Documentation of Assignment 3

MastodonGram

Author: Ali Ramezanian Nik

main.py:

- **Libraries**
  - google.oauth2.id_token
    - Parse and verify an ID Token issued by Google OAuth 2.0 authorization server
  - google.cloud
    - Access datastore module from App engine and Also Buckets.
  - flask
    - Access Flask Framework which is a simple handler of HTTP methods
  - google.auth.transport
    - This library simplifies using Google various server-to-server authentication mechanisms to access Google APIs

- **Models and data structures**
  - User:
    - Description: User model is used to save information of a profile in Our app “MastodonGram” . The saved information and fields are explained below:
      - key: This is the key to access this model. Key is only contained the email as it is different for every user.
      - profile_name: the profile_name of user. The default is his/her email name without its domain.
      - followers: This field conclude users and the time they followed the user . The user save as map of {“Username: “date Followed” }.

        - following: This structure of this field is as same as followers field but this field save information about the users that this user follow them.
        - profile_photo: The address of where profile photo is saved in Bucket.
    - Reason: With only 5 field , we have saved everything belonging to user . Also we ca retrieve followers and followings of user in chronological order as we save timestamp in the value of dict item.
- Post:
    - Description: Post model is used to save post of every user with its caption.
        - key: the author of post and created date of that post as timestamp will combined to generate a unique key for accessing post. We used this key because every unique user can not create two post simultaneously so adding timestamp to the author will make it a unique key.
        - author: username of author of the post.
        - post_photo: the address of post photo. The content of photo is saved in datastorage bucket.
        - created_date: the date that the post is created saved as timestamp.
        - caption: the caption of post shorter than 200 characaters.
    - Reason: we did not add comments as another field for post model as it make a big field with a lot of comment indexes.
- comment:
    - Description: This model stores the comments which users write under posts.
        - key: we combined key of post and the author of comment with created_date of that comment and it gives us a unique key.
        - author: as its name suggest it is the author name of the comment.
        - created_date: the time which user wrotes the comment .
        - text: the text of comment . The same policies of caption also applied for comment text.
        - post_key: the key of post that is used to retrieve comments for every post.
    - Reason: we defined post and comment models separated as it make it easy to add comments without querying and loading post model into memory.

• Functions:

- flash_redirect(message, path):retrieve_row(kind, name)

        - type: String, String → Option(None, google.cloud.datastore.entity.Entity)
    - description: fetch the information of the specified row and return a copy of that data. As the fetch information has type of google.cloud.datastore.entity.Entity but we prefer to return Dic.
        - description: This function not just simply forward to another page and instead it also carry a message to page redirected.
    - replace_address_with_photo(data):
        - description: This function iterate over a list of addresses of photos and load the binary data of those addresses from bucket into memory.
    - retrieve_row(kind, name)
        - description: fetch the information of the specified row and return a copy of that data. As the fetch information has type of google.cloud.datastore.entity.Entity but we prefer to return Dic. This also cast address of retrieved photos to their binary contents.
    - create_row((kind, name, data):
        - Description: this function gets an object called “data” and its type is an Entity object (dict) and create row with that data.
    - upload_blob(address, content):
        - Description: Upload binary content to the address specified in datastorage bucket. It also encode the data to base64 .
    - download_blob(address):
        - Description: Download the photo content from specified address and decode it as utf-8 and store it in memory.
    - add_user_if_not_added(key):
        - Description: initiate the user in the first login. In the initiation process, A default avatar is set (may the 4th be with you) .
    - get_session_info()
        - description: validate the user based on the cookie token. If the user is validated it first try to initiate the user if it is the first time that used joined, or only return the user information.
    - get_one_post(userkey, created_date)
        - Description: retrieve one post from the database with all of its comments. The userkey and created_date (combination of these two fields will form the post key) are also provided for this function.
    - get_user_posts(users, timeline=False)

        - Description: Retrieve all posts and their comments that is belong to user list(users) provided to this function , there is flag that can be passed to this function to limit retrieved items and its name is “timeline”. When showing posts in timeline we should only show the last 50 posts.
    - action(actiontype, userkey):
        - Description: this function follow or unfollow a user based on the argument passed to it. The actiontype argument define the action and userkey defines who the authorized user should follow or unfollow.
    - users_list(searchtype, userkey):
        - Description: this function should return a list of users. The context is vary : 1-when we search with the search bar, 2- when clicking on the follower list 3- when clicking on the following list. The context is defined by searchtype argument. The userkey argument defines which user we want to see the list of its following or followers.
    - userpage(username):
        - Description: userpage retrieve the profile page of user passed as argument. Also it validate the authenticity of user. If the user want to see its own page , some modification applies to not see duplicate items in UI.
    - addcomment(user, post_key, created_time):
        - Description: this function as it name suggests , add a comment to a post . The post key is in the argument passed to this function and the data of comment is retrieved by the post form data.
    - singlepost(author, created_date):
        - Description: this function is as like userpage function but the different is that it only fetches 1 post.
    - addpost():
        - Description: this function first validate the content of add post form (check the format of photo , the size of caption and …) and then if everything is fine, upload the binary content to datastorage and save the address of uploaded photo into memory and then create an index in the datastore .
    - error():
        - Description: render template page of error which shows errors in a fancy way.
    - Root():
        - Description: This function tries to validate the user . If it does not, it render a signup page. If it does, try to show the timeline of user authenticated.