**Web Scraping Scripts Generator**

A Report for the Evaluation 1 of Project 2

**Akhand Mishra**

Admission No.: 15SCSE101052

Under the Supervision of

**Mr. Surender Kumar**



**School of Computing Science and Engineering**

**Greater Noida, Uttar Pradesh**

**2018**

**Table of Contents**

1. **Abstract**
2. **Introduction**
   1. Problem Statement
   2. Overall Description
   3. Purpose
   4. Motivation and Scope
3. **Proposed Model**
   1. System Architecture
   2. User Interface Design
4. **References**

**Abstract**

*Web Scraping scripts generator acts a Universal tool to generate web scraping scripts for any website out there. The tool thus, removes the need to analyze different websites separately and then generate scripts for the same. The GUI will help the users that aren’t familiar with HTML, to build scrapers and gather data. Thus, the best use-case for this tool will be in the field of Data Engineering, where it is required to gather data from different parts of the web. Due to this Scraper Generator, any personnel even with no Knowledge of HTML can build scrapers to collect data from different websites. Most of all, this tool will help the Web Scrapers and work as a template to generate scrapers for different websites in a matter of few clicks.Enter the world of Artificial Intelligence; data has become valuable to organizations and developers. It is apparent that the biggest source of data is the web and hence, the need for scraping different websites has arisen. Developers need an automated way out of the problem of building separate scripts for different website. Thus, this software is a must*

**Introduction**

**2.1 Problem Statement**

Writing web scrapers in Python is a cumbersome task; moreover writing a web scraping script requires *in-depth knowledge of HTML*, an eye for *pattern recognition* and most importantly, **Python Language**.

The structure of each *HTML* document for *different websites is different* in all ways and thus, one must write a separate scraper for separate websites. So given a data collection project with 6-7 websites, will **increase** the amount of unnecessary work for developers.

A perfect solution would be the one which provides an **interactive** way to write web scrapers for different website even **without** the knowledge of web or Python.

**2.2 Overall Description**

**Universal Web Scraping Scripts generator** is a **desktop GUI tool** that aims to make the task of **writing** **web scrapers and crawlers**, easy.

First, let us try to understand what Web scraping or crawling is, and why it is required in today’s technological era.

**Web scraping or crawling**

* It is a field that deals with parsing any webpage on the internet and *retrieving useful information* from the same.
* The purpose of a scraper is to fetch different kinds of data from the website, such as tables, images, hyperlinks or other embedded files.
* Writing a scraper would simply automate the task to fetch the data from website provided that the website adds data dynamically every day.
* For example, a company wants to track the activity about their product on Social media, they can write an automation scraper which will scrape the posts with the organization’s product name in it and fetch the comments, images and links related to it.

Web scrapers at first sound very intimidating as they are built with the core concepts of Automation and even follow the principles related to it. However, web scraping scripts have their own **disadvantages**.

* Every website out there has a differently structured HTML and hence the scraper for every webpage has to be different.
* If the website owner decides to change the HTML structure later on, which happens in most cases, the scraper must be re-written to work again.
* Some websites out there do not allow robots (or automated scripts) to see the HTML of the website, in short for some websites, web scraping is illegal.
* Use of web scraping may diminish as the era of API breaks through.

The tool, Universal Web Scraping Scripts Generator is completely based on the concept of Web scraping and crawling. It limits the disadvantage of pure web scraping as the tool provides an interactive way to write web scraping scripts for different websites. As we have already discussed that how different websites have different structures, the skill of web scraping deals with recognition of this structure. To acquire this skill one must learn both, HTML and Python. This is where my tool steps in.

* Universal Web Scraping Scripts generator takes in a URL as input.
* After which it verifies the robots.txt file for that URL (i.e. it verifies whether it is legal to scrape the website or not).
* Post verification it fetches the HTML for that URL and executes any scripts.
* It then constructs a DOM tree for the HTML and displays all the web elements in form of button to the user.
* After a click on the button, the scripts generator starts to work and it generates a Python script for the clicked element.

**2.3 Purpose**

The purpose of Universal Web Scraping Scripts Generator is:

* To make it easy for developers to write web scrapers for different website.
* To provide an interactive way to generate Python scripts even without any knowledge of Python or HTML.
* Base for high-end developers to edit and add on the scripts on the go and even change it easily as per the changing structure of websites.

**2.4 Motivation and Scope**

**Motivation**

Enter the world of Artificial Intelligence; data has become valuable to organizations and developers. It is apparent that the biggest source of data is the web and hence, the need for scraping different websites has arisen. Developers need an automated way out of the problem of building separate scripts for different website. Thus, this software is a must.

