Retail Automation Toolbox

Version 0.0.1

Jacob Kobs

1. Project Overview

1.1 Brief Description

I'm sure many, if not all of you, have registered for an account on a retail site, such as Target or Walmart. This project mainly focuses on those functionalities, but with an automated approach. In the botting world, you need to generate hundreds of accounts for a better chance at copping today's most in demand items. If you were to manually register for accounts, it would take many valuable hours, whereas with a registration bot, all you need to supply is valid proxies, unique account usernames and passwords, letting the registration software do all the heavy lifting for you.

1.2 Technologies Used

- **Puppeteer** – A fantastic automation library that's mainly used for testing the functionality of a web page to ensure that it's user friendly and behaves as expected, such as when a user clicks a link, it redirects them to the expected page. It's also an easy way to bypass Target's anti-bot security system.

- **Node.js** – A backend JavaScript runtime environment that runs on Chrome's V8 engine and allows JavaScript code to run outside of a web browser. Since Node.js is primarily used for non-blocking, event-driven applications, it's perfect for automation because the toolbox is event-driven, and code execution is based on what specific event occurs. For example, say we try to create an account that already exists on Target. Target will return an error message stating that the account is already in use. From there, we can have our application redirect the flow to another event based on said error, which also acts as a failsafe for the program.

- **BASH (Bourne Again Shell)** – BASH is a cli tool that makes file navigation very simple. Not much of a tool that the user will use but will help speed up programming. It also has git integration, allowing us to quickly connect to GitHub by logging into our account via cli.

- **HTML, CSS, and React (usage is TBD)** – Initially, the application will strictly be through a terminal. If time allows, it may be converted into a desktop web application. React would be a perfect JavaScript library to use. It enables the use of reusable components on webpages, allowing code reuse, which saves time in the long run. HTML and CSS will also be used in conjunction with React to style and display elements on the webpages.

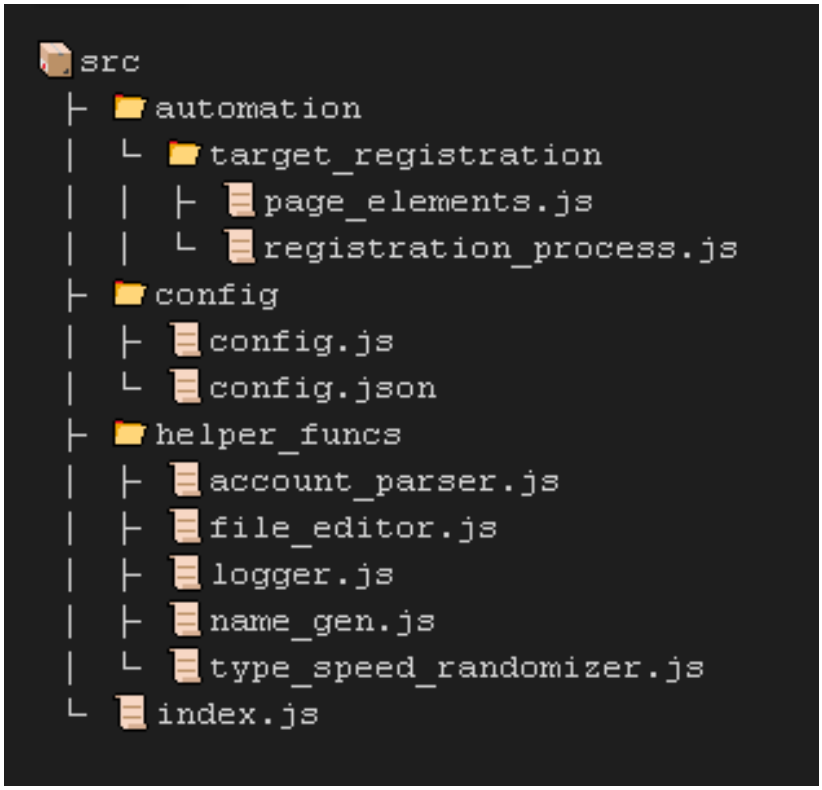
- **Electron (usage is TBD)** – Electron allows you to turn a web application into a desktop application. It still acts as a web browser but can be loaded directly from your desktop and allows you to disable the inspect element commands as well as hides the URL bar, giving a more professional feel that you would see on a typical desktop application. If you've ever used Visual Studio Code, then you've seen React and Electron in action!

- **MongoDB (usage is TBD)** – MongoDB is a quick and efficient way to store data while using less space compared to SQL-based relational database management systems, as it doesn't use SQL (NoSQL). If implemented, Mongo will be used to issue and hold authentication keys for access to the bot's interface. Other than that, all information will be stored on the user's local machine to help encapsulate sensitive data/files.

1.3 File Structure

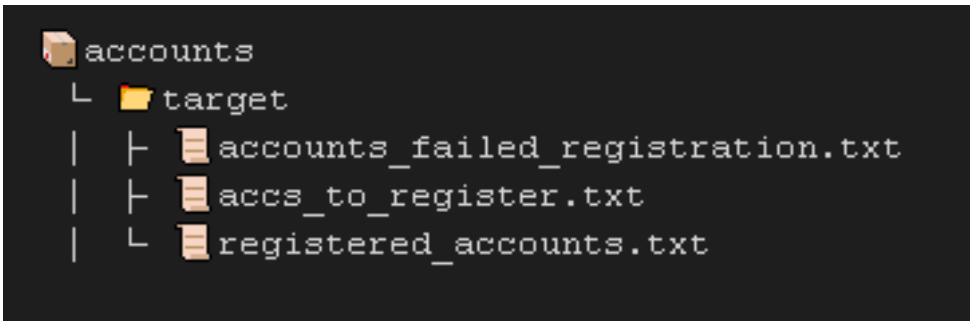
SRC Directory –

Where all execution/logic takes place.



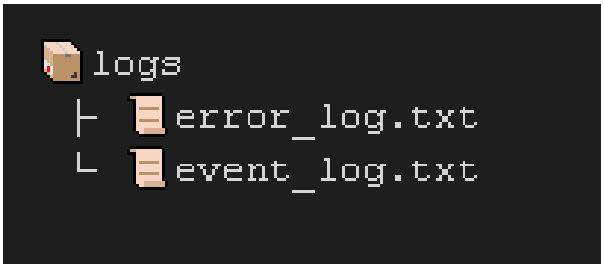
Accounts Directory –

Keeps track of all account credentials. Other sites will be added. Currently working on target.



Logs Directory –

Mainly for debugging purposes. Logs keep track of bot events and potential errors (both user and bot related).



Overall Project Structure –