**Automated Driver Download Tool - Documentation**

* **Why this project?**

This project serves the scope to help the user automate the download of new drivers that come out for their printers/faxes/scanners

* **Must have:**

1. Python: The programming language used to run the script.

2. Selenium WebDriver: A library for automating web browsers.

3. Google Chrome: The web browser used by the WebDriver.

4. ChromeDriver: A tool that connects WebDriver with Chrome.

* **Functionality**

1. Data preparation:

- The tool reads driver information from the provided Excel file, mapping the columns to English names for consistency.

- The data is converted to a JSON string, parsed into a list of dictionaries for easier processing.

2. Driver information processing:

- The tool processes driver names, extracting product names and removing unnecessary characters like '(WHQL)' from product names for creating the proper folders.

- It checks for the availability of driver updates and selects the latest version for each driver name.

3. Search and Download:

- The tool uses Selenium WebDriver to navigate to the FujiFilm website and perform driver searches based on the driver name.

- It automatically selects the appropriate operating system, operating system version, and software type for each driver based on the architecture (32-bit or 64-bit).

4. File Management:

- The tool creates the necessary folder structure to store downloaded drivers, organizing them by product name, version and architecture.

5. Error Handling:

- The tool logs exceptions that occur during the driver download process, indicating which drivers encountered issues.

6. Main Entry Point - 'process function' -:

- The 'process' function serves as the main entry point of the tool.

- It procesess driver information for both architectures based on the sheet name provided ('32bit' or '64bit').

- The function calls various helper functions to perform specific tasks and handle different driver types.

* **Version 1.0.0**

Steps on how the project was done:

1. Created a CSV file with the names of the drivers and the operating systems

2. Read the driver names and operating systems and appended them into empty lists

3. Opened the website https://www.fujifilm.com/fb/download?lnk=header

4. Created a program that runs in a for loop that automatically searches for the driver name, enters the OS, OS Version and the software

type and downloads the necessary driver

5. The program renames the downloaded drivers under the format "driverName\_operatingSystem.exe" if the OS is windows and

"driverName\_operatingSystem.dmg" if the OS is Mac OS

* **Functionality for version 1.0.0:**

1. Open Chrome

2. Go to https://www.fujifilm.com/fb/download?lnk=header

3. Search for the driver name

4. Enter the OS/OS Version and software type

5. Download the driver and rename it under the format "driverName\_operatingSystem.exe" if the OS is Windows and

"driverName\_operatingSystem.dmg" if the OS is Mac OS

* **Version 1.1.0**

1. Implemented the option to read the operatingSytemVersion and the softwareType from the csv instead entering them directly into the code

2. Automated the installation of the drivers until the UAC page

3. Modified the rename option: Now the program checks if you already downloaded the driver and if so, the new file will be renamed to "driverName\_operatingSystem\_softwareType\_timestamp.exe/dmg"

4. Added 4 more drivers to the csv file. Now the program also downloads printer and scanner drivers instead of only faxes.

* **Version 1.2.0**

1. The code reads the drivers from an excel file (only from 64bit sheet for now) and parses the data into a JSON-LD

2. Check for the driver name on the website and search for the latest version. If there are multiple versions in the excel, it will choose

only the latest version.

3. Download the driver and move it into PSLAD3->64bit->productName->version

4. The program assumes that all the drivers were made for printers.

* **Version 1.2.1**

1. Created a function that takes as an argument the sheetName from excel and downloads the drivers from that specific sheet.

2. Implemented a condition to check what type of driver needs to be downloaded (fax or printer)

//Bugs to fix:

1. When the code reads a driver meant for faxes (example: FX driverName FAX), it should only search for driverName – FIXED

* **Version 1.3**

**Major updates:**

1. Platform support

- The tool now supports both 32-bit and 64-bit architectures. It processes driver information for each architecture separately, allowing users to download the appropriate drivers based on their system requirements.

2. Enhanced driver selection

- The tool uses StrictVersion comparison to ensure that only the latest driver version is processed and downloaded for each driver name. This

prevents unnecessary duplicate downloads and keeps the system up-to-date.

3. Improved file management

- The tool now creates the necessary folder structure to store downloaded drivers for each architecture (32-bit or 64-bit). Drivers are organized by product name, version and architecture in their respective folders.

4. Expanded driver types

- In addition to fax drivers, the tool now supports the download of printer and scanner drivers. The tool automatically determines the driver type based on the provided driver name and selects the appropriate software type on the FujiFilm website.

5. Excel data processing

- The tool reads driver information from an Excel file, extracting data from both the 32-bit and 64-bit sheets. It then converts the data into JSON-LD format for easier handling and processing.

6. Enhanced error handling

- The tool implements error handling mechanisms to log any exceptions that occur during the driver download process. Failed downloads are logged in the file 'failed.txt', while succesful downloads are logged in the file 'success.txt'.

7. Streamlined Process function

- The 'process' function' has been refactored to handle driver information for both 32-bit and 64-bit architectures separately. It now accepts the sheet name as an argument and procesess the coresponding drivers accordingly.

* Version 1.4

Added an HTML table that shows the user the status of the downloaded drivers. If everything was fine, it shows ‘ok’ and no remark. If the program encountered an error, that status changes to ‘not ok’ and leaves a message in the remark section informing what went wrong.