

# **Web Scraping of Zauba Corp**

- Proof of Concept (POC)

July 2024

December 2023

**Contents**

[**Web Scraping of Zauba Corp** 1](#_Toc171948735)

[**1** **Purpose** 3](#_Toc171948736)

[**2** **Objective** 3](#_Toc171948737)

[**3** **Introduction:** 3](#_Toc171948738)

[**4** **Methodology** 4](#_Toc171948739)

[**4.1** **Initial Setup Creating CSV File**: 4](#_Toc171948740)

[**4.2** **Distributed Web Scraping:** 5](#_Toc171948741)

[**4.3** **Batch processing:** 6](#_Toc171948742)

[**4.4** **Efficient MySQL Data Storage**: 6](#_Toc171948743)

[**5** **Limitations:** 7](#_Toc171948744)

[**6** **Tools and Techniques** 7](#_Toc171948745)

[**7** **Output:** 8](#_Toc171948746)

# **Purpose**

In today's digital era, access to reliable and comprehensive corporate information is crucial for various governmental and regulatory activities. The Zauba Corp website lists over 27 lakh companies, providing a wealth of data that is currently underutilized. Given the vast number of companies and the critical nature of the information, this project aims to bridge the gap between raw data and actionable insights. This effort is essential for empowering government agencies with the information needed for thorough due diligence, market analysis, compliance checks, and fostering transparency in corporate governance.

Ultimately, this project seeks to demonstrate the strategic value of leveraging technology for large-scale data collection and management, thereby supporting informed decision-making and promoting a more transparent business environment. Therefore, it was decided to undertake this project to systematically scrape and organize the data available on the Zauba Corp website. By doing so, the project aims to transform the way corporate information is accessed and utilized, ensuring it becomes a valuable resource for various governmental stakeholders.

# **Objective**

Following are the key objectives of this report:

* To scrape all the freely available data of the companies, including their director names, for all companies (over 2.7 million) listed on the Zauba Corp website.
* To design and implement a MySQL database that can efficiently store the scraped data with ensuring data integrity and accuracy.

# **Introduction:**

Zauba Corp is a comprehensive online platform that provides detailed information about companies and their directors. Despite its extensive database, the site employs certain restrictions to limit the ease of data extraction. Zauba Corp does not readily display all 27 lakh companies for general browsing.

The website is designed to present its data in a paginated format, allowing users to view only 13,333 pages, with each page listing 30 companies. This pagination restricts direct access to the entire database in a single view, making it difficult to scrape the data in bulk.

However, Zauba Corp offers various sorting and filtering options, such as Status, Age Category, Paid Up Capital, and ROC. These filters, rather than adding complexity, facilitate comprehensive data collection. Users can try different permutations and combinations to get the most unique list of companies possible.

# **Methodology**

Following is the process flow of the methodology

## **Initial Setup Creating CSV File:**

To effectively scrape data from the Zauba Corp website, leveraging various sorting and filtering options provided by the site is essential. These filters allow for categorizing companies into different financial and operational brackets.

