**Documentation for Samsung Refrigerators Data Extraction.**

1. **Bot Overview:**

This Python project aims to automate the extraction of data from the Samsung website for refrigerators. The bot uses web scraping techniques to navigate through the website, collect information about different refrigerator models, and store the data in an Excel spreadsheet. The purpose of the bot is to save time and effort in manually gathering data for analysis or other purposes. It targets the specific task of extracting information about Samsung refrigerators from the website.

1. **Process Description:**
2. **Initialize:**

* The bot starts by initializing the necessary variables and file paths.
* It sets the driver variable as a Chrome web driver instance and defines the excel file path for the Excel spreadsheet where the extracted data will be saved.
* It also sets the website URL as the URL of the Samsung website and serial no as the initial serial number.

1. **Run:**

* The run() method is called to execute the bot's workflow.
* The bot maximizes the Chrome window and navigates to the Samsung website using driver.get(website\_url).
* The click() method is called.

1. **Click:**

* The click() method performs the following steps:
* Finds the "Home Appliances" menu button using an XPath selector and clicks on it.
* Finds the "All Refrigerators" link using an XPath selector and clicks on it.
* Performs multiple scrolls and clicks on the "View more" button to load all refrigerator models.
* The scroll and click actions are performed in a loop until either 5 scrolls are completed or no "View more" button is found.

1. **Process:**

* The process() method executes the main data extraction process:
* Scrolls to the top of the page using driver.execute\_script ("window.scrollTo(0, 0);").
* Calls the get\_number\_of\_models() method to get the total number of refrigerator models displayed on the page.
* Iterates from 1 to the number of models:
  + Finds and clicks on the "View more" button for each refrigerator model using an XPath selector.
  + Calls various methods to extract specific data for the current model, such as title, price, capacity, dimensions, weight, and rating.
  + Prints the extracted data.
  + Writes the extracted data to the Excel spreadsheet using the write\_to\_excel() method.
  + Increments the serial\_no for the next model.
  + Finds and clicks on the "Close" button to close the current model.

1. **Data Extraction Methods:**

The bot uses multiple methods to extract specific data from the refrigerator model details:

* extract\_title(): Extracts the title of the model.
* extract\_price(): Extracts the price of the model.
* extract\_capacity(): Extracts the capacity (net total or gross total) of the model.
* extract\_width(): Extracts the width (net width or net dimension) of the model.
* extract\_depth(): Extracts the depth (net depth with or without door handle, or net dimension) of the model.
* extract\_height(): Extracts the height (net case height with hinge or net dimension) of the model.
* extract\_weight(): Extracts the weight (net weight) of the model. extract\_rating(): Extracts the rating (energy star rating or energy grade) of the model.
* These methods use XPath selectors to locate the corresponding elements on the page and extract the text values.

1. **Write to Excel:**

* The write\_to\_excel() method writes the extracted data to the Excel spreadsheet.
* If the spreadsheet file exists, it loads the workbook using load\_workbook() and gets the active worksheet.
* If the spreadsheet file does not exist, it creates a new workbook and adds a worksheet.
* It appends the extracted data as a new row in the worksheet.
* Finally, it saves the workbook using save().

1. **Bot Architecture:**

* The bot is implemented using the Python programming language and relies on several external libraries, including Selenium and openpyxl.
* It uses Selenium's WebDriver API to interact with the Chrome web browser and perform actions like clicking buttons, scrolling, and extracting data.
* The openpyxl library is used to read from and write to Excel spreadsheets.

1. **Input and Output File Paths:**

* Input File: The input file path for the bot is the Haier website URL “https://www.samsung.com/in/”.
* Output File:

The output file path for the bot is the Excel spreadsheet

“E:\Kintan\POC2-210623\Samsung\_300623.xlsx” where the extracted data will be stored.

1. **Requirements:**

* Python 3.x.
* PyCharm (Version- PyCharm 3.9).
* Selenium library (Version- Selenium 4.0.0).
* Chrome web driver (compatible with the installed Chrome browser version).
* openpyxl library.(Version- openpyxl 3.x)
* PyCharm (Version- PyCharm 3.9).