**REPORT**

**TITLE:** URL Shortener.

**ABSTRACT**: Sometimes we need to share or send links and this can be tiresome and annoying to copy and paste long URLs. That is where URL shorteners come in. Not only it helps in shortening the URL but it also allows the user to copy the shortened URL with a click of a button.

**INTRODUCTION:** URL shorteners are apps, typically web apps, that take a long URL and shrink it down to fewer characters. An example is Tiny URL. If you copy and paste a long link into the main field on Tiny URL, the web page spits out a much shorter link that goes to the same page. Shortened links display better in emails, print material, and other places.

Some URL shortener services turn long links into small ones, and that's it. The more capable one’s double as link-tracking services. In other words, the app shows you the traffic coming from each link.

This capability is useful because you can then make specific links for specific purposes. Let's say you want people to go to the page MyCompany.com/business-partnerships/new-opportunities. That's a long URL, so you generate two short versions, A and B. You use link A exclusively in cold emails reaching out to new potential partners. Link B goes exclusively into emails with existing partners. That way, you can differentiate the email link traffic coming from new versus existing partners.

**APPROACH:**

* The project consists of 2 parts:

1. Frontend (done with HTML, CSS and Bootstrap)

2. Backend - Flask (Python)

3. Backend - Database ORM

* The front-end consists of 2 web pages:

1. Home Page - A page will be shown where the user can enter the URL he/she wants to shorten. After the ‘shorten’ button is clicked, the shortened URL is displayed in the text field which the user can copy using the copy button.

2. History Page - Containing all the Original URLs along with the Shortened URLs.

Before shortening any URL we have to verify if the URL entered by the user is valid or not it can be done by an HTML parameter called pattern=” ” which partially verifies if the URL is valid or not.

**Step 1 — Creating Database:**

Import required libraries for creating a database, here I am using sqlalchemy. Configure sqlalchemy and create a database model, which contains the columns of the database, we are creating the schema of the database.

The database contains the following columns:

* Id – it gives every URL an integer number which is also a primary key.
* Original\_url – contains user entered URL.
* Shorten\_url – contains 4 digits generated URL.

**Step 2 — Creating the Index Page for Shortening URLs:**

In this step, you will create a Flask route for the index page, which will allow users to enter a URL that you then save into the database. For every URL entered a random number containing four digits is assigned to the link, then render as a result.

The home\_post() function is a Flask view function, which is a function decorated using the special @app.route [decorator](https://en.wikipedia.org/wiki/Python_syntax_and_semantics#Decorators). Its return value gets converted into an HTTP response that an HTTP client, such as a web browser, displays.

Inside the home\_post() view function, you accept POST requests by passing methods=('POST') to the app.route() decorator. You open a database connection.

Then if the request is a POST request, it takes input from the HTML page textbox generates 4 digits random number and assigns it to the URL, and saves it in the database and finally, it renders a template called index.html, which will contain shortened URL.

if the user has submitted an empty form, you flash the message please fill out this field!  and it stays on the index page, this can be done by using an HTML parameter called requires minlength=”4”.

## Step 3 — Adding the Redirect Route:

## In this step, you will add a new route that takes a short URL that is assigned to its original URL. This can be done by querying the database by validating the short URL to its original URL and if they validate we take them to the original URL. Finally, you will redirect users to the original URL.

**Step 4 — Creating the History Page:**

History page contains all the URL's that has been submitted to the web app along with its 4digits number, it is done by querying the database and it renders the history.html, a user can also delete the history of the links using the delete button, we can do this by giving every URL a unique id, and passing this id to delete() this function queries the database if the URL is present in the database it is deleted.

**FURTHER DEVELOPMENT:** one can add sign-up, sign-in pages to create a separate account for different users so a user can log in to their account to get shortened URLs and they can also see their previous shorten URLs On the history page.