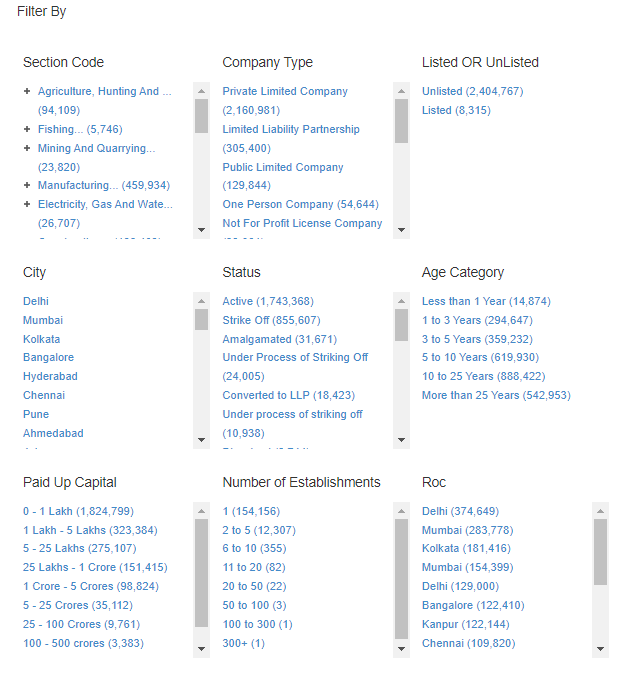


Figure 1: Filtering options available on the Zauba Corp website

After examining the available sorting options and compiling the company counts for each category, Paid Up Capital, Status, and Age Category provide the highest number of results, respectively. For instance, the Paid-Up Capital category (0 – 1 Lakh) includes 1,824,799 companies (see Figure 1), which exceeds the maximum allowable count of 399,990 (13,333 pages \* 30 companies per page) per filter. Therefore, additional filters should be applied until the count is under 399,990, as each filter can retrieve a maximum of 399,990 companies.

Once all the sufficient filters are applied, the page displays the total number of pages which is less than or equal to 13333, and the URL pattern for the filtered results (see Figure 2). By iterating through the page numbers and replacing them in the URL, it is possible to scrape and store the details of all listed companies in the database. This approach was used to create a CSV file containing the total page count and URL pattern for all permutations and combinations of the filtering criteria.

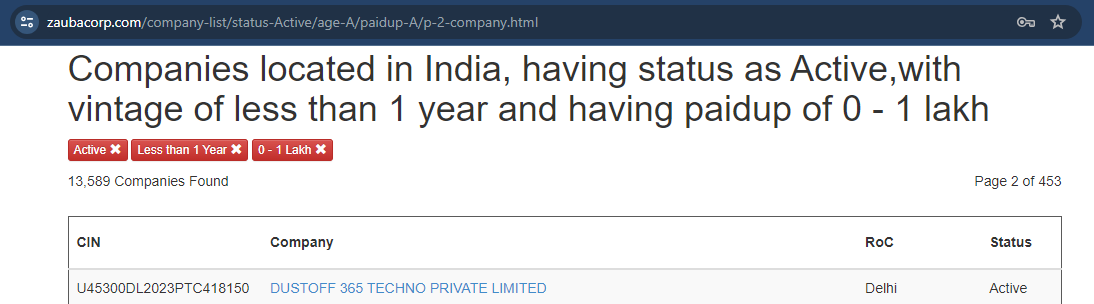


Figure 2: URL pattern and the total number of pages for the filtered results after applying filters.

## **Distributed Web Scraping:**

* 1. Distributed Scraping Architecture

To efficiently gather data from 27 lakh URLs on Zauba without overloading a local PC or risking IP blocks, a distributed scraping approach was devised. By utilizing 100 Render accounts, each hosting a server with a function to handle URL requests and return scraped data, the process was optimized. Decentralizing the scraping tasks avoided the risk of the local PC getting blocked and significantly reduced processing time. This approach ensured continuous operation and resource optimization, enabling the systematic population of the database with a comprehensive dataset.

Leveraging this distributed setup mitigated the challenges associated with scraping a large volume of web pages. Each server's capability to handle requests independently and concurrently allowed for effective parallelization of the scraping process. This distributed architecture enhanced reliability and expedited data collection, reducing what could have been a daunting month-long task into a manageable timeframe. Now using this architecture first 4 lakh data using 25 servers took 6 days from 7 June to 12 June and the other 23 lakh data using 100 servers took 5 days from 22 June to 26 June.

