deliverables:
    1. Implementation of custom request headers to mimic legitimate browser requests.
    2. Integration of rate-limiting mechanisms to control the frequency of requests and prevent excessive traffic.
    3. Validation testing to ensure that the scraper can retrieve data consistently without triggering IP blocks.

6. **Resources and Budget**:
    a. Human Resources: Allocation of developers, testers, and other team members for development and testing.
    b. Infrastructure: Requirement of necessary hardware and software (laptop and proper internet connection) for development and deployment.

By encompassing these features within its product scope, the TripAdvisor website name scraper can offer a comprehensive solution for extracting valuable data about tourist attractions, catering to the needs of various users across different domains.

1.3 Definitions, Acronyms and Abbreviations

Definitions, acronyms, and abbreviations that might be relevant for a website name scraper focused on TripAdvisor:

Term	Definition
Web Scraping	The process of extracting data from websites. It involves parsing the HTML structure of web pages to retrieve specific information.
CSS	Cascading Style Sheets
TripAdvisor	An online platform that provides reviews and information on travel-related content, including hotels, restaurants, attractions, and more.
HTML	Hypertext Markup Language
CSV:	Comma-Separated Values:- A file format used for storing tabular data, where each line represents a row of data, and commas separate the values within each row.
HTTPS	Hypertext Transfer Protocol Secure
API	A set of rules and protocols that allows different software applications to communicate with each other. APIs are commonly used to access and retrieve data from web services.
URL	A web address that specifies the location of a resource on the internet. URLs consist of several components, including the protocol (e.g., "http://" or "https://"), domain name, path, and optional query parameters
IEEE	Institute of Electrical and Electronics Engineers
Scalability	The ability of a system, network, or process to handle a growing amount of work or data in a graceful manner, without sacrificing performance or reliability.
JS	JavaScript
XPath	XML Path Language. A query language is used for selecting nodes from an XML document. It is commonly used in web scraping to navigate and extract data from HTML documents based on their structure.
Error Handling	The process of anticipating, detecting, and resolving errors or exceptions that may occur during the execution of a program or operation.
JSON	JavaScript Object Notation. A lightweight data-interchange format commonly used for representing structured data. It is often used for transmitting data between a server and a web application.

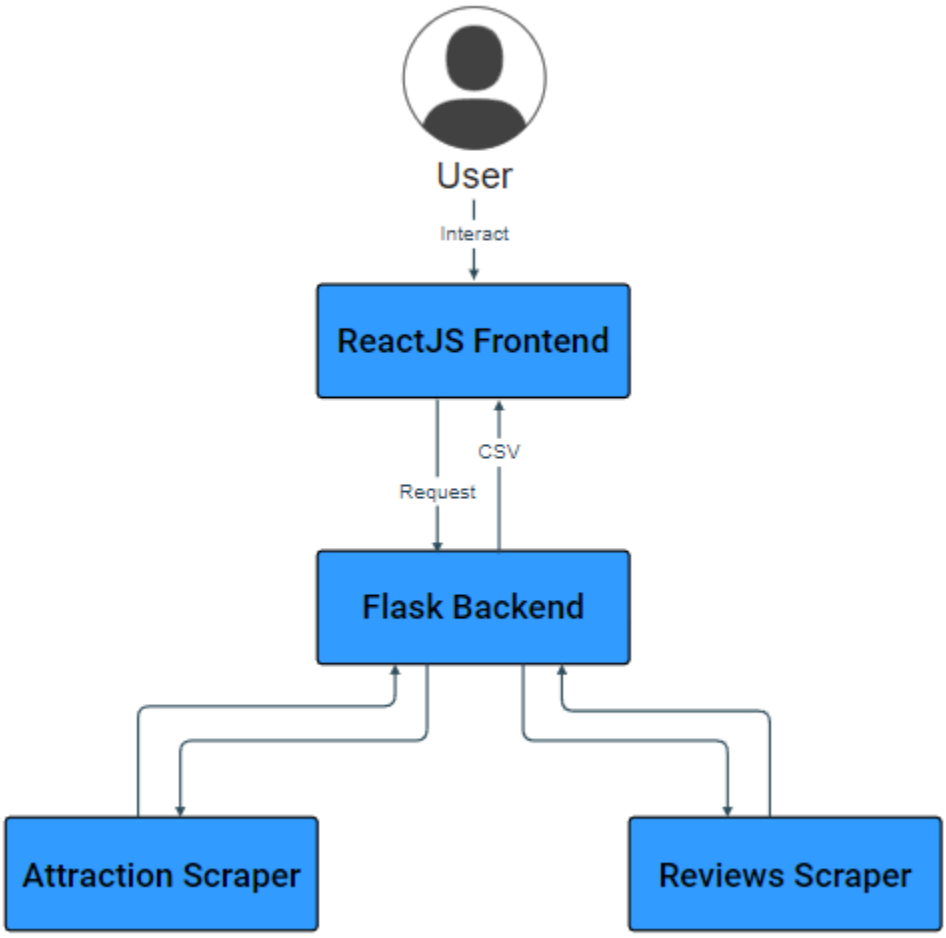
These definitions, acronyms, and abbreviations should provide a foundational understanding of some of the key concepts and terms relevant to website scraper.

