Shortlink Architecture Specifications

Introduction

Shortlink is a link shortener built over Python and Django. For this project, I have used full Django MVT including the templates. The main workflow of this is to generate a random alphabetical keyword that would be used as the link. Once that is generated, we check if it doesn't already exist. If it does not, then we add the keyword and the full link in the database. After that, the keyword is shown to the user on the frontend.

DB Model:

I am only using a single table that is storing the URL keyword, user link and date created. The intend here is not have it as simple as possible so I am not adding the user login mechanism. Here is how the table looks like:

Keyword Generation and Insertion

A random keyword is generated using ascii lowercase strings. The length of the keyword is 5 that means that the maximum limit of the links that this link shortener can store is 26^5 or 11,881,376.

Before inserting the keyword and its corresponding link, the program checks if it the keyword doesn't already exist. If it does, we keep on generating new keyword till the time it does not find a new unique keyword.

Link validation at Input

A link could be in many forms, it could be with http and www or with either or with none. TO validate this, I just used the Django's URLField and then checked for form errors when trying to get the data. However, now in the database, I would have different link values, some could have http, some could not. We will solve this in next step

Link Validation at Inferencing

To handle the link redirection part, I created a function that will format the link so that it is assured that the link we redirect to contains atleast http. If a link stored in the db is without http or https, we

assume that the link would be an http link so we redirect to the http version of it. If the link already has https, then we redirect to the https address.

URL Formatting:

```
def formaturl(url):
    if not re.match('(?:http|https)://', url):
        return 'http://{}'.format(url)
    return url
```

The scope of this could be much more than what I have implemented but it still brought its own challenges. You can access this online at https://shortlinker.herokuapp.com/.

For any other details you can contact me at sethi.nishant24@gmail.com.