Assignment 1 - Group 28

UvA - Web Services and Cloud-Based Systems

Tianhao Xu
14129027

Summer Xia
14094584

Yiming Xu
13284657

1 DESIGN

In this assignment, we create a website using Flask and SQLite to deploy a RESTful service. It has four features: URL Shortener, Short URLs Statistics, Data Clear, and Authentication.
URL Shortener: We first check the correctness of a new URL by regular expression. If it is a valid URL, we generate a unique short ID for it by combining three random numbers and letters. By clicking the short URL, users will be redirected to the original website.
Short URLs Statistics: On this page, users can change the short ID or delete any existing short URL stored in the database. We also keep track of each short URL and count how many times it is visited.
Data Clear: Delete all of the short URLs in the database.
Authentication: This is a user login process, which ensures that only registered users can use the service. Username and password are needed to be input when new users log in to the website. The default account is mentioned in *readme.md*

1.1 Design UML

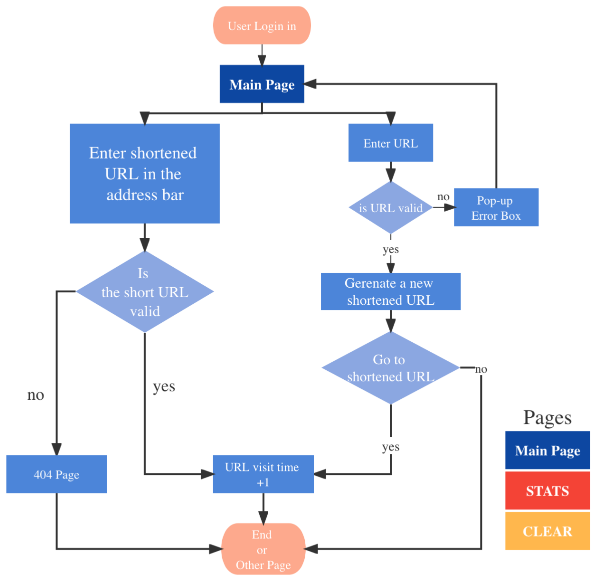


Figure 1: UML of URL shortener

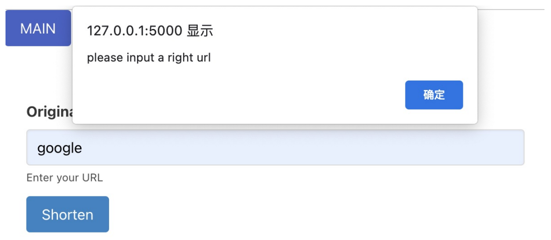
2 IMPLEMENTATION

2.1 Data storage

We use a SQLite database to store our data, creating a database with five columns, id., orginal_URL, short_URL, visits, data_created. We set id to be a primary key with an integer, which is unique for each item, and 'orginal_URL' and 'short_URL' represent long URL and short URL respectively. The 'visits' attributes count the number of times the URL is accessed, and 'data_created' is an auto-generated column with the time of creation.

2.2 URL Shortener

URL check It starts with determining whether the URL entered is correct or not with regular expressions in 'CheckURL'. This is implemented on both the HTML page and the route procedure, since we have to block the query from uploading to the database. If the entered URL is unreachable, users will see a pop-up, reminding to enter the correct URL.


127.0.0.1:5000 显示

please input a right url

Figure 2: URL correctness check

URL query If receiving a valid URL, it creates a unique short URL and record the query as a row to the database containing the 5 attributes (2.1).