1.4 References and Acknowledgments

1. **TripAdvisor**: We acknowledge TripAdvisor as the primary source of the data being scraped. For guidance on data usage and compliance, please refer to TripAdvisor's terms of service.
2. **Beautiful Soup and Selenium**: Acknowledgment of BeautifulSoup and Selenium as fundamental tools for parsing HTML, interacting with web elements, and extracting data from TripAdvisor's website. For more information, visit the official documentation for BeautifulSoup and Selenium.
3. **Python Requests Library**: Acknowledgment of the Python Requests library for its role in making HTTP requests to retrieve web pages. For detailed information and usage instructions, refer to the Python Requests documentation.
4. **Open Source Community**: Acknowledgment of the contributions of the open-source community. Many libraries and resources used in the development process are maintained by open-source contributors. Explore the Python Package Index (PyPI) for a wide range of open-source Python packages.

2. Implementation Details

Our Web Scraping Application can essentially be seen as three components - Frontend, Backend and Web Scraper. The Web Scraper has two scrapers in it, namely the attraction scraper and the reviews scraper. The application architecture can be seen as a small scale implementation of a Model View Controller Architecture where the Frontend is the View Component where the user is allowed to interact in accord to his need. The model is the backend which processes the user interaction to figure out the need of the user and then sends the Scrape Query to the Model where it goes to the right scraper and after the required Scraping time, the results are sent to the Model and then to the User in the form of a downloadable CSV.



1. **Frontend:**

   The frontend of the web scraping application is essentially a ReactJS App, implemented using ReactJS, a popular JavaScript library for building user interfaces. ReactJS allows building reusable UI components and managing the application's state efficiently, while React Router is used for declarative routing in the application, allowing navigation between different pages. We prefer ReactJS over its alternatives because of the following reasons:

   a. **Performance Optimization:** ReactJS features like memoization (optimize performance of functions by caching results of expensive function calls), PureComponent (base class that implements a shallow comparison of props and state to determine whether a component should re-render) and React.memo (higher-order component provided by React that memorizes the result of a functional component rendering) to optimize rendering performance and reduce unnecessary re-renders.
   b. **Virtual DOM:** In a state change of the application (State Diagram) only the necessary components of the Virtual DOM (in-memory representation of the actual DOM) get updated. This efficient reconciliation algorithm minimizes the performance overhead associated with updating the UI.
   c. **Component-Based Architecture:** React breaks down the UI elements into reusable modular components which enhances maintainability and scalability in large complex UIs.
   d. **Unidirectional Data Flow:** React promotes a unidirectional data flow from parent components to child components. This simplifies data management and makes it easier to reason about the state of the application while development phase of the software

2. **Backend**

   The backend of the web scraping application is implemented using Python and Flask, a micro web framework for building web applications. Flask is used for creating the RESTful API endpoints and handling HTTP requests from the frontend. Flask-CORS is used to enable Cross-Origin Resource Sharing (CORS), allowing the frontend to make requests to the backend from a different origin. It handles which origins have the permission to access the API endpoint. Flask is a much simpler and fast alternative due to the following reasons:

   a. **Simplicity and Minimalism:** Flask is designed to be lightweight and minimalistic, providing just the essentials for web development. Its simple and intuitive API makes it easy for developers to understand and use. This simplicity reduces the learning curve and allows for faster development of web applications.
   b. **Flexibility:** Flask follows a "micro-framework" approach, meaning it provides only the core components needed for web development, such as routing and request handling. This minimalistic design allows developers to choose and integrate additional libraries and extensions based on the specific requirements of their project. Flask's flexibility makes it suitable for a wide range of applications, from simple APIs to complex web applications.

c. **Scalability:** While Flask is known for its simplicity and lightweight design, it is also capable of handling scalable web applications. Flask applications can be deployed in various environments, including traditional web servers, containerized environments (e.g., Docker), and cloud platforms (e.g., AWS, Google Cloud). With proper architecture and design, Flask applications can scale to accommodate increasing traffic and workload demands.

d. **Extensibility:** While Flask itself is minimalistic, it offers a robust ecosystem of extensions that can be used to add functionality to the framework. These extensions cover various aspects of web development, including authentication, database integration, form validation, and more. Developers can choose from a wide range of extensions to enhance their Flask applications without reinventing the wheel.

e. **JSON Serialization:** Flask automatically serializes Python objects into JSON format, making it easy to send data between the frontend and backend.