**Scope**

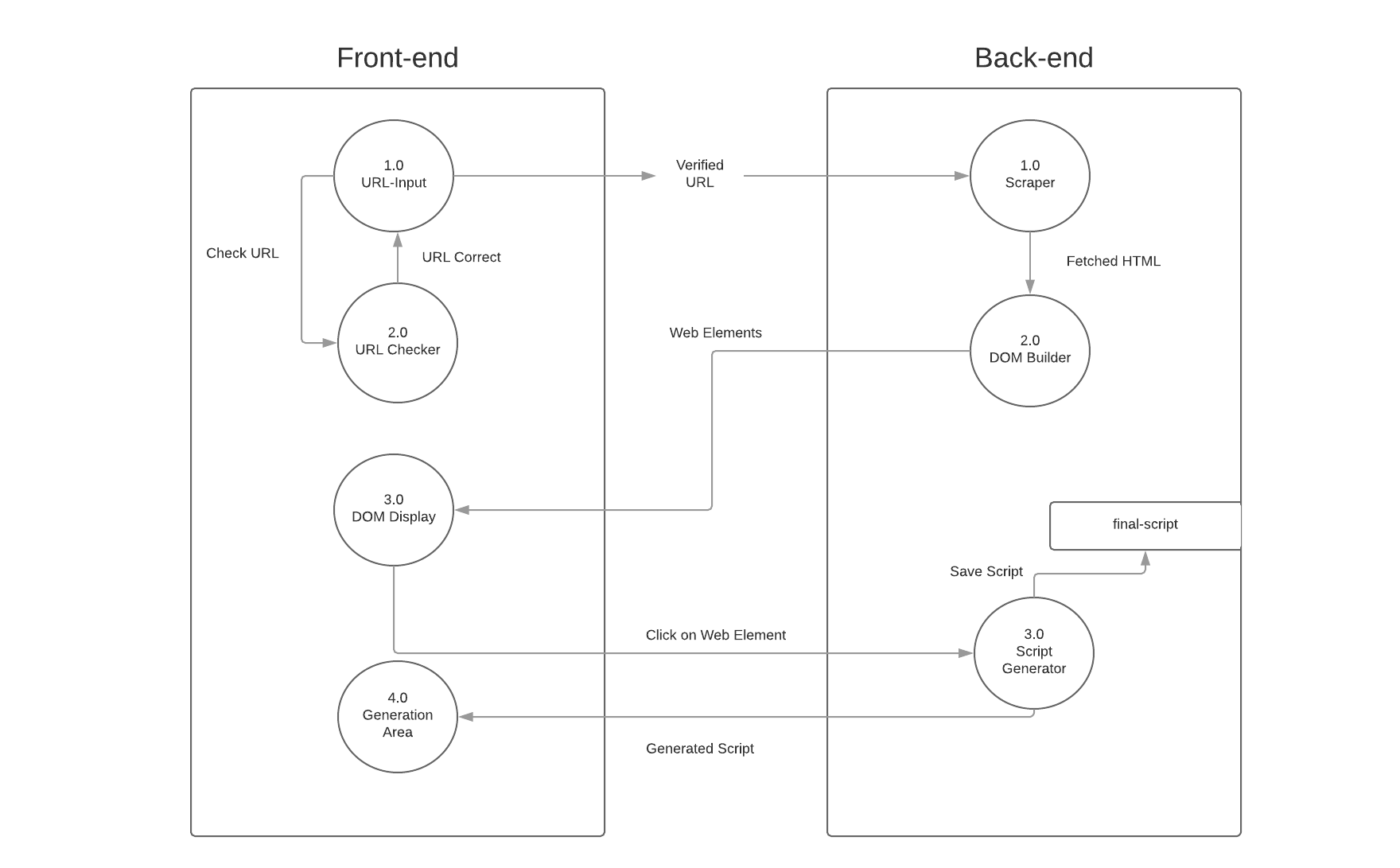
This project will consist of creating a desktop GUI tool that is based on the concept of Web Scraping and Crawling. It deals with the process of automating and easing the writing of scraping scripts in Python without any prior knowledge of HTML and Python. The desktop tool shall be a click and create type. Once completed, the project shall allow the user to create web scraping in just a matter of clicks, also it shall allow the user to track the changes made by the owner on the website or any dynamically added content.

