Proof Of Concept: Homoglyph Shortener

Submitted to: Digisuraksha Parhari Foundation
By: P.Harini, Intern ID: 385
Team Name: VIPER, (*Vastly Intricate Protocol Enforcement Regiment*)
Members: Harini Porumamilla (385), Adwitya Deep Verma (186)

About the Tool

- **Name**: Homoglyph Link Shortener.
- **Creator**: Adwitya Deep Verma, Intern ID: 186.
- **Description**: This is a Python-based command-line tool designed for generating shortened urls, Homoglyphic in nature, obviously for learning purposes-

How It Works

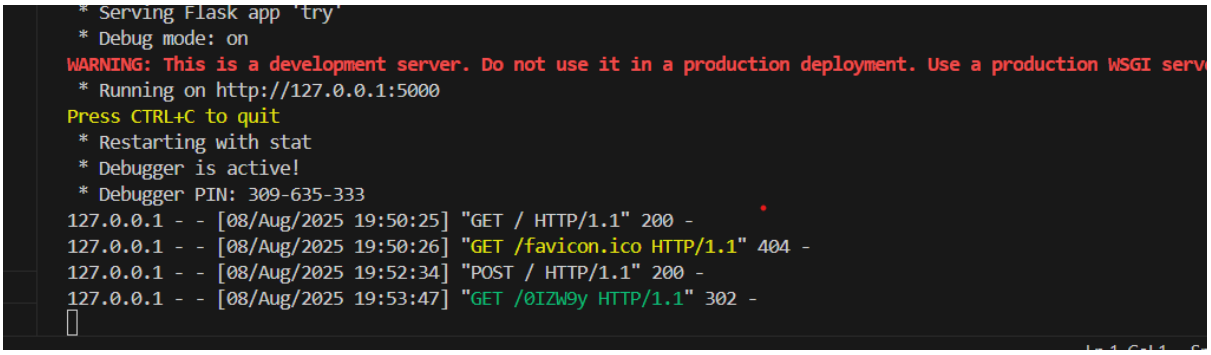
The system uses the Flask framework to create a web application with a simple interface for shortening URLs. The key components are:

- **URL Shortening**: The index() function handles both the display of the shortening form and the processing of submitted URLs. When a user enters a long URL, the application generates a unique, random 6-character alphanumeric slug using the `generate_short_slug()` function.
- **Database**: The application uses a SQLite database (urls.db) to store the mapping between the original long URL and its unique short slug. The `init_db()` function ensures the urls table exists on startup.
- **Redirection**: When a user accesses a short URL (e.g., http://localhost:5000/aBcDeF), the `redirect_to_long_url()` function retrieves the corresponding long URL from the database and redirects the user to it.
- **Homoglyph Demonstration**: For demonstration purposes, a separate `create_homoglyph_string()` function replaces specific characters in a display URL (e.g., "youtube.com") with visually similar "homoglyph" characters from other languages, like Cyrillic or Greek. This creates a fake, non-functional link to illustrate the deceptive nature of homoglyph attacks.

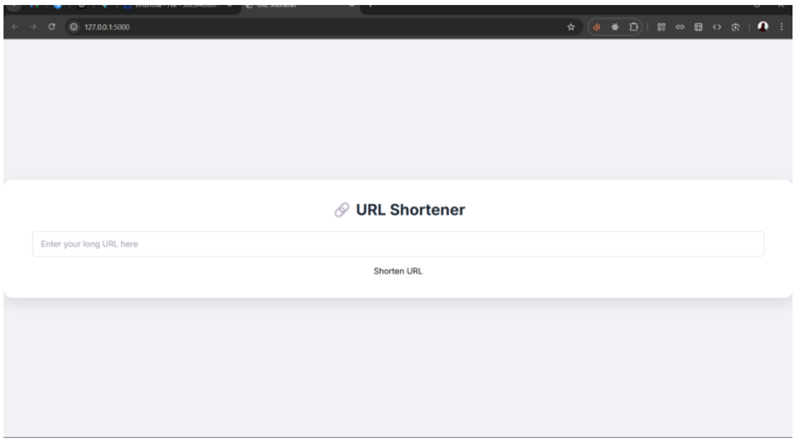
Screenshots of working:

Firstly we will run our program, and it gives us a local server environment, using flask and sql database.

