INNOMATİCS
RESEARCH LABS

PROJECT REPORT

OF BUILDING A URL-SHORTNER WEB APPLICATION

BY USING PYTHON AND FLASK FRAMEWORK

By:-

Atishay Jain

This project is a part of Data science internship at Innomatics Research Labs

Contents

1. Problem statement and the system design

- Given a long URL, the service should generate a shorter and unique alias of it.
- When the user hits a short link, the service should redirect to the original link.
- Links should be stored in a database history
- The system should be highly available. This is really important to consider because if the service goes down, all the URL redirection will start failing.
- URL redirection should happen in real-time with minimal latency.
- Shortened links should not be predictable.

2. Introduction of the Project.

This report introduces the process of creating a URL-Shortener web application. URL shortening is a technique on the World Wide Web in which a Uniform Resource Locator may be made substantially shorter and still direct to the required page. This website has three major components: homepage, history page and error handling page. The implementation uses a tool called Flask Framework which is an excellent open-source web application frame work for complex data-driven website development. The major part of this report will introduce how to use Flask to create a database table, web page user interface and inner logic to handle user request by going through the implementation process.

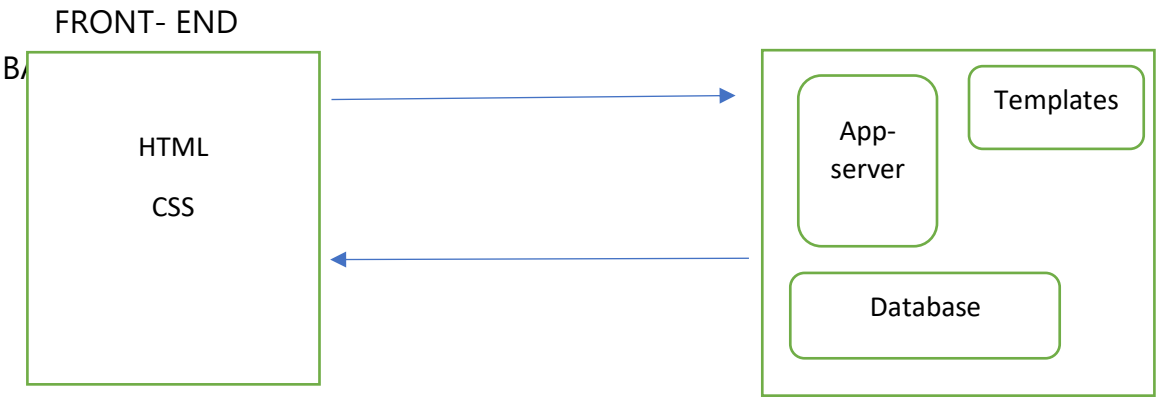
2. System Requirements of the Project.

We know that any website has 2 parts:-

- Frontend
- Backend

Front-end web development is the development of the graphical user interface of a website, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that website.

Backend Development is also known as server-side development. It is everything that the users don't see and contains behind-the-scenes activities that occur when performing any action on a website. It focuses primarily on databases, backend logic, APIs, and Servers.



The knowledge of following tools is required to proceed with the project:-

- HTML
- CSS
- JavaScript

- Python
- Flask
- SQL

2. Python Coding.

Step 1) Importing the required libraries and creating a flask app

We need to import the following libraries in order to proceed with the coding part:-

- Flask
- render_template
- request
- redirect
- url_for
- SQLAlchemy
- Migrate
- random
- String
- Os

Now, we need to create a mini flask app by defining the app name and writing the app.run command.

Step 2) SQLAlchemy configuration

Set up the base directory and configure the database URI by defining the path for database. Secondly, we need to pass our application to SQLAlchemy. Further, we need to create a table in the database and define all the columns and required functions/constructors. Then we need to initialize the flask db and migrate on the same path.

Step3) Defining the routes

Our application will contain the following routes:-

"/"

"/<short_url>"

"/display/<url>"

"/history"

We have successfully set up the backend part that is, application server and the database. Now, it is required to work on templates and the frontend (HTML,CSS) part.

Step 4) Adding templates

Our application will contain the following templates: -

- base.html
- index.html
- shorturl.html
- home.html
- notfound.html
- history.html

The static folder will contain all the required background images and gifs for CSS.

