PROG8150-22S-Sec2

Group 1

Final Capstone Project

Course Name/ [Course Code]: Mobile Application Development - IOS [PROG8470]

Submission date: 2022-08-14

Students Name:

1. Iulia Danilov (8816991)

2. Krupa Suhagiya (8813230)

3. Smit Mehta (8813480)

4. Jianxuan Li (8807952)

5. Feng Zhou (8808141)

6. Parshwa Shah ()

Table of Contents

1 Project overview:

The iOS operating system's native app is called Wish Tracker. It is in the category of programmes that take images in certain places.The application includes a login and registration feature that supports an account interface.

Everyone you know is using their smartphones to capture unforgettable, funny, beautiful, and goofy moments in their lives. While there are many ways to store these photos in the cloud, most of us have trouble remembering where pictures were.

EXIF format stores essential data such as the date and time of the photo, ISO, shutter speed, white balance, and camera model. These are crucial values for photographers. Most smartphones, cameras, and DSLRs have this feature.

Now, we are all sorted. We store location data for each photo we take, but another issue exists. How do you access this data? Here are six sites to help you find the EXIF data of any pic you want.

Note that this method will also work for old photos you took with a camera but can’t remember the location.

The application smart photo has a straightforward appearance and will assist you in finding the photo's location quickly. To upload an image, simply use the upload area. You will see the latitude and longitude information along with the specific address if the exit data contains the location information.

Here, smart photo goes a step further and also displays a number of user-uploaded images. You can immediately view city photos on the map. When you do, an interactive map and the location and address of the photo will be displayed.

The home screen is a dashboard with lots of content that connects to many features of the submission. Additionally, it serves as the application's default view upon access authentication.

Users of smart photo can also "Like" photos that other users have posted, which can aid viewers in locating the most appealing locations.

It is a social application that allows users to explore various locations.

2 TOOLS

- iOS 14
- XCode 12
- Python with Django
- PostgreSQL with PostGIS
- Kubernetes
- Docker
- GitHub
- GitHub actions

3 flow diagram

Below is the flow of the application represented in form of a flowchart as shown in the image below.

4 Details

We explain in two parts

Part 1: UI interface and related technologies

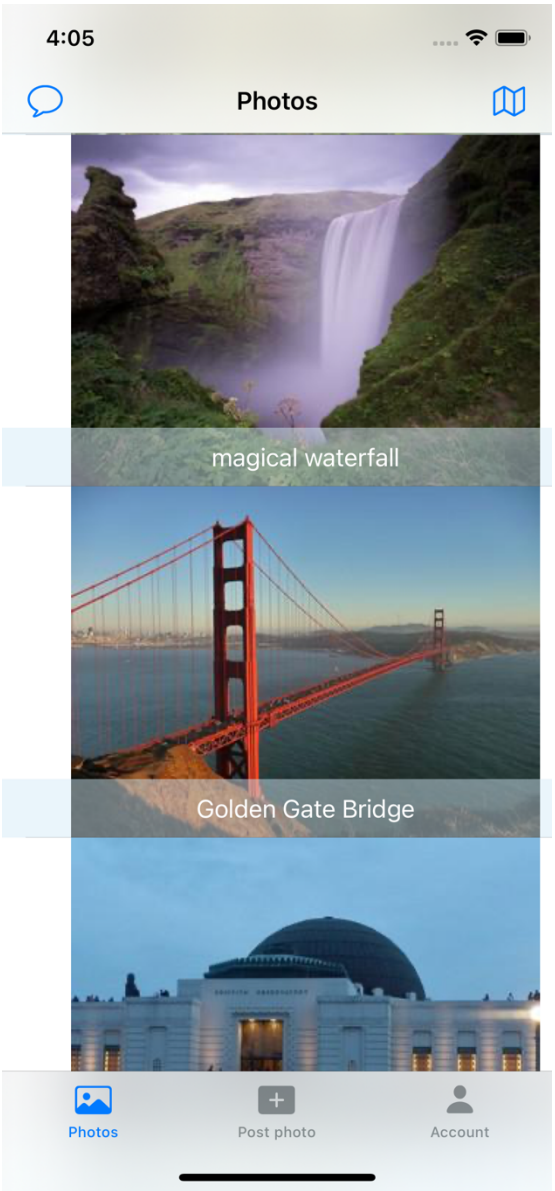
Part II: Code Organization and Implementation

4.1 UI interface and related technologies

4.1.1 Overall layout

We use the tab bar Controller to set up three pages for the application, namely Photos, post Photos, and account. And use Navigation Controller to switch sub-pages.