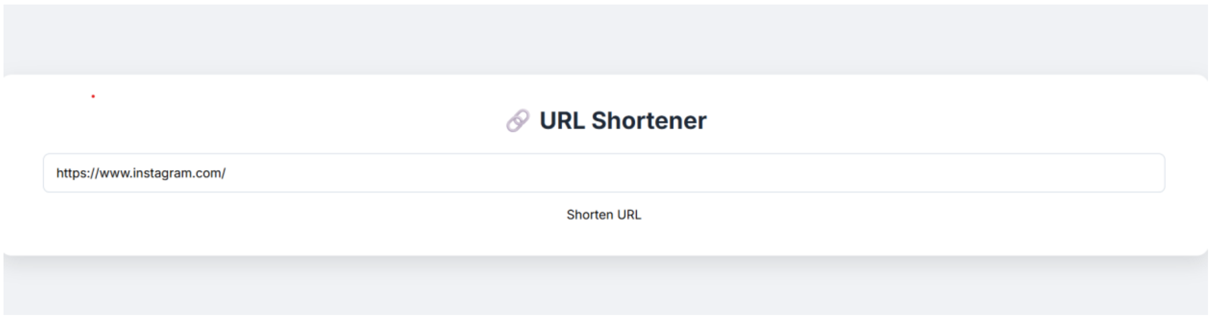
This gives us an address to open our webapp, to enter and get our shortener going.



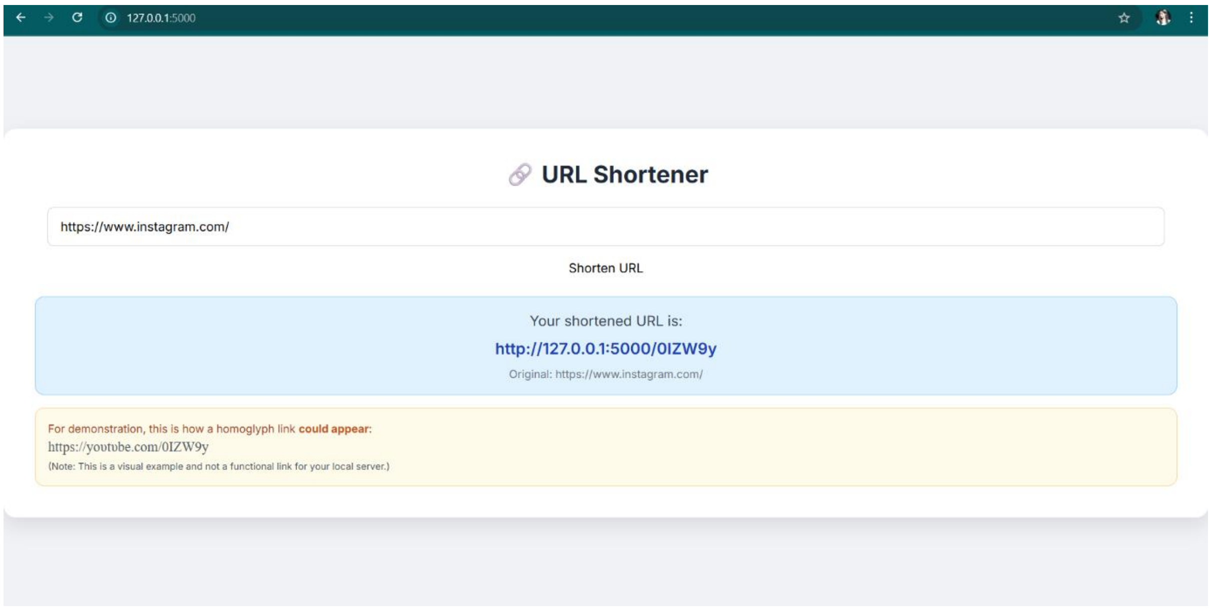
```
 * Serving Flask app 'try'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serv
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 309-635-333
127.0.0.1 - - [08/Aug/2025 19:50:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [08/Aug/2025 19:50:26] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [08/Aug/2025 19:52:34] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [08/Aug/2025 19:53:47] "GET /0IZW9y HTTP/1.1" 302 -
```