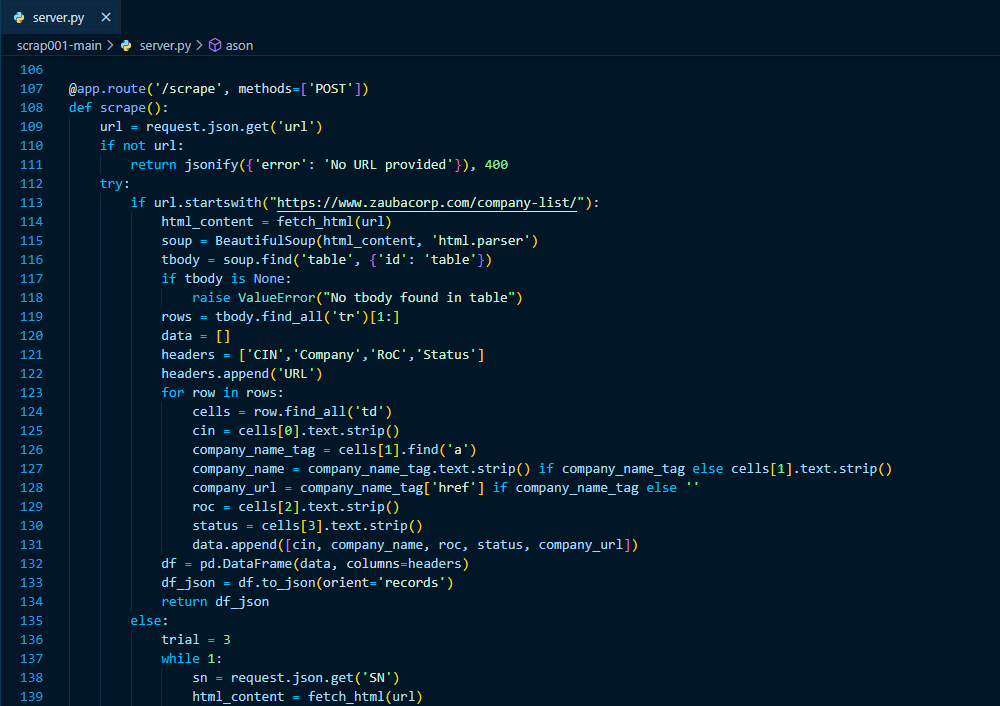


Figure 3: server.py shows how it handles different URLs for company lists and direct company pages.

This server.py includes a scrape function capable of distinguishing between a list of companies' URLs and a direct company URL, performing the appropriate scraping for each. It returns the scraped data accordingly (see Figure 3). To reliably fetch HTML pages, the script utilizes an external JavaScript file, as direct Python methods often fail.

* 1. Deployment and Maintenance

A public GitHub repository was created, containing the necessary files such as server.py, fetchAndExtract.js, and a templates folder with index.html. A requirements.txt file was also added, ensuring all versions were specified to avoid conflicts. This repository, available at <https://github.com/black-raven-001/zauba-scrape>, was used to set up a dummy Render account. The deployment process involved creating a new web service, linking the repository, and using the build command "pip install -r requirements.txt && npm install." After resolving all errors, the server.py was successfully deployed. This method was replicated to create 99 additional Render accounts, each following the same naming convention from scrap001 to scrap100, ensuring consistent deployment without further debugging.

To maintain server functionality, 50 cron-job accounts were set up to issue page requests every 10 minutes across the 100 deployed server.py instances. This was done to prevent downtime, as a lack of activity for 15 minutes could result in a server being down, necessitating a restart. The URL <https://scrap001.onrender.com/success> returns an HTML page, indicating that server.py is operational. This systematic approach ensured that 100 working instances of server.py were maintained across different addresses, providing a robust solution for distributed web scraping.

## **Batch processing:**

The process involves forming batches, with each batch containing 400 or 500 requests, depending on server capacity. These batches are further divided into smaller groups of four or five URLs, distributed across different servers to optimize parallel processing. Utilizing Python's “concurrent.futures.ThreadPoolExecutor”, multiple threads handle these requests simultaneously, enhancing efficiency and speed. Requests are allocated to different servers using a round-robin approach based on SN, ensuring balanced load distribution.

Using the previously created CSV file, batches of page URLs are formed by page ranges to first retrieve company URLs. At the time of storing these URLs using the “add\_new\_urls\_to\_detail()” function, each URL is assigned an ID. After storing all 27 lakh URLs, batches are created using these IDs, and the data is scraped and stored using the “add\_entry\_to\_full\_detail()” and “add\_director\_from\_dataframe()” functions.