4.1.2 Photos Page (tab bar 1)



The technologies involved are

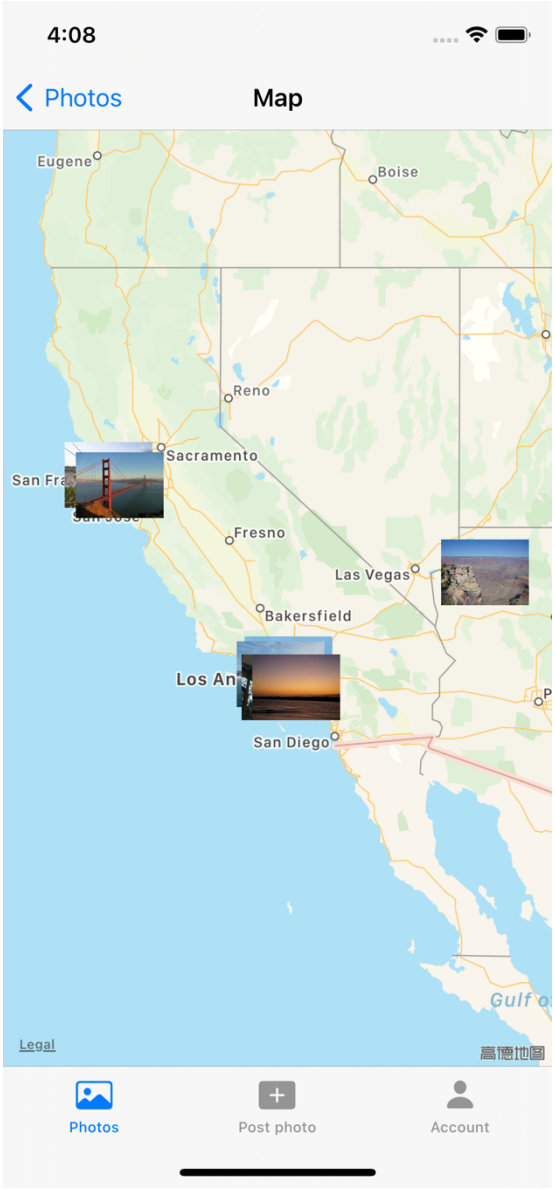
- TableView
- http Request
- Django Rest API for list
- PostgreSQL
- MapKit
- PostGis
- location coordinate
- Openweathermap

This page is a list. We display list item information through tableView

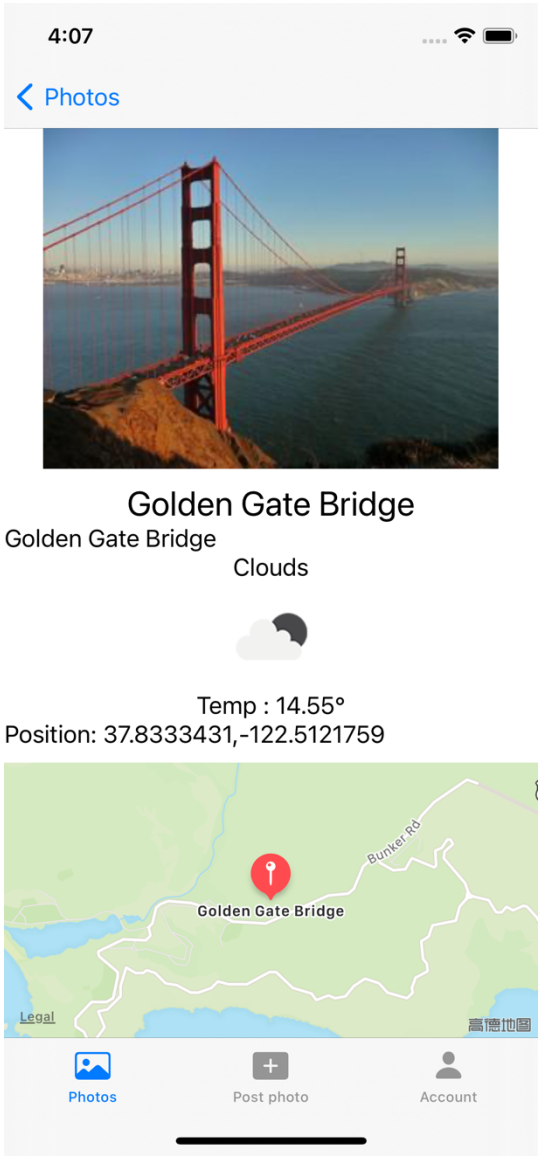
Each item contains an image and a title. The most recently added photo is at the top of the list. Whenever you enter this page, the back-end request interface will call Api, and the API will query data from the database and return it to tableView.

The number of list items is limited (8), because the memory leak caused by the excessive amount of data must be considered.

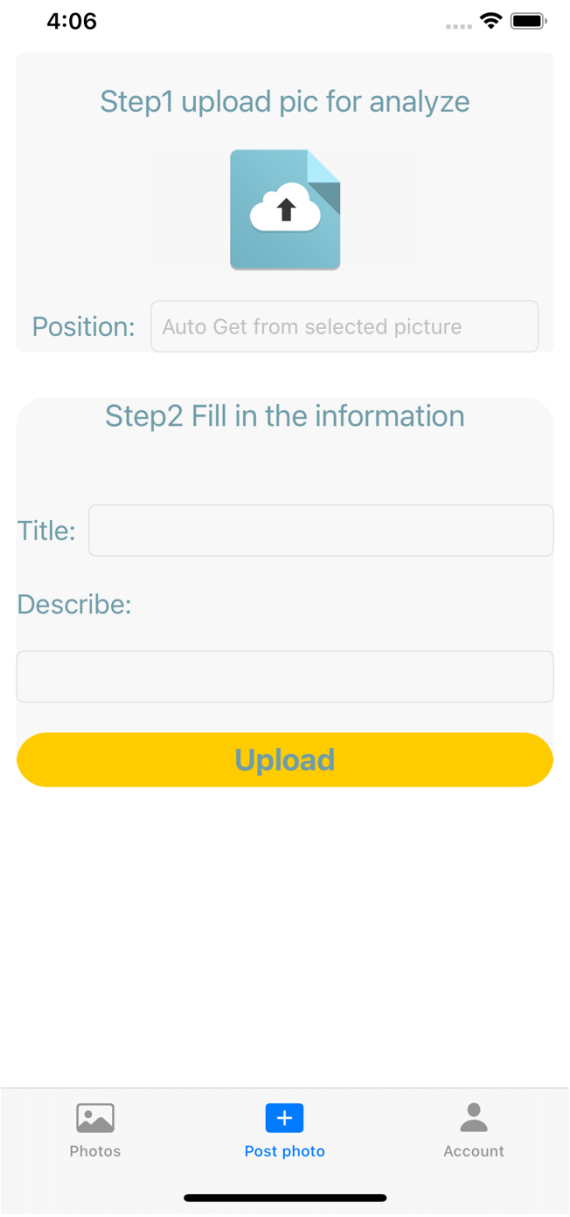
When you click the map icon in the upper right corner of this page, you will enter the map mode ， and the pictures in the Photos list will be displayed in various positions on the map.



When you click on each item in the list mode or click on each image at a different location in the map mode, you will be taken to an image detail page, which includes a larger image and a real-time weather forecast for the geographic location of the image. The weather forecast is implemented through Openweathermap's Api. There is also a small map at the bottom.


4:07
Photos
Golden Gate Bridge
Golden Gate Bridge
Clouds
Temp : 14.55°
Position: 37.8333431,-122.5121759
Bunker Rd
Golden Gate Bridge
Legal
高德地图
Photos
Post photo
Account

4.1.3 Post Photo (tab bar 2)



Post photo page

The technologies involved are

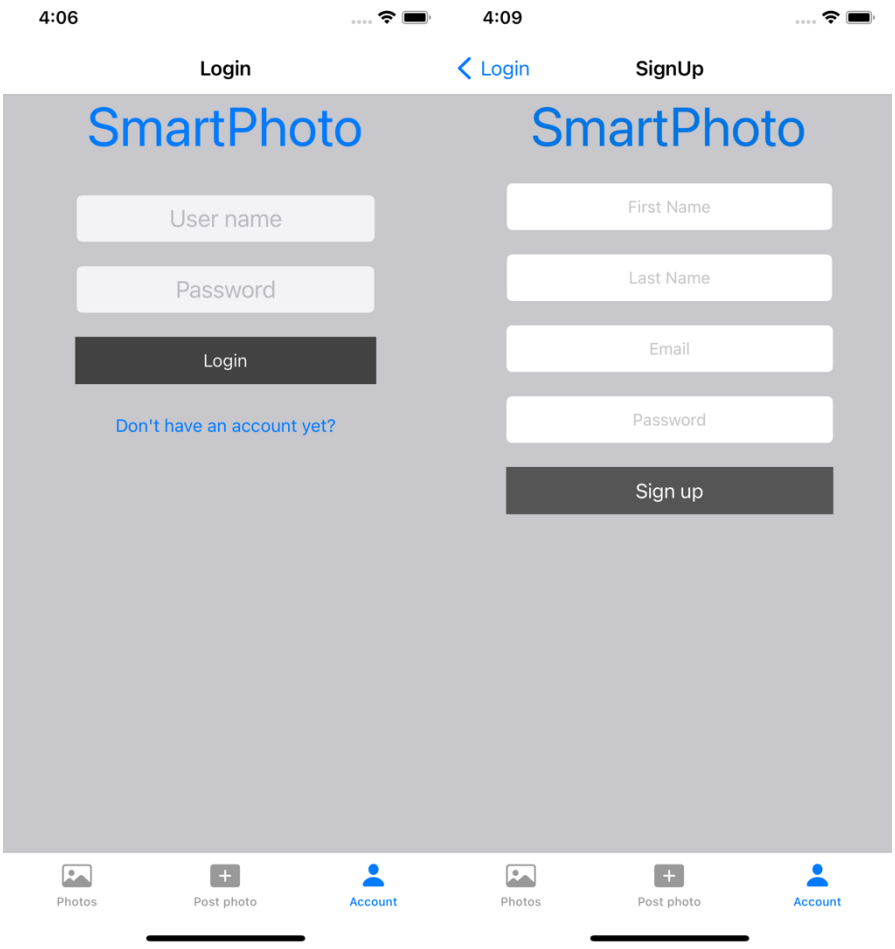
- PHPickerViewController
- UISession
- UIAlertController
- TabBarController
- Privacy
- Aws APIs
- Django APIs

The function of the page is divided into two steps

Step 1: Call the local photo library. At this time, you need to enable the iOS privacy permission reminder function Privacy - Photo Library Usage Description, when a photo is selected, the system enters the image analysis function. At this time, PHPickerViewController is used to obtain the Assets information of the image, including the longitude and latitude of the geographic location. At the same time, the image is uploaded to the AWS image server in the background to compress the image and return the URL of the image.

Step 2: Fill in the title and description information of the picture and click the upload button. At this time, the system will pass the filled-in information together with the URL result information of the first step to the Api in Json format and store it in the database. If successful, you will see a pop-up prompt box, which is implemented by UIAlertController, when you click OK, the page will help you return to the photos list page through TabBarController, this is where you will see your latest upload.