Fill in the details of which link you want to shorten.



And we get the shortened url easily,


127.0.0.1:5000

URL Shortener

https://www.instagram.com/

Shorten URL

Your shortened URL is:

http://127.0.0.1:5000/0IZW9y

Original: https://www.instagram.com/

For demonstration, this is how a homoglyph link **could appear**:

https://youtube.com/0IZW9y

(Note: This is a visual example and not a functional link for your local server.)

A fair warning, because this link is hosted on local servers, we cant share that, buying domain and actually giving it some meaning would be awesome, a part for later endeavours

Why This Tool Is Useful

This tool serves as a **warning** and an **educational demonstration** of a security threat rather than a practical service.

- **Cybersecurity Awareness**: It highlights how easily malicious actors can create deceptive URLs that look legitimate but lead to harmful sites.
- **Demonstration**: The tool provides a tangible example of a **homoglyph attack** or **IDN homograph attack**, where characters in a domain name are replaced with homoglyphs.
- **Testing**: Developers and security researchers can use this POC to understand the mechanics of URL shortening and the creation of deceptive links.

Use case Examples

The main use case is to demonstrate the security risks of homoglyph links.

- **Phishing Simulation**: A security team could use a similar tool to create mock phishing emails with homoglyph links to train employees on how to spot deceptive URLs. For instance, a link that appears to be apple.com could be a malicious site using the Cyrillic 'a' (a).
- **Educational Content**: Cybersecurity educators can use this POC to explain homoglyph attacks to students, showing them how a link like paypal.com (using Cyrillic characters) looks identical to the real paypal.com.

- **Website Hardening**: A web service provider could use this concept to proactively identify and block lookalike domains that use homoglyphs of their own brand name.

Who Should Use It

This POC is primarily for:

- **Cybersecurity Professionals**: For educational purposes and as a part of security training programs.
- **Developers**: To understand the code and build similar tools with enhanced security features.
- **Educators and Students**: As a practical learning tool to visualize and understand phishing and homoglyph attacks.

Future Enhancements

- **Full Production Deployment**:
  - Deploy the application on a dedicated server with a custom domain to make the shortened links globally accessible.
  - Use a more robust database like PostgreSQL or MySQL for better scalability and performance.
- **Enhanced Security Features**:
  - Implement **link expiration** to automatically remove old short URLs.
  - Add **password protection** for sensitive long URLs.
  - Integrate a **web analytics dashboard** to track clicks and user geography.
  - Create a **blocklist of common homoglyph domains** to prevent malicious links from being shortened.
- **User Interface Improvements**:
  - Develop a user dashboard to manage created URLs.
  - Improve the homoglyph display to be more interactive, perhaps with a toggle to switch between the real and homoglyph characters.

AI Integration Possibility

AI could be used to enhance the security and user experience of a homoglyph-aware URL shortener.

- **Malicious URL Detection**: An AI model could be trained to identify and flag potential phishing or malware URLs before they are shortened. It would analyse the target URL's content, domain age, and reputation.

- **Advanced Homoglyph Generation**: Instead of a simple map, an AI could dynamically generate more convincing homoglyph variants for a given URL by analysing Unicode character sets and visual similarity.
- **Predictive Security Analysis**: The AI could predict which newly created short URLs are most likely to be used for malicious purposes by analysing patterns in the slugs and the original long URLs.