MSc. Big Data Analytics & Management

Cloud Computing - [MSCBD-CC]

Assignment 03 - Documentation

MERT ALTINTAS - 2982775

---

1. Project Overview

In this assignment, the given task was to make a Twitter Implementation using the Google Cloud Platform and its components. The project's back-end was realized by Python programming language and Google Cloud's NoSQL database (ndb) was used in order to store the user details, tweets and images. On the other hand, the front-end tasks were performed by using HTML, CSS, Bootstrap, and JavaScript. The version control was made via GitHub.

In this paper, firstly, the classes and their methods will be explained. After that section, the detailed information about the data structure and the models are going to be presented. Finally, the methodology behind the UI will be described.

2. Classes & Methods

The functionality of the application is basically realized with 7 Python classes and their methods. The classes used in the project are as follows:

- **MainPage Class**

  This class is mainly responsible for rendering the home and landing pages, user authentication, and letting the user search for other users plus the tweets. It has two methods such as get and post.

  The get method checks if the user logged in or not. If the user didn't log in, the method redirects the user to the landing page where the login process can be made. Otherwise, it controls if it is the user's first login. There are two scenarios according to that. If it is the user's first login, the method redirects the user to the first login page where the user can set a username. If the user has already registered to the system, the method shows the home page where the live feed can be visualized, the search operations can be made and a tweet can be created.

  The post method is responsible for searching other users and the contents in the tweets. When the search button is pressed, the method takes the input and splits the words. Firstly, it checks if there are any related users matching to the input, secondly it checks the tweets' contents. Finally, it gives the results in two different arrays to present them to the user.

- **FirstLogin Class**

This class is responsible for dealing with the user's first login process. It has 2 methods.

The get method checks the user authentication and after that it redirects the user to the First Login page where the user can set a unique username.

The post method controls the Next button. When it is pressed, it gets the username and checks if that username already exists in the system. If the user entered a unique username, it registers the user to the system and redirects to edit page where user can set the other details such as the name and the profile description.

- **Edit Class**

Edit class is responsible for rendering the edit page and getting the inputs from the user. It has 2 methods like the other classes.

The get method controls the user authentication and according to the result it redirects the user to the landing, the first login or to the edit page. If the user was not registered before it presents the first login page and on the other hand, if the user has already registered it shows the edit page where user can edit its name and profile description. The username is immutable.

The post method controls the button and when it is pressed, it gets the inputs and makes the relevant changes in the datastore. Finally, it redirects the user to the profile page by passing the username via the URL.

- **Profile Class**

Profile class is responsible for showing users' profile pages. It has 2 methods.

After controlling the user authentication, the get method gets the username from the page URL and checks if the visited profile page belongs to the user. If it is not the user's profile page it need to show "Follow" or "Unfollow" button. However, if it is the user's own profile page it needs to show "Edit My Profile" button. Thus, it checks if the username fetched from the URL matches the user's username. After that control, it gets the last 50 tweets and renders the profile page.

The post method controls the buttons on that page. If the user clicks the follow button, the method adds the followed user's id to the both user's relevant lists. In the same manner, if the unfollow button is pressed, it deletes the unfollowed user's id from the both user's relevant lists. Delete button deletes the tweet from the user's tweet list and also from the datastore. At the end of each action the method redirects the user to the relevant profile page.

- **tweetEdit Class**

tweetEdit class is responsible for editing an existing Tweet. It has two methods such as get and post.

The get method controls the user authentication as in the other classes and gets the tweet that the user wants to edit. It renders the tweetEdit page by giving the relevant features to the template.

The post method controls the Submit button and gets the changes from user when it is pressed. It applies the changes in the datastore and redirects the user to its own user profile.

- **UploadHandler and DownloadHandler Classes**

These two classes are responsible for uploading the image to the blobstore and retrieving it.

UploadHandler class controls the form in the main page where the user can create a tweet. After pressing the Tweet button, the post method gets the current user, the tweet text and the image if there is any. If the user uploaded an image, the method puts the image to the blobstore, gets the blob key and creates the tweet by setting the blob key as one of the tweet's attribute. If there is no image, the method just creates the tweet with the entered text. Finally, it redirects the user to the main page.

DownloadHandler is responsible from showing the image to the user. It gets an URL link from the relevant page and fetches the image key from this URL. If the key is valid, the method sends the blob to the page.

3. Models

Four models are used during the realization of the project.

MyUser stands for the user model. Each user has a name, username, profile description, tweetsList, followersList, followedList and tweet_count. The key is specified as user id. Name, username and profile description are the basic user informations. tweetsList attribute holds the tweet ids, followersList holds the followers' user_ids, fallowedList consists of the user_ids which the user follows and tweet_count indicates the total number of tweets that user posted.

Tweets model consists of 5 attributes and 4 methods. The key for this model is set in a format such as "user_id:tweet_count". tweetText shows the raw text of the tweet. Username is the username of the user who posted that tweet. tweetWords is a list that contains each word in the tweetText. This attribute makes the content search in tweets easier. timestamp shows the date and time when the tweet is created and blobKey is the shortcut to the blob key to upload the blob image.

strip_punctuation method gets the tweetText and removes the punctuations and splits it into its words. getTweetsByKey method gets the tweets via direct key access. It fetches the tweet_ids in tweetsList attribute under MyUser model and returns the top 50 most recent tweets. On the other hand, getTweets is used for getting the tweets for the live feed on the home page. It runs a query over Tweets and gets the top 50 most recent tweets from the followed users. Finally, getUsername method gets the usernames from the fallowedList attribute under MyUser class and returns them in a list.

AllUsers class contains all the users in the system. The key is specified as the username and the values are user_ids. Thus, it provides direct key access with reaching the user_id by specifying the username ie. if the username is known, the related user_id can be gathered using this model.

BlobCollection is the model for storing images under blob keys. Filename shows the each files' name and blob stores the blob keys.

4. UI Design

The UI is designed to present a user-friendly environment to the user in order to facilitate the usage of the application. During the development of the interface, CSS, Bootstrap, and JavaScript are used. Thanks to the Bootstrap, pages become responsive which means that they change their form automatically according to the window size.

In the UI, the navigation bar is placed on the left side. Two tabs such as "Home" and "Your Profile" are linked to the navigation bar. In the home page, the user information and the statistics are shown on the navigation bar. However, in the profile page, they are removed because of the reason that these statistics are shown in the body part of that page. The search box can be found in the header of the main page and the logout button is fixed on the left hand side of the header in each page. The footer consists of the credentials. And the body part is placed between these three parts ie. the navigation bar, the header and the footer.

The main color scheme for the application is chosen as blue combined with light gray. According to the design perspective, blue is a calming natural color. Plus, shades of blue are perceived to be friendly, authoritative, peaceful, and trustworthy.

The tables, forms, and buttons are designed proportionally so that it presents a proper view to the user. No unnecessary information is used. On the other hand, complexity is avoided while designing the pages. In addition, all the platforms such as Internet Explorer, Google Chrome, Safari, etc. are considered while placing each element on the pages.