**House Host Tracking Advertiser**

Table

Description automatically generated

# Description

I took it upon myself to develop new features whilst troubleshooting and resolving any issues that arose on a sophisticated auto typing software application used by selected clients. The foundation was coded in Java by Mr Dakota N., the original author of the application, to allow selected clients to output customised messages on RuneScape, a massively multiplayer online role-playing game (MMORPG). With his approval, I was granted full ownership of the application and devoted many hours improving the codebase using Eclipse integrated development environment (IDE). It consists of over 3000 lines of code, where over 90% of the codebase is a result of my work.

To this day, I continue to improve my knowledge in Java and optimise the codebase to ensure it is easier to maintain and advance. The application grabs data in the form of JavaScript Object Notation (JSON) from a webpage every 30 seconds, formats it into a table, to then output it in-game. It involves many variables, classes, methods, array lists, and more whilst handling several threads to achieve asynchronous behaviour. Clients could contact me at any time via Discord to provide feedback or report any bugs that they may encounter.

The world and location could be selected in a dropdown menu by the client to advertise correct Gilded altar hosts. The webpages were populated from a Discord bot that allowed people to type in a command when they were hosting. For example, someone could type *!Host W330 Rimmington OSRS*, and the Discord bot would automatically store that information and update the JSON file, located on the Altar.rs webserver.

To further elaborate, upon starting the auto typing application, a .txt configuration file would automatically generate on the client’s desktop if it did not already exist. This file allowed clients to customise everything i.e., the messages, colours and/or effects, stop timer, message typing speed, message delay and more! After the introduction of many new features and preventative solutions, they could customise everything while the application was running real-time. Any time a client encountered issues or errors while using the application, an error file would automatically generate on their desktop.

# Initial Start-up

1. Client started up the application.
2. The application then checked for a .txt configuration file on the desktop. It supported both MacOS and Windows platforms.
3. If the configuration file was not located on the desktop or if it was corrupt, it automatically created a new one.
4. If the configuration file was located on the desktop, the program read the settings inside the file and updated accordingly.

# Functionality

This application grabbed website data in the form of JavaScript Object Notation (JSON) from specific URLs, and did the following:

* If a client’s RuneScape display name was on the blacklist, they could not be outputted/advertised.
* Read all the names, locations, and worlds. It then formatted that data into a table. This is shown on the program’s main interface.
* Read and applied any new changes from the configuration file and/or graphical user interfaces and updated the configuration file accordingly.
* Every 30 seconds by default, the program updated the host table automatically by looking up the URLs and grabbing the webpage data.

In addition to that, there were many customizable options to choose from before starting the advertising process:

* **Pin To Top:** Program window appears on top of every other window.
* **Only Verified Hosts:** Only advertises people that have verified themselves on our Discord server. This process involves a bot visiting their RuneScape high scores URL and checking if they have a high enough Construction level for a Gilded altar.
* **Advertise Tracker Hosts:** Whether to advertise the people that are hosting.
* **Advertise Community Lines:** Whether to advertise the community lines listed in the configuration file or on the Advertisers interface.
* **Advertise Bone Job Lines:** Whether to advertise the bone job lines (decommissioned discord community) listed in the configuration file or on the Advertisers interface.
* **Colours and Effects:** Choice of what colours and effects to use while it is advertising.
* **Configs:** Settings you can customise to adjust the typing speed, the delay between each line, the time to stop advertising, and how many lines you wish to advertise hosts before typing a community or bone job line.
* **Advertisements:** Allows you to edit any line that gets advertised i.e., lines that are advertised when there is no host, community lines, and bone job lines.

**Screenshots**

**Graphical user interface, text, application, email

Description automatically generated**

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated