LOCAL URL SHORTENER WITH WEB INTERFACE

By: Atharv Dubey (NITJ '28)

General Information:

The local URL Shortener works by hosting local servers onto the user system which listen to calls to 'localhost:5000' and 'localhost:8000' where the former handles the redirection logic and the latter deals with transfer of input and output from front-end to back-end and vice versa. The project is built to learn about transfer of information from one environment to the other and to get an elementary idea about how a server functions. The shortener works on the concept of **Base62 Encoding**. One of the initial aims was to integrate a database system to store information (sqlite), however due to time constraints the project uses a simple text file to store long and short URLs currently, which will eventually be updated to use the intended systems with later version(s). The next update will implement various 'premium' features, one such feature is custom URLs for paid subscribers.

Languages Used:

• C++	Handles the URL shortening logic
• Python	Both the server scripts are built using Python
• HTML/CSS	To build and design the web interface
• JavaScript	Provides functionality to the webpage form

External Libraries Used:

• Python	'websockets' and 'flask'

Prerequisites:

- A working python installation (Download Python | Python.org)

Note: If any external library is missing from your system, the server scripts will automatically download and install the missing library given an active internet connect and that the python installation is added to 'PATH' in system environment variables.

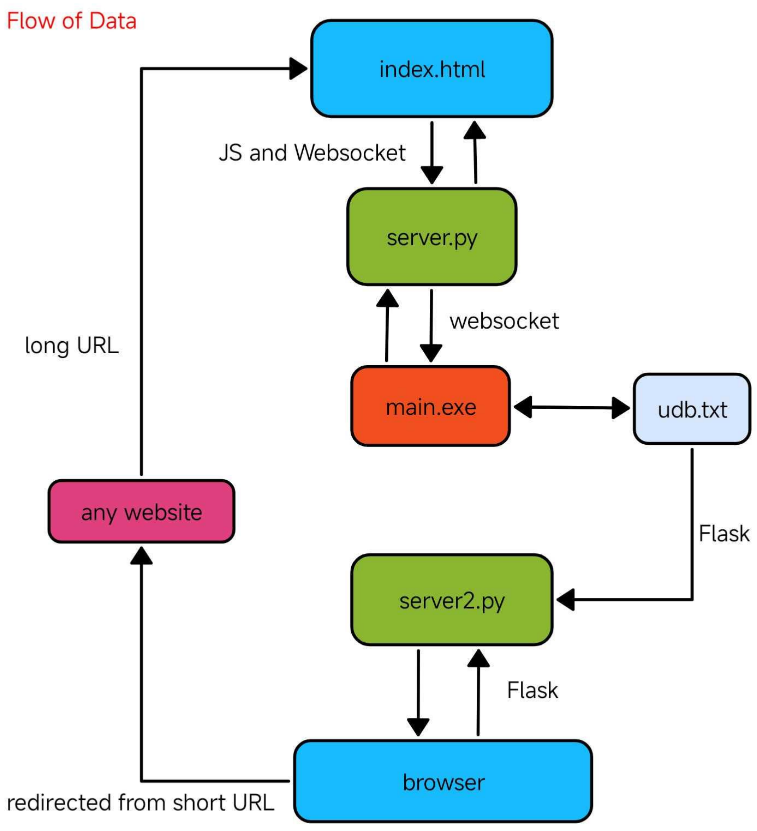
Usage Instructions:

- Run 'server.py' and 'server2.py' scripts. This will load the two servers and a console for each server will be visible.
- Launch 'index.html' webpage using any browser.
- Input a URL and press 'Shorten' button, this will show a shortened URL of format 'localhost:5000/(placeholder)'.
- Copy this URL and input it into the search bar of a browser while 'server2.py' is running. This will redirect the user to the original URL.

Additional Notes:

- The “udb.txt” file contains all the information in a CSV format (long URL, short URL, unique ID). You can also modify the text file by adding your URLs manually to it and the short URLs would work as expected as long as “server2.py” is running.
- In case you encounter unexpected errors and (or) are modifying the source code, you can enable debug mode in “server2.py” by changing ‘(debug=False)’ to ‘(debug=True)’ in the last line of code, this will generate a debug key that will be visible on the console.
- You can also rename the server names or ports to your liking by making the appropriate changes in the server and HTML scripts. Since the “main.exe” only deals with the latter part of the short URL (localhost:5000/**(placeholder)**), you do not need to change “main.cpp” file and hence don’t need to recompile the program every time you change server ports/names.
- The premium features as described in the ‘General Information’ section is about ~90% coded and is present in the main script, however, they are disabled within the code. To activate them, you can simply make minor changes to servers to allow multiple inputs and make the CPP script take appropriate inputs to try the custom URL feature.

Flow of Data:


Flow of Data
index.html
JS and Websocket
server.py
websocket
long URL
main.exe
udb.txt
any website
Flask
server2.py
Flask
browser
redirected from short URL