During execution, each thread sends a request to the designated server and waits for a response. Upon processing all requests in a batch, results are collected and validated. If errors occur, such as network issues or server errors, the entire batch is discarded and retried after a specified interval (typically one minute), with a maximum of five attempts to prevent indefinite looping.

## **Efficient MySQL Data Storage:**

The DBConnection class efficiently stores scraped data into a MySQL database, managing connections, adjusting the schema dynamically, and ensuring data integrity with comprehensive error handling.

* Methods like “add\_new\_urls\_to\_detail()”, “add\_entry\_to\_full\_detail()”, and “add\_director\_from\_dataframe()” dynamically construct SQL queries from a provided dictionary of column names and values.
* For new columns, the system detects "Unknown column" errors and updates the table schema by adding the new column with a default value of NULL.
* To handle MySQL's 64-character limit on column names, long keys are shortened using a hashing mechanism, with mappings maintained in a runtime dictionary.
* If data exceeds column length limits, the method increases the column size and retries insertion, accommodating all data without truncation.

This robust mechanism ensures accurate and efficient data insertion into the database with minimal manual intervention, even when encountering new data fields and variations.

# **Limitations:**

* **Website Performance**: The Zauba Corp website was sometimes slow, taking a considerable amount of time to load pages. This latency impacted the overall efficiency and speed of the data collection process.
* **Data Availability**: Although approximately 27 lakh companies are listed on Zauba Corp, the website only provides access to 13,333 pages, each containing 30 company URLs. This limitation restricts access to about 4 lakh companies through this approach. However, different filtering techniques were employed to bypass this and access more company data.
* **IP Blocking:** Repeatedly requesting web pages without proper management can result in IP blocking by Zauba Corp's servers.
* **Data Access Limitations:** Access limitations for non-authenticated users restrict access to certain data fields that require user login.
* **Inconsistent Data Fields**: The data fields for companies were not consistent. Some companies had a Company Identification Number (CIN), while others had an LLP or Registration Number. This inconsistency required additional handling to capture all relevant data fields correctly.
* **Server Management**: The project utilized 100 servers to distribute the scraping load. If any server became unresponsive or was temporarily blocked by Zauba Corp, it required manual intervention to restart the server and resume scraping. Monitoring and maintaining these servers added to the complexity of the project.
* **Data Inconsistencies**: Despite collecting URLs using details.py, some URLs led to pages with the message "Company not found." This issue required additional handling to ensure data completeness and accuracy.
* **Volume of Data**: Dealing with 27 lakh companies involved fetching over 27 lakh pages and storing the data in a MySQL database. This massive volume of data made the process time-consuming, requiring the code to run continuously for 4 to 5 days, depending on the speed and reliability of the servers.
* **Dynamic Database**: The Zauba Corp database is frequently updated with new companies, meaning the data collected could quickly become outdated. At any given time, it was impossible to capture all companies. For example, after completing the scraping process, 27,19,728 companies were scraped out of the currently available 27,23,070, leading to a small percentage of missing data. Nevertheless, the accuracy achieved was more than 99.8%.
* **Database Structure Changes**: The variability in company data required dynamic changes to the database structure during runtime. New types of data necessitated adding new columns to the database, which was challenging given the large volume of existing data. Additionally, if a new value exceeded the predefined length for a column, the table structure had to be modified accordingly, complicating the data management process.
* **Resource Intensive**: Even with 100 servers running parallelly. the project required significant computational resources and careful orchestration. Any minor delay or change in server speed could extend the total processing time, illustrating the resource-intensive nature of large-scale web scraping projects.

# **Tools and Techniques**

* **Python**: Utilized for scripting and data manipulation tasks, encompassing the entire codebase from local file operations to the Flask web application.
* **Flask**: A lightweight Python web framework employed for constructing and hosting 100 web applications dedicated to online web scraping.
* **BeautifulSoup**: A Python library extensively used for parsing HTML and XML documents, essential for web scraping tasks.
* **Render**: Utilized for deploying and hosting 100 web applications and services. These applications accept URL inputs via POST requests and return scraped data.
* **GitHub**: A collaborative platform for version control and code management. Used for pushing updates across all deployed Flask applications.
* **Cron job**: A scheduler in Unix-like systems, used for automating repetitive tasks. In this context, cron jobs maintain the uptime of the 100 Flask applications by periodically requesting HTML pages every 10 minutes.
* **MySQL**: A relational database management system responsible for storing structured data. The entire database management and storage were handled using MySQL.