➢ **How does the shorten_url( ) function work?**

The function contains variable letters in which all possible letters and numbers are present

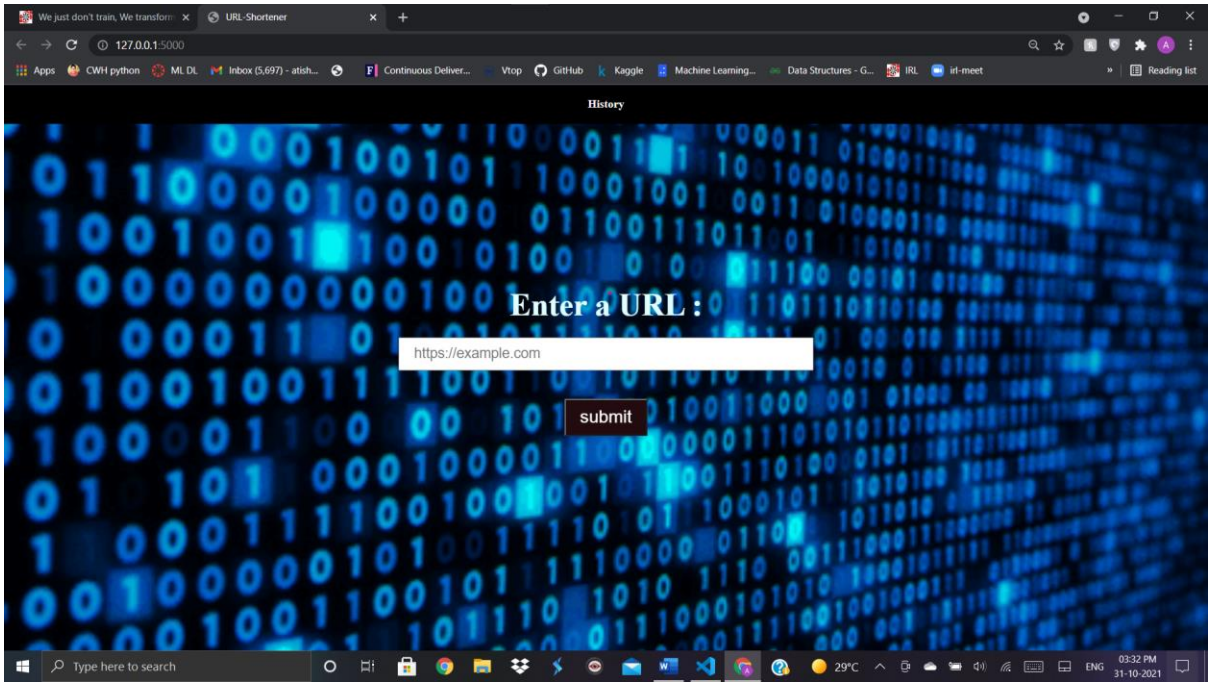
```
letters = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

The function check if the URL is already present in the database. If the URL is not present, then the function maps the original URL with ip/random 5 letters.

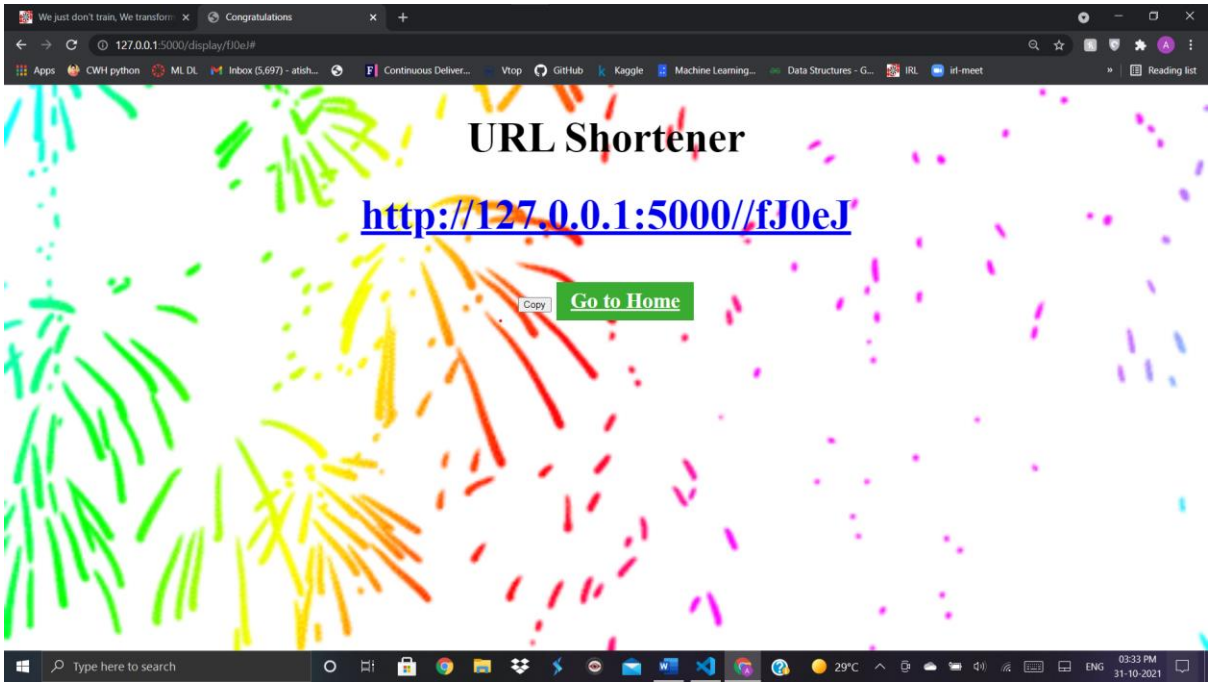
```
def shorten_url():
    letters  =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