We have also used Flask-SocketIO, a Flask extension that enables WebSocket communication in Flask applications. WebSocket is a protocol that allows for full-duplex communication channels over a single TCP connection. Flask-SocketIO facilitates real-time, bidirectional communication between the client and server. With WebSocket support, Flask-SocketIO enhances the interactivity and responsiveness of web applications compared to traditional HTTP-based communication.

3. **Web Scraper:** The web scraper component is implemented using Python and libraries such as BeautifulSoup and pandas for scraping data from web pages and generating CSV files. BeautifulSoup is used for parsing HTML and XML documents, extracting data from web pages, and navigating the HTML tree structure. We prefer BeautifulSoup due to the following reasons:

a. **Ease of Use:** Beautiful Soup provides a simple and intuitive interface for parsing HTML and XML documents. Its syntax is easy to understand and use, making it accessible to both beginners and experienced developers. With Beautiful Soup, developers can quickly write scraping scripts without extensive knowledge of HTML parsing techniques.

b. **Powerful Parsing:** Beautiful Soup excels at parsing poorly formatted or invalid HTML documents. It can handle complex HTML structures, nested tags, and malformed markup gracefully, allowing developers to extract data reliably from a wide range of web pages. This robust parsing capability is essential for web scraping tasks where data quality and consistency are paramount.

c. **Flexible Navigational API:** Beautiful Soup offers a flexible navigational API that enables developers to traverse the parse tree and locate specific elements efficiently. Developers can use various methods and selectors to find elements based on tag names, CSS classes, attributes, text content, and more. This versatility allows for precise and targeted scraping of desired information from web pages.

d. **Integration with Requests:** Beautiful Soup seamlessly integrates with the Requests library, a popular HTTP client for making web requests in Python. Together, Requests and BeautifulSoup provide a comprehensive solution for fetching web pages and extracting data from them. Developers can use Requests to fetch HTML content and then pass it to Beautiful Soup for parsing and extraction.

3 Codebase

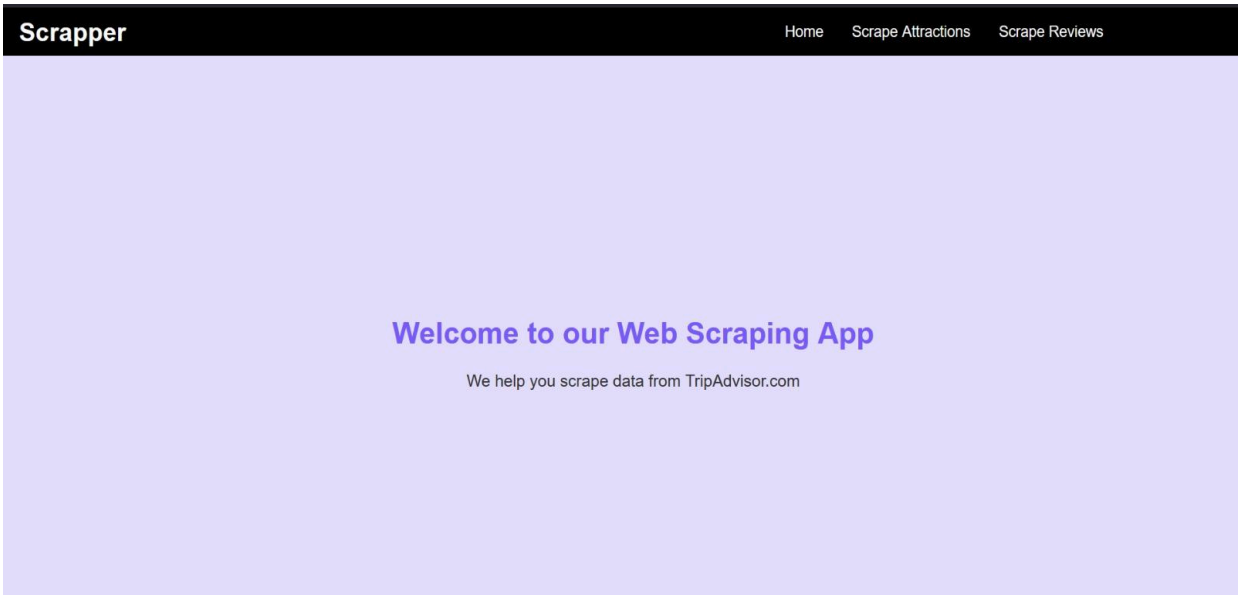
Our project codebase is present on GitHub for version control and collaboration purposes. You can access the repository through the following link:

https://github.com/Harsh-Agrawal-425/Scraper_App.git

Feel free to explore the codebase to gain a deeper understanding of our implementation and structure.

User Manual

Upon entering our website, you'll be greeted by our homepage

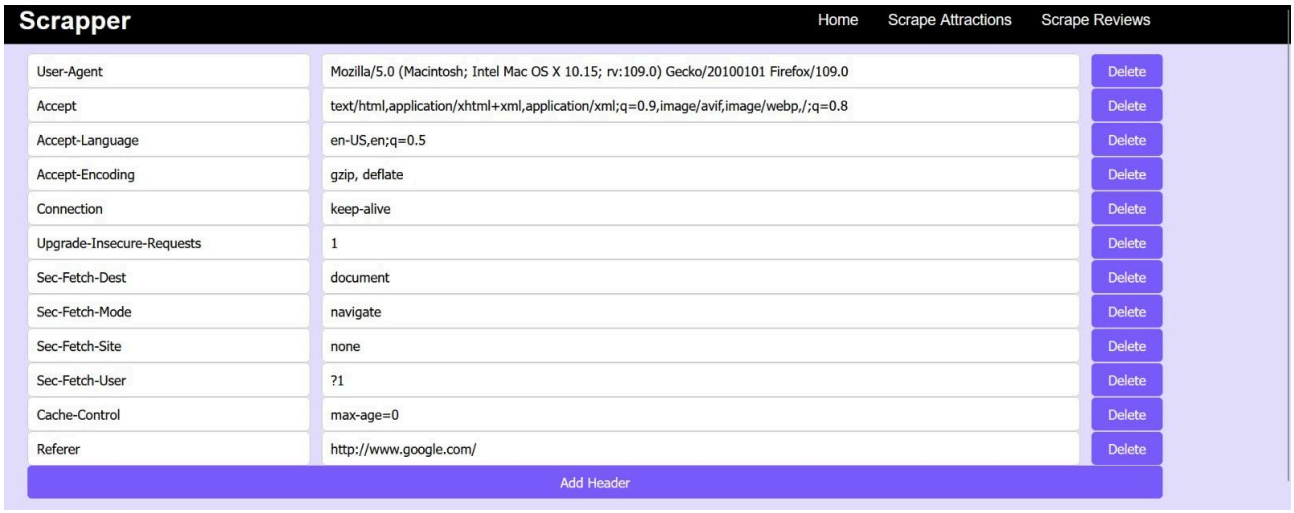


This is the home page of our application.The scrape attractions and scrape reviews are the two key aspects of our scraping program, which are available on this main page.

Homepage Overview:

- **Greetings:** At the center of the homepage, you'll find a friendly greeting.
- **Navigation Tabs:** Located on the top right corner of the page, you'll see three tabs that serve as gateways to different sections of our website. These tabs are designed for intuitive navigation, allowing you to seamlessly explore and access the pages you're interested in.

SCRAPE ATTRACTIONS PAGE



1. **Headers:**

- A header in the context of a website refers to the information sent between a client (such as a web browser) and a server during an HTTP request. It contains metadata about the request or the response, including details like the type of content being sent, the encoding method used, and authentication credentials. By displaying the headers used in the website, users can access this information without the need to alter their IP address.
- Headers are mostly used to keep the server from banning our program for scraping data. Changing the header details manually is supported by our application.Using several template headers that are readily available online, the user can change the header details in case of blocking by the server.