# **Output:**

After completing the data scraping, processing, and storing procedures, the team organized the information into two primary tables full\_detail and director.

**Final Tables:**

1. **Full\_Details Table**:

* **Content**: Contains comprehensive company information.
* **Rows**: 2,719,728 unique companies.
* **Primary Key**: Unique Serial Number (SN) for each company.

|  |  |
| --- | --- |
| SN | 1 |
| CIN | U74999MH2021OPC368405 |
| LLP\_Identification\_Number | NULL |
| Company\_Name | AM DAILY SERVICES (OPC) PRIVATE LIMITED |
| Company\_Status | Active |
| ROC | ROC Mumbai |
| Registration\_Number | 368405 |
| Company\_Category | Company limited by shares |
| Company\_Sub\_Category | Non-government company |
| Class\_of\_Company | One Person Company |
| Date\_of\_Incorporation | 29-Sep-21 |
| Age\_of\_Company | 2 years, 8 month, 8 days |
| Activity | Business activities n.e.c. |
| Number\_of\_Members | 0 |
| Authorised\_Capital | 500000 |
| Paid\_up\_capital | 500000 |
| Listing\_status | Unlisted |
| Date\_of\_Last\_Annual\_General\_Meeting | N/A |
| Date\_of\_Latest\_Balance\_Sheet | 31-Mar-22 |
| Email\_ID | info@amdailyservices.com |
| Address | 1,Floor-0,Plot-202/204,BhavnagariCottage S S Wagh Marg, Naigaon, Dadar (E) , Mumbai, Maharashtra, India - 400014 |
| Url | https://www.zaubacorp.com/company/AM-DAILY-SERVICES-OPC-PRIVATE-LIMITED/U74999MH2021OPC368405 |
| Date\_of\_last\_financial\_year\_end\_date\_for\_which\_Annual\_Return\_filed | NULL |
| Date\_of\_last\_financial\_year\_end\_date\_for\_which\_Statement\_of\_Accounts\_and\_Solvency\_filed | NULL |
| Description\_of\_main\_division | NULL |
| Main\_division\_of\_business\_activity\_to\_be\_carried\_out\_in\_India | NULL |
| Number\_Of\_Partners | NULL |
| Number\_of\_Designated\_Partners | NULL |
| Total\_Obligation\_of\_Contribution | NULL |
| Country\_of\_Incorporation | NULL |
| Foreign\_Company\_Registration\_Number | NULL |
| Type\_Of\_Office | NULL |
| As\_on | June 12, 2024 |

1. **Director Table:**

* **Content**: Includes information about company directors and their relationships to companies.
* **Rows**: 5,844,094 unique company-director pairs.
* **Mapping**: Directors are mapped to companies using the company's SN.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SN | DIN | Director\_Name | Designation | Appointment\_Date |
| 1 | 9338819 | ANIL ARUN MANE | Director | 29-Sep-21 |
| 1 | 9794864 | SNEHAL ANIL MANE | Director | 18-Nov-22 |

1. **Joined Table:**

**After joining the above tables on SN and matching them by the director's name provided, the resulting dataset shows all companies associated with that director.**

|  |  |  |  |
| --- | --- | --- | --- |
| Company\_Name | Director\_Name | Designation | Appointment\_Date |
| L&T SEAWOODS LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 29-Sep-15 |
| LTIMINDTREE LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 31-May-16 |
| LARSEN AND TOUBRO LIMITED | RAMAMURTHI SHANKAR RAMAN | Whole-time director | 01-Oct-11 |
| MINDTREE LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 16-Jul-19 |
| L&T REALTY LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 25-Sep-14 |
| L&T HYDROCARBON ENGINEERING LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 22-Aug-13 |
| L AND T EMSYS PRIVATE LIMITED | RAMAMURTHI SHANKAR RAMAN | Director | 17-Jul-09 |