**Image Storage Service**

# **Functions and their Description**

The name and the detail of all the functions written in app.py file are described below:

* 1. **Login Function**

The login function is defined as def login ( ): This function render the sign in html template to the browser. This is the sign in page where if the user have account and can enter his email and password access his galleries. We used the flask session for the user session handling and firebase authentication for the user registration. For this purpose we have turned on the firebase authentication with email and password. In this page if user enters the correct email and password then he is redirected to the galleries page otherwise an error is shown on the top of the page.

* 1. **Signup Function**

The signup function is defined as def signup ( ). This function render the signup page by taking the signup.html template. On this page user have to enter the user name, name, email and password to create his account. If the username and the email is not already registered then it will create the account otherwise it will show the error. The route for this page is ("/signup") .

* 1. **Logout Function**

The logout function is defined as def logout (). This function logouts the user by removing the credentials of the user from the flask session after this, it is redirected to the sign in page.

* 1. **Galleries**

This function is defined as def galleries (). In this function galleries.html template is rendered first .After logging in to the system the galleries page is shown to the user .In this page user can see his all the galleries which he has created or share by the other users. on this page he can add new gallery , delete the gallery which he has created and edit the name of gallery. Button of the duplicate images is given to check the duplicate images in all the galleries. User can view all the images by clicking the name of the gallery or by pressing the eye button on every gallery grid.

* 1. **Sign in Function**

In this function the data taken from the sign in form is get. This data is passed to the firebase authentication to check if the user is registered or not. if the user email and password is correct, then user is allowed to enter the galleries page otherwise it will show the error on the sign in page.

* 1. **User Register**

This function is to register an new user to the system. The user information is taken from the user registration form which is given on the sign up page. User enters his name, email, password and username to create is his account in the system. This data is first checked if the same username exists or the user with the same email is already registered. If same email or username already registered then system will not allow the user to create the account. It will show the error on the signup page. if the user enter the unique information then the system will allow the user to create the account and his information is passed to the firebase authentication and the google date store model to store the user information .

* 1. **Add a new gallery**

After creating the account in the system user enters the galleries page. For the new user this page is empty and have no gallery. User can create is new gallery by pressing the “add gallery” button given on the top navigation bar of the galleries page. When the user clicks the add gallery button a bootstraps modal is popped up with an input form. User enters the name of the gallery and press the add button. The system first check if the gallery with this name already exists or not. If there is gallery with the same name is already in the gallery list of that user it will so the message that the gallery with the same name already exists. Otherwise it will create the new empty gallery with the given name. The name of this function is defined as add\_gallery (). And the URL of this function is “/add-gallery”.

* 1. **Edit the name of gallery**

In the gallery grid of the user there is a pencil button below the name of all the galleries of the user to edit the name of any gallery. When the user clicks the button a bootstraps modal is popped up with an input form already filled with the old gallery name. User enters the new name of the gallery and press the update button. The system first check if the gallery with this name already exists or not. If there is gallery with the same name is already in the gallery list of that user it will so the message that the gallery with the same name already exists. Otherwise it will change or update the name of the gallery. The name of this function is defined as edit\_gallery (). And the URL of this function is “/edit-gallery” with the id of that gallery.

* 1. **Delete a gallery**

In this function, the gallery and is all images are deleted when the user press the delete button in the bottom of gallery name. When user clicks the delete button, the id of that gallery is passed to the function through the URL, this function first check if the gallery contains any image or number of the images. If the gallery is not empty then the system will show the warning with the number of the image that the gallery contains. If the user confirm to delete the gallery. The first all of the images of that gallery are deleted and the object of that gallery is removed from the google data store and the images are remove from the google cloud storage bucket. The name of the function is defined as delete\_gallery (id ). And the URL for this is "/delete-gallery/<id>" with the id as argument,

* 1. **View images of the gallery**

In this function id of the gallery is passed to the function from the URL given in eye button on the every card of the gallery. When the user click on the name or the eye button, a new page open which contain the grid of all the images uploaded in this gallery. On this page there are three buttons given to upload the new image to the gallery, share the gallery with any other registered user and to check the duplicate images in the gallery. User can also delete any of the image within the gallery. All the images in the gallery are arranged in the grid system with image name and the delete button of the image. The button to return the galleries page is given on the top navigation bar of this page. User can also return the galleries page by home button on the navigation bar. The name of this function is defined as def gallery\_details (id). And the URL is "/gallery-details/<id>" with the gallery id as argument.

* 1. **Upload image to gallery**

This function performs the upload image function in the gallery. When the user clicks the upload image in the gallery detail page. The bootstraps modal pops up with the file input field. User select the image to be uploaded in the gallery. And press the upload button. This send the Post request to the add\_image function. For the image Google cloud storage is used every image will be saved on the cloud storage. This function create a unique name of random 20 character hexadecimal which is used to store image name. File extension is separated and attached to the generated hexadecimal code to create a new file name for the image. For example the previous name was “image.jpg” after renaming it will be like “33ec09-0r0f83-d15ce1.jpg” this new name will be saved in the Image model as name and the file with this name will be uploaded to the gallery\_images folder of the Google bucket. The name of this function is defined as add\_image ().

* 1. **Delete image**

This function is to delete the single image from the gallery. The id if the image is passed though the URL to the function. The function first get the object of the image from the google data store and the delete the image from the google cloud storage bucket. After deleting image from bucket it will remove the image from database. The gallery id is used to the redirection of the system that the system return to the same gallery after deleting the image from the gallery. The name of this function is defined as def delete\_image (id, gallery\_id ). The URL for this function is ("/delete\_image/<id>/<gallery\_id>".

* 1. **Check Duplicates in all Galleries**

This function searches all the duplicate images in all the galleries and render then in the grid form. The process of finding the duplicate images is done by using the MD5 hash. The hashed code of each image uploaded is save while uploading the image to the bucket. In this function for loop iterates from all the galleries and check all the images hashed code. After that if the hashed code of two or more images matches these images will be shown on duplicate images page. The name of the function is def check\_duplicates ( ). And the URL is "/check\_duplicates".

* 1. **Check Duplicates in a Gallery**

This function searches all the duplicate images in a single gallery and render then in the grid form. The process of finding the duplicate images is done by using the MD5 hash. The hashed code of each image uploaded is save while uploading the image to the bucket. In this function for loop iterates to check all the images hashed code. After that if the hashed code of two or more images matches these images will be shown on duplicate images page. . The name of the function is def check\_duplicates\_gallery ( ). And the URL is "/check\_duplicates\_gallery".

* 1. **Share gallery with other user**

This function is to share the gallery with the other users. When the user clicks the add user to gallery button given on the gallery detail page, a bootstraps modal is pooped up on the top of the page with the dropdown list of the registered users in the system. The user select the user to share the gallery with. This function add that selected user to list of users that can access that gallery. The name of this function is defined as add\_user\_to\_gallery (). The URL for this function is "/add-user-to-gallery".

* 1. **Upload image to the in google bucket**

This function is to upload the image blob in the google cloud storage bucket. File name, path of the image and the name of the google bucket is given to this function and this function upload that image to that bucket. The object of this function is mused whenever to add the image to the bucket. The name of this function is defined as def image\_upload\_to\_bucket (filename, file\_path, name\_of\_bucket)

* 1. **Hash image with MD5**

This function takes the image file and convert it into the hash code. The object of this function is used while uploading the image to the bucket. Hashed code are stored in google datastore to check the duplicate images in the project. The name of this function is defined as def hash\_image (file):

# **Models**

We have used three models in our program which are described below: we are using the NoSQL of the google data store that’s why all the models are in the form of JSON or Python dictionary.

* 1. **Users**

User model have following columns

1. Name
2. Username
3. Email
4. Gallery list
   1. **Gallery**

Gallery model have following columns

1. Gallery name
2. Username
3. User list
   1. **Image**

Image model have following columns

1. Image name
2. Gallery name
3. Hash code
4. Blob URL

# **Frameworks Used in System**

* 1. **Material Design Gallery Grid**

We have used the UI of MD bootstrap to design our gallery. **Material Design is a design language, originated from Google**, which is intensively used in interface projects for the Android system. You may ask why it is used. The answer is simple -– it is already a mature and organized tool which corresponds to the usability and legibility. Moreover, it is developing and changing dynamically. The result is a tool that may be a perfect match for screens, as well as the starting point for inexperienced people in UI/UX designing.

* 1. **Python Flask**

[Flask](http://flask.pocoo.org/) is a [web framework](https://www.fullstackpython.com/web-frameworks.html) built with a [small core and easy to extend philosophy](http://flask.pocoo.org/docs/design/) written in the Python Programming language. It is assumed more [Pythonic](http://blog.startifact.com/posts/older/what-is-pythonic.html) than the [Django](https://www.fullstackpython.com/django.html) web framework because of basic situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a fresher because there is little piece code for getting a simple app up and running.

* 1. **Google DataStore and cloud storage**

Google Datastore is a highly scalable NoSQL database for applications. Datastore automatically handles sharing and replication, providing you with a highly available and durable database that scales automatically to handle your applications' load. Datastore provides a many of functions such as ACID transactions, SQL-like queries, indexes, and much more. Google cloud storage also provide the Bundles of function to store data on it.

* 1. **Firebase Authentication**

Firebase authentication is used to know the user identification in the application. This is also frameworks of the google with following features

* Easy to use in any platform
* Flexible, drop-in UI
* Comprehensive security
* Fast implementation