**Proposed Model**

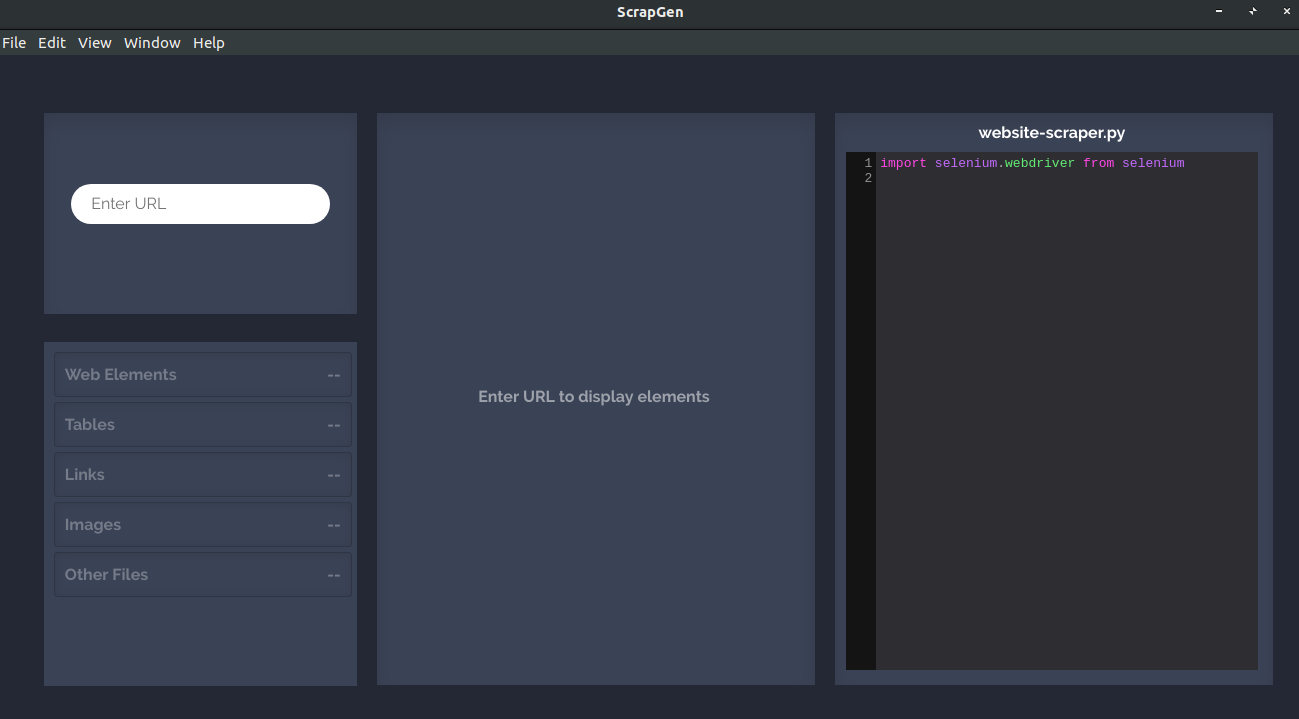
**3.1 System Architecture**

The software is structured to follow a simple Front and Back End Communicative architecture. The front-end comprises of the UI and the components for the user to interact with the software. The back-end has scripts which are triggered when the user clicks on a specific area in the front-end. Following is the detailed description of the tools and components involved in the Front-end and Back-end:

* **Front-End:** The front-end, as mentioned above, consists of the GUI and the tools for the user to trigger the back-end scripts. Front-End is written completely using the modern tool named as Electron JS, which allows one to use the knowledge of HTML, CSS and JS to develop desktop apps.
* **Back-End:** The back-end, which is packed within the software package is packed with many Python scripts such as the scraper.py to fetch the HTML for the webpage and many others. These back-end scripts get triggered depending upon the actions of the user on the front-end. The output of these scripts is channel to the JavaScript file using a library named “python-shell”.



**3.2 User Interface Design**

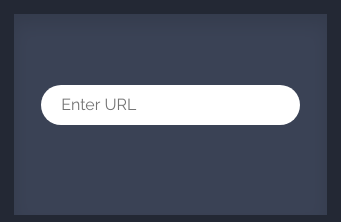


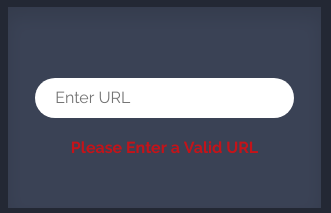
The app is single paged.



