**Scraping Process Describer**

*Author: Ewan Nikhil Thomas*

This document explains the scraping process designed for procuring data from the [NRLM Analytical Reports](https://nrlm.gov.in/outerReportAction.do?methodName=showReportMaster). It applies to the following reports: F1b, F1c, F2b, F2c, Fl2. The webpages include information on Revolving Funds and Community Investment Funds distributed to Self Help Groups (SHG). “F1c and F2c” reports have granularity up to village level and “F1b and F2b” reports have granularity up to SHG level. All the reports are available at monthly level from 2018-19 to 2022-23.

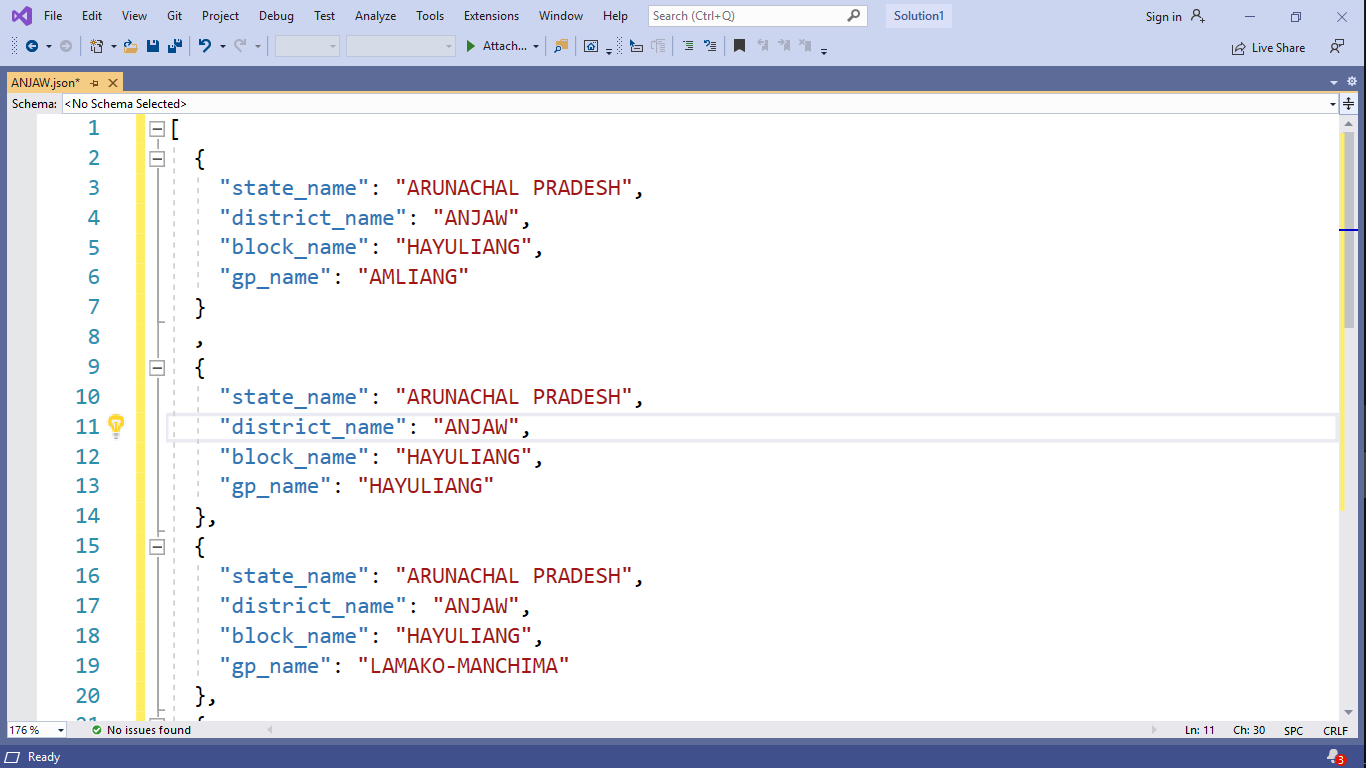
The scraper is designed to process in 3 separate stages using 3 python scripts:

1. **NAME SCRAPING**

script: “./src/data/{report\_name}\_name\_scraper.py”

The first process is name scraping or scraping of the names of geographical granularities which are active hyperlinks. That’s whenever there is a disbursement of funds to a village or SHG, the state, district, block, gram panchayat, village names mentioned in tables will be active hyperlinks. For months where no disbursements are made, the same names would be text. This is the key aspect of the webpage that’s guides the scraper in identifying the names to granularities to scrape. The name scraper looks for active hyperlinks in each page (state, district, block, gram panchayat, village) and extracts the text attribute or the names of these hyperlinks and then clicks on each of them to enter those granularities in the order of the items in the webpage.