    print(letters)
    # check if url already exits
    while True:
        rand_letters = random.choices(letters, k=5)
        rand_letters = "".join(rand_letters)
        short_url = Urls.query.filter_by(short=rand_letters).first()
        if not short_url:
            return rand_letters
```

4. Output of the project

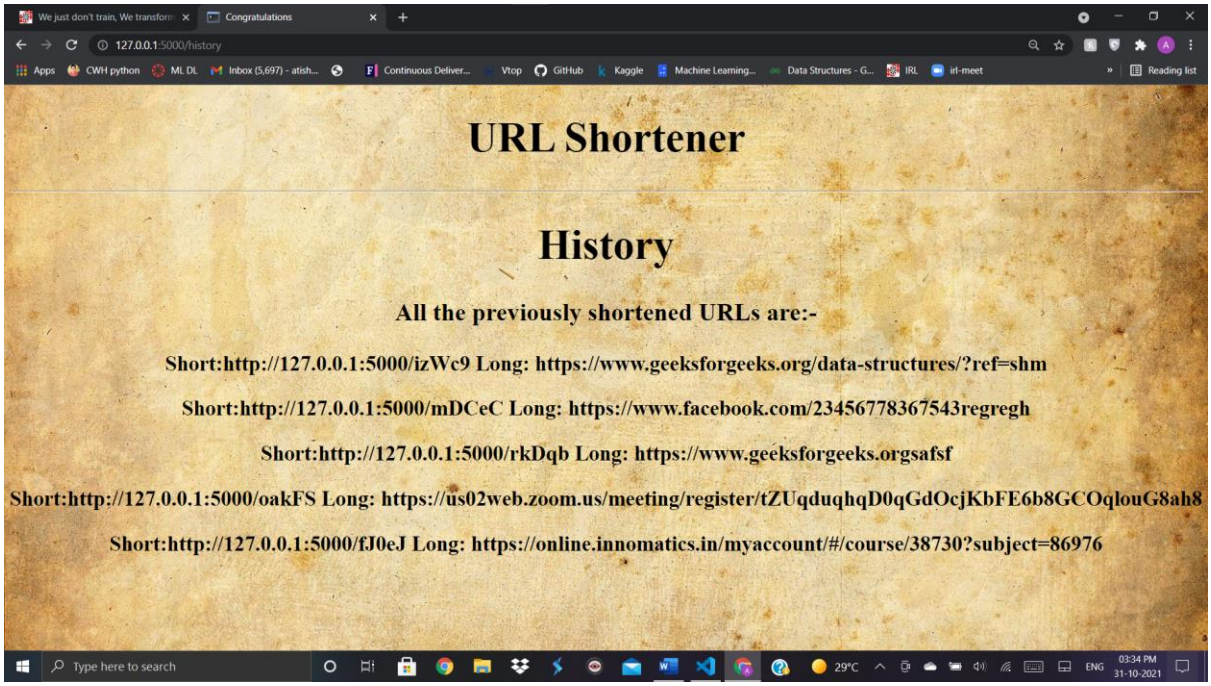
➢ Home screen



➢ Shortened- URL page



➢ History page



- **Checking if the URL is entered or not**

5. Conclusion: -

This report covers the approach towards creating a URL-Shortener web application using python and Flask framework. We have divided the project into the following sub divisions:-

Problem statement
System Requirements of the Project.
Python Coding.
Output of the Project.

We have briefly covered the sub divisions and have observed the processing of the project.