It contains 3 containers namely: URL Input, Display Area & Generation Area



**The URL Input Box**



Consists only of an input element to enter the URL for the website to be scraped.



The user will start the usage of the application from here.



The URL Input box will display an error message if the entered URL is invalid in any way.

Once you have corrected the URL, the message will disappear.



User can press the "Enter" key to submit the URL for being scraped.



Since the application is only made for PC's the URL input box won't work on mobile devices.



**The Display Area**

This is the area where user will interact with the DOM elements of the web-page to be scraped.



The area is empty when the app loads.



As soon as the user presses the Enter key the DOM elements of the entered URL will be loaded on the Display area in form of buttons.



These buttons are clickable.



The buttons display the heading/name of the HTML element.

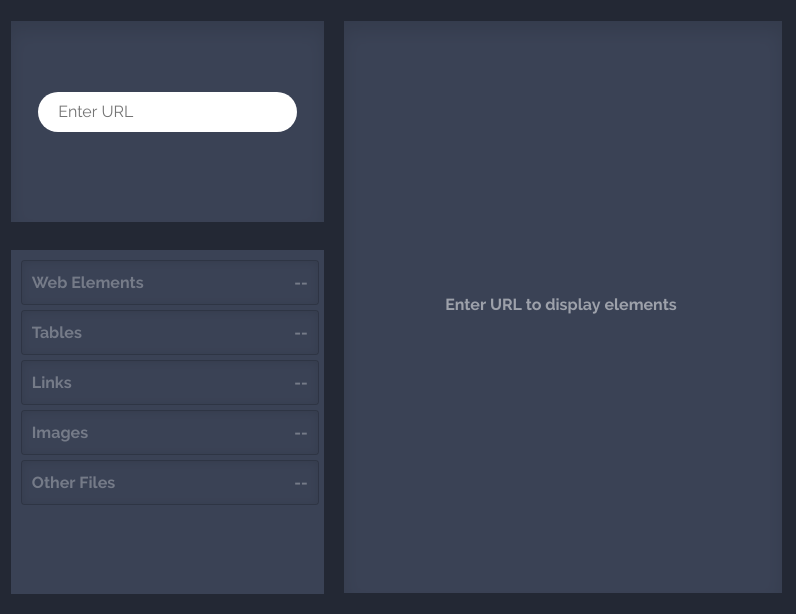


The buttons also display 3 of the attributes of the HTML element, such as "id", "class", "src" (for image elements) and so on.



On clicking any of the buttons, respective script to parse the clicked element is inserted in the Generation Area.





**The Generation Area**

This is the area where the script is generated.

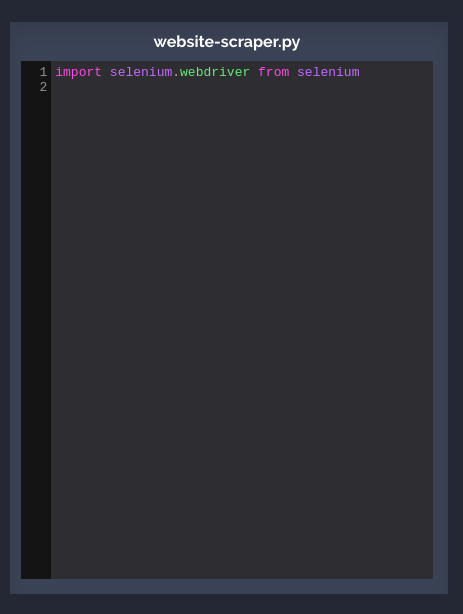


The script generated is editable



The editor is embedded using the CodeMirror Tool.



The script for each HTML Element will be followed by a comment which tells about the name and type of element.

**References**

1. **Selenium Documents:** These documents gave an in-depth description on how to use the selenium library with Python. (https://www.seleniumhq.org/docs/)
2. **BeautifulSoup Documentation:** These documents gave an in-depth description on how to use the bs4 library with Python. (https://www.crummy.com/software/BeautifulSoup/doc)
3. **Web Crawler Research Paper:** This document formed the base of the idea. (<https://www.researchgate.net/publication/254460232_Web_crawler_research_methodology>)
4. **Electron JS Videos:** This video gave abstract and essential knowledge to build UIs with Electron JS. (<https://www.youtube.com/watch?v=kN1Czs0m1SU>)
5. **Lucidchart:** An online tool to prepare project related diagrams. (<https://www.lucidchart.com/>)
6. **Atlassian:** An online tool for DevOps operations of a project. (https://www.atlassian.com/)