The name scraper continues to scrape names and click on hyperlinks until the second last granularity of the report. We stop at second last because the last granularity page (village or SHG) will only be a text table and no hyperlinks. Each time the name scraper extracts a name and clicks on it, the same information is stored as a dictionary in the order of the granularity nesting. A sample image of the same is attached here:



Once all the active hyperlinks for each district is exhausted the entire bunch of dictionaries are appended in to single list for that district and exported as “.json” file to the sub directory “./data/raw/{time\_stamp}/jsons/{district\_name}.json” where time stamp is the concatenation of year and month concerned, e.g.: “2021\_22\_March”. Now once a file is created for a district and now consider that the scraper encountered an error and broke in between. When we restart the scarper, the name scraper has a checker (if condition) which looks for files with the above district name and ignores the district for which the “{district\_name}.json” exists and proceeds to scrape the rest.

***Point of Manual Iteration***

The reports are available at monthly level and the intensity of the scraping process in terms of time and volume of files and data created is high. So the person in charge of the scraper should keep a regular check on the scraper and whenever the entire scraping exercise for one month is completed, she may manually enter the year value and month value in the script under the heading “\*\*\*POINT OF MANUAL ITERATION\*\*\*\*”. She should also correct the “time\_stamp” value defined under the comment “# defining directories”.

1. **JSON FLATTENING**

script: “./src/data/json\_flat.py”

When the name scraper concludes, we must initiate the JSON flattening process. The JSON flattening process is a simple operation where in each {district\_name}.json file is called in appended into a single list and exported to the interim subdirectory under the path “./data/interim/{time\_stamp}/all\_names.json”. The “all\_names.json” essentially contains the names of all the geographical granularities which are active hyperlinks for that specific month and year. This file would provide support to the data scraper in terms of avoiding duplication of scraping files and information.

***Point of Manual Iteration***

The person in charge of scraping should correct the “time\_stamp” value defined under the comment “# defining directories”.

1. **DATA SCRAPING**

script: “./src/data/{report\_name}\_scraper.py”

The data scraping exercise for each month begins by importing the “all\_names.json” file and using the key values in each nested dictionary to call for hyperlinks from the webpage. There will only be one for loop in this entire process which will run through each dictionary in the “all\_names.json”. The scraper will build a file using the following generic approach:

“./data/raw/{time\_stamp}/{state\_name}/{district\_name}/{block\_name}/{gp\_name}.csv” or

“./data/raw/{time\_stamp}/{state\_name}/{district\_name}/{block\_name}/{gp\_name}/{village}.csv”

The scraper checks whether the file exists and if it doesn’t, then initiates a scraping process specifically for the dictionary from “all\_names.json” under consideration. Check the earlier image to see the keys and values for the dictionary are set. For reports except F1b and F2b, only the table in the last granularity, that is village level table is craped. However for F1b and F2b, tables at each granularity are scraped simultaneously and stored and checked for during re-runs of scraper.

By using the “all\_names.json” file at the beginning of the data scraping process, we know the exact locations that require scraping and even without calling the driver or the webpage, we can check whether those files exist or not. This a huge time saver especially when you are rerunning your scraper after an unexpected break. The data scraper will initiate webdriver call only when it observes that a file of the above path structure doesn’t exist.

***Point of Manual Iteration***

The reports are available at monthly level and the intensity of the scraping process in terms of time and volume of files and data created is high. So the person in charge of the scraper should keep a regular check on the scraper and whenever the entire scraping exercise for one month is completed, she may manually enter the year value and month value in the script under the heading “\*\*\*POINT OF MANUAL ITERATION\*\*\*\*”. She should also correct the “time\_stamp” value defined under the comment “# defining directories”.

**Order of Implementation**

The order of implementation of the scraping process is as explained as above.