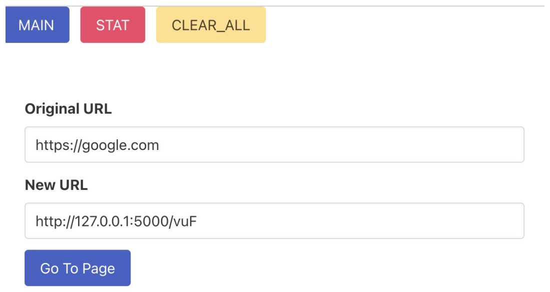
After submitting the query, it returns the short URL and the user can directly go to the original website by using this short URL. Users are also able to enter a short URL at the address bar, the service will check the availability and redirect to the original website if exists. IF an arbitrary short URL (invalid short URL) is entered, the service will return a 404 page. The '404 NOT FOUND' is a page written by us, not the real return HTTP code, the latter can be found in by the network inspection tool or flask terminal.

Short URL generation In this case, we define the length of a short ID by 3. For each URL, we use a random combination of uppercase letters [A-Z], lowercase letters [a-z], and numbers [0-9], which means the maximum storage is $62^3 \approx 216{,}000$ websites. The number of lengths and rules can be changed in models.py

2.3 Short URLs Statistics

On this page, we display all of the original URLs and short URLs in the database, as well as the ability to count visits, delete and change short URLs.

When a user enters a new custom short ID and clicks the 'Update' button, the database's corresponding old short ID will be overwritten, and the new short URL is displayed on the 'stats' page.



Original URL

https://google.com

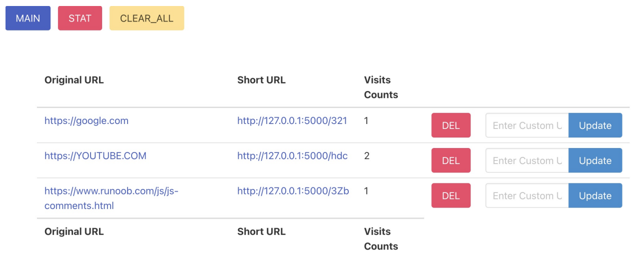
New URL

http://127.0.0.1:5000/vuF

Figure 3: Short URL generate

The 'DEL' button deletes the pair of original URLs and short IDs from the database, and the related information will be also deleted on the 'stats' page.

We also keep track of how many times a person visits each web page, which is reflected in the 'Visits Counts column'.



Original URL	Short URL	Visits Counts
https://google.com	http://127.0.0.1:5000/321	1
https://YOUTUBE.COM	http://127.0.0.1:5000/hdc	2
https://www.runoob.com/js/js-comments.html	http://127.0.0.1:5000/3Zb	1
Original URL	Short URL	Visits Counts

Figure 4: Short URLs Statistics

2.4 Data Clear

The 'CLEAR_ALL button' allows users to delete all data from the database. If these records are deleted successfully, the operation page will display '200 DONE'. As a note, the '200 DONE' is a page written by us, not the real return HTTP code, the latter can be found in by the network inspection tool flask terminal.

2.5 Authentication

We want the route pages to be protected by some kind of login, because users using the service don't necessarily need to modify the links they just need to be able to use the links. The basic HTTP authentication codes are quoted from public flask snippets. The purpose is to ensure the security of the service provider and the user system is better for further permissions management.

When a new user logs into the website for the first time, there will be a pop-up box to enter a username and password. We set the administrator account in the .env file.

Enter the following account password when asked to log in: ***Username*** *admin* ***Password*** *password*

3 ANSWER TO QUESTION 3

The service utilizes an SQLite database to represent and record all the shortened URLs. When a user enters a shortened URL, the service will check the availability of URL in the database. Therefore, for each generated shortened URL it can be used by different users, leading to a multi-user service. Moreover, to make each user with unique URL storage, we can create multiple databases/tables to separate each person combining with the authentication system.

Considering the performance of SQLite, it may have trouble dealing with high concurrency situations resulting in a relative processing speed. We can have faster lookups by keeping the URLs in memory like using Redis, however, it requires a (remote) endpoint storage to prevent data loss.

4 GROUP MEMBER CONTRIBUTION

Task	Contribution Person
Research & Literature	Yiming, Tianhao
flask code	Summer, Tianhao
web page	Yiming, Summer
Report	Summer, Tianhao, Yiming