2. **Input Fields:**

This section contains two main parts:

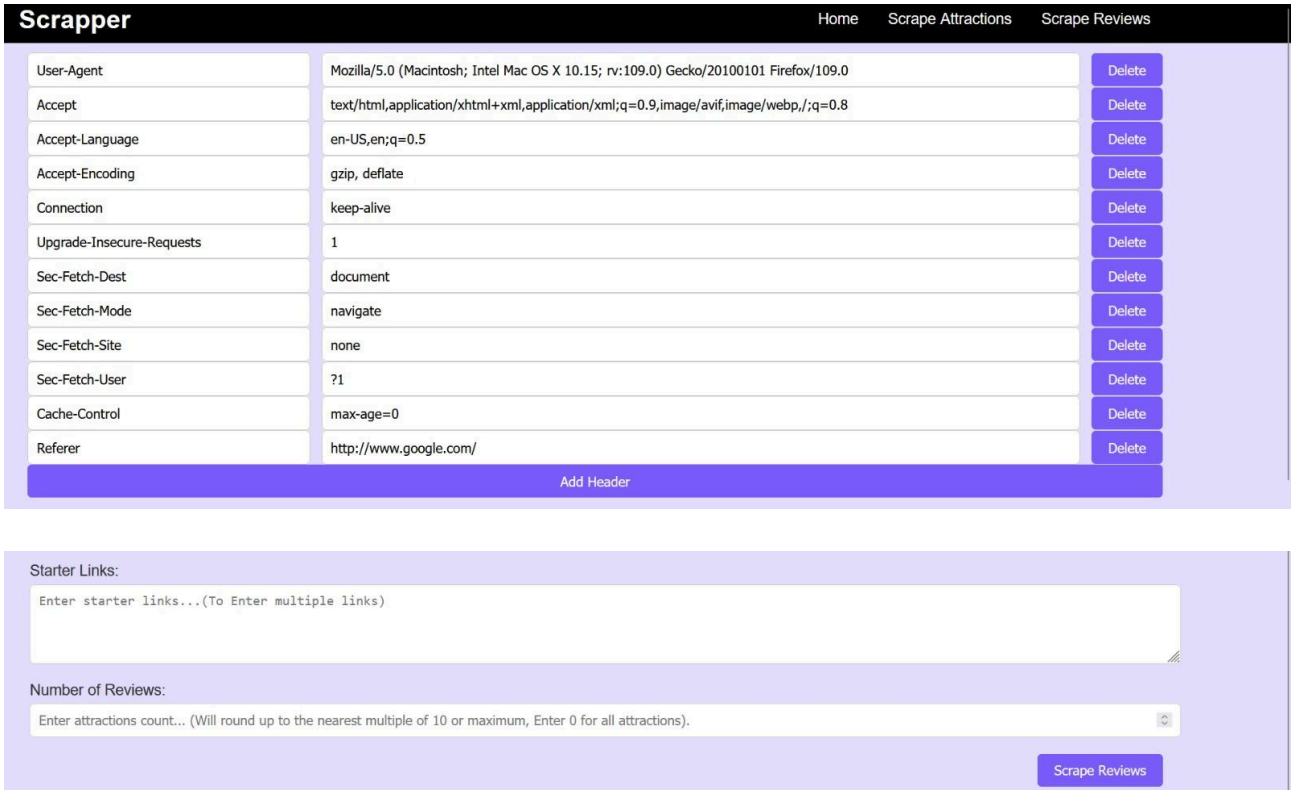
- **Starter Links:** This is the text field where the user enters the URLs of the websites from which they want to scrap their data from.The user can also scrape data from multiple URLs in a single process by specifying different URLs separated by commas.
- **Number of Attractions:** Using the initial links as a starting point, users can enter the number of attractions they wish the Scraper tool to scrape from the given link in this

text field. In order to scrape every page of the specified link, the user must enter 0. If not, the scraper will automatically scrape the next multiple of 30 attractions i.e. let's say a user enters 31 in no. of attractions, then scraper will automatically scrap 60 attractions. In case of multiple URLs, the same no. of attractions will be scrapped for each URL.

3. **Button**:

- This is a single button labeled "Scrape Attractions" which on clicking initiates the data scraping process.

SCRAPE REVIEWS PAGE



The 'scrape reviews' page is quite similar to the 'scrape attractions' page with only differences lying in the input fields.

Input Fields:

This section contains two main parts:

- **<u>Starter Links:</u>** This is the text field where the user enters the URL links of the attraction whose review data they want to scrape.The user can also scrape data from multiple URLs in a single process by specifying different URLs in a different line.

- **Number of Attractions:** Using the initial links as a starting point, users can enter the number of attractions they wish the Scraper tool to scrape from the given link in this text field. In order to scrape all the reviews of the specified link, the user must enter 0. If not, the scraper will automatically scrape the next multiple of 10 attractions i.e. let's say a user enters 31 in no. of attractions, then scraper will automatically scrap 40 attractions. In case of multiple URLs, the same no. of attractions will be scrapped for each URL.

Button:

- This is a single button labeled "Scrape Attractions" which on clicking initiates the data scraping process.