Documentation

Documentation of functions

def Query(in_kind):
The function is written to decrease number of lines, used to do the queries to google datastore

def next_value(my_value):
The function returns the next string value in lexicographic order. This function is then used to perform a search on the social network

def NameQuery(in_kind, attr_name, attr_value):
The function is written to speed up the call of queries with Name attribute to google datastore. This function is used later to check if the user with this user name exists

def NameQuery2(in_kind, attr_name, attr_value):
The function does the query for the beginning of the name. It is used to search for the users with such profile name.

def create_set(var):
The function iterates through all the data found in the storage stored in the var variable and returns them in the specified format: python set

def get_next_id(id):
The function returns the next ascending id

def find_max(in_set):
The function converts the ids stored in string form into numbers and returns the maximum number of them

def get_id():
The function creates a new id for the ‘Post’ and ‘Comment’ entity

def count(text):
The function checks if the comment length is less than 200 characters. It is used for ‘Comment’ entity

def get_time():
This function gets the current time in a string format. It used to make posts in chronological order.

def toTime(in_string):

This function converts the time from string format back to datetime format to do comparison

def retrieveUserId(claims):
This function returns id of the current ‘User’. This can be important for the UI.

def retrieveEntityById(kind, id):
This function returns the entity with this id from the datastore. Some of the entities store a list of ids of other entities. Therefore, it is convenient to have this function to get the entity itself/

def retrieveUser(claims):
This function returns the user with these claims. It returns the entity if it exists. It is used to check if the user logs in for the first time

def createUser(claims, user_name, profile_name):
This function creates ‘User’ entity with this user_name and profile_name. It also checks user_name to be unique.

def Follow(user_id, cur_id):
This function is used to make cur_id follower of user_id

def Unfollow(user_id, cur_id):
This function is used to make cur_id unfollow user_id

def CreateComment(post_id, user_id, comment_text):
This function creates a ‘Comment’ entity and adds it to the ‘comments’ attribute of the ‘Post’ entity

def addFile(file):
This function adds a picture to the google storage, using buckets.

def createPost(user_id, public_url, caption):
This function creates Post for the user with this user_id with the picture, located at public_url with this caption

def the_sorting(inds, A):
This function has inds and A input. inds are indices of sorted times of posts. A is an array with some attributes of these posts. The function substitutes the indices to get the attributes in the correct order.

@app.route('/followers/<string:user_id>')
def followers(user_id):
This function is used to prepare all the information needed to be shown in followers.html template - the page with the number of followers of the user.

@app.route('/create_post/<string:user_id>/<path:public_url>', methods=['POST'])

def CreatePost(user_id, public_url):

This function is used to create a post for this user with an image at this public url.

@app.route('/upload_file/<string:user_id>', methods=['post'])
def UploadFile(user_id):

This function is used to upload a file for the user with this user_id. It checks for the file format.

@app.route('/followings/<string:user_id>')
def followings(user_id):

This function is used to prepare all the information needed to be shown in followings.html template - the page with the number of followings of the user.

@app.route('/search', methods=['GET', 'POST'])
def search():

This function is used to search different users. It calls **NameQuery2** function, that is used to search for the users with profile name with the beginning, got by the form from the search.html template

@app.route('/home_page/<string:user_id>')
def HomePage(user_id):

This function is used to prepare all information needed to render the home page of the user with this user id. It collects all the needed information for the 50 post in reverse chronological order

@app.route('/add_comment/<string:post_id>/<string:user_id>/<string:page>', methods=['POST'])
def AddComment(post_id, user_id, page):

This function is used to add comments for this user with this post and be redirected to the page, where the comment was posted.

@app.route('/see_post/<string:post_id>')
def Post(post_id):

This function is used to prepare all the information to see the post with this post id.

@app.route('/profile_page/<string:user_id>')
def profilePage(user_id):

This function is used to prepare all the needed information to see the profile page of the user with this user id.

@app.route('/create_profile', methods=['GET', 'POST'])
def createProfile():
This function is used to create a User entity for the current user, render template, that creates the profile page.

@app.route('/follow/<string:user_id>')
def follow(user_id):
The function is used to prepare information to run the **Follow** function and redirect to the profile page with the user.

@app.route('/unfollow/<string:user_id>')
def unfollow(user_id):
The function is used to prepare information to run the **Unfollow** function and redirect to the profile page with the user.

@app.route('/')
def root():
This function is root handler. It redirects to create_profile page, when the User logs in for the first time and to profile page otherwise.

Documentation of data

User:

‘User’ represents users of the social network. Users can follow/unfollow each other, add posts and write comments.

'id' - unique identifier, created with using claims - used to get entity by id and connected with claims - email from authentication
'user_name' - user name of the User - unique name of the user
'profile_name' - profile name of the User - name of the user, that can be not unique
'followers' - list of user ids of the followers of the User - it is convenient to get the ids of followers of the user of this page
'followings' - list of user ids of the following of the User - it is convenient to get the ids of followings of the user of this page
'posts' - list of post ids of the User - it is convenient to get the ids of posts of the user of this page and use them to get the posts attributes

Post:

‘Post’ represents a post entity - a picture, a caption under the picture, and the user owner. Users can upload a picture and add a caption. Comments can also be added under any ‘Post’ entity.

'id' - unique identifier of the post - used to get entity by id
'picture' - public url of the picture, that is used in this post - used to show the picture of the post
'caption' - caption under the post - used to store the text of the post
'user_id' - id of the user of the post - used to get the id of the user, then to get the entity of this users and the attributes
'time' : - current time of the post - used to store the actual time the post was created
'comments' - list of comment ids that were left under the post - used to store all the comment ids to get them easily

Comment:

‘Comment’ entity represents the text of the comment and the user who wrote it.

'id' - unique identifier of the comment - used to get entity by id
'text' - text of the comment - used to store the text to show it in the post
'user_id' - id of the user that left a comment - used to store the id of the user, that created