4.1.4 Account (tab bar 3)



The technologies involved are

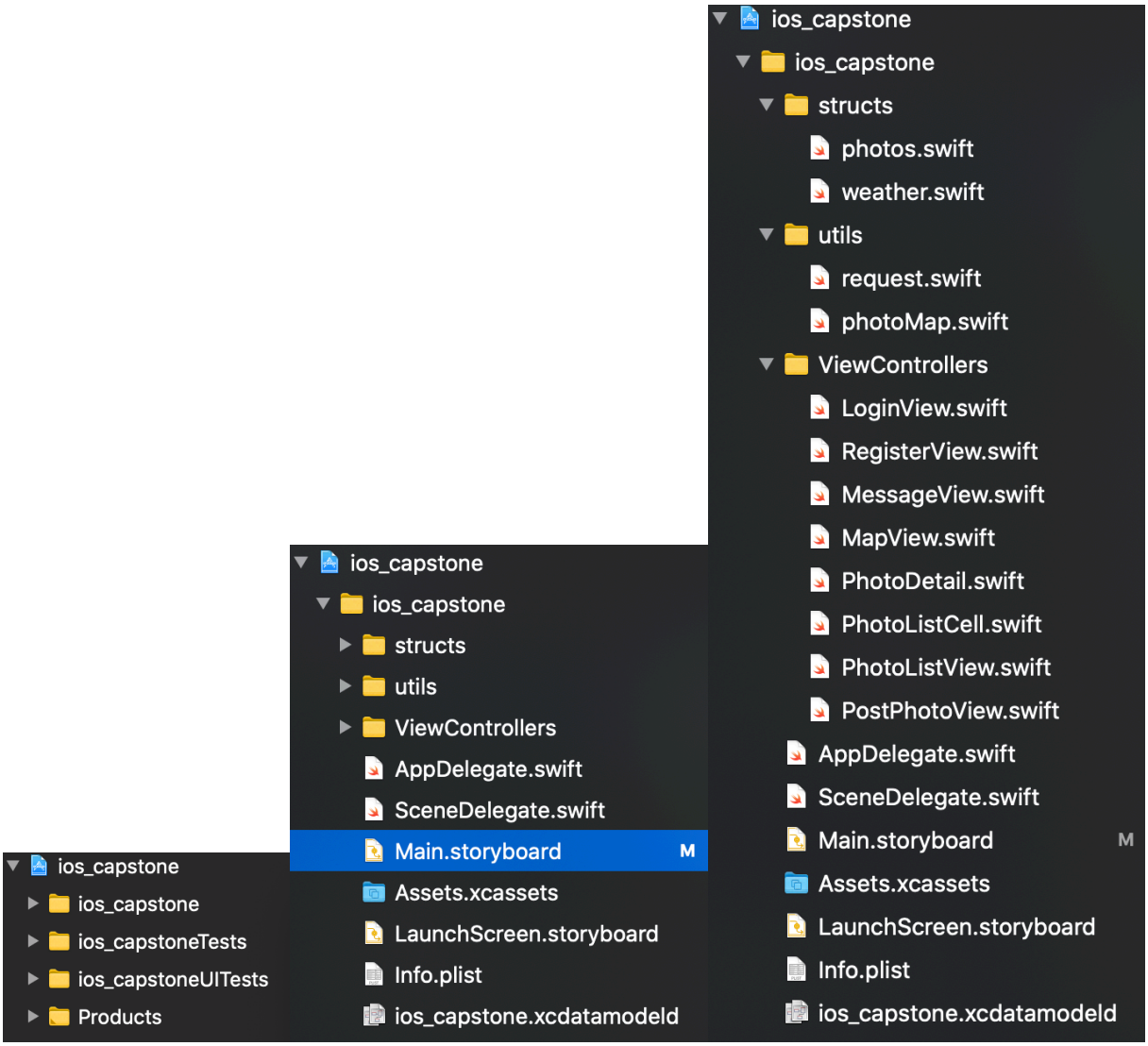
- text field
- button
- Django API
- Constraint layout

The content Type of the password field of the login and registration pages is set to Password, and the input process will appear in the form of origin.

Both communicate with the background database through Api and complete the login and registration by detecting the existing account information and the matching of username and password.

4.2 Code Organization and Implementation

The code directory structure expands from left to right



We have made a clear division of the code

It is composed of three parts, they are Structs, Utils, ViewControllers

Structs:

There are two files in the structure folder. The photo file defines the picture list paging structure, the link structure, the picture information structure, and the longitude and latitude structure. The weather file defines the weather information structure.

Structures make programs more scalable.

Utils:

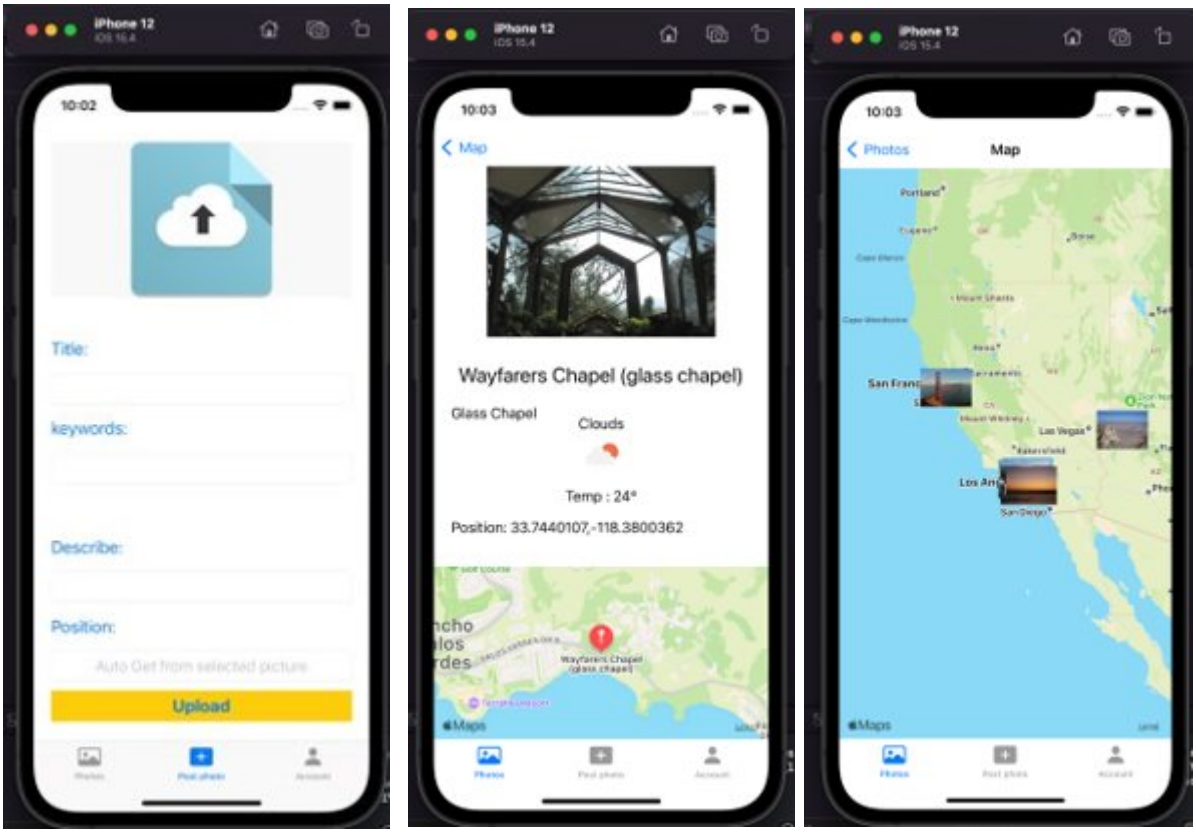
The Utils folder stores public packages. We have extracted a lot of common code to form a toolkit for easy reuse. Under this folder we have two parts:

The first part is the server Api request method, which includes the general web server request method (Django Rest Api) and the weather request method.

The second part is the map method, including placing the image into the map anchor point and resize images, etc.

ViewControllers:

The ViewControllers folder contains all our ViewController classes. Used to drive the behavior of each UI page.



5 backend

5.1 Data base design

Auth_user: save user identity, the password encrypted by sha256.

```
ioscapstone=# select * from auth_user;
 id |                                  password                                  |          last_login           | is_superus
er |          username          | first_name | last_name |           email           | is_staff | is_active |          date_joined
----+--------------------------------------------------------------------------------+-------------------------------+----------
---+----------------------------+------------+-----------+---------------------------+----------+-----------+-------------------------------
----
  1 | pbkdf2_sha256$260000$Rr7uk2BDcCPAizbrX5QVu5$wYU0hFg19h4PJoLb+R4b/rhVfAwLTTwFZa5EU7uGpzo= | 2022-08-04 00:46:48.609676+00 | f
   | liujin834@gmail.com        | Jack       | Li        | liujin834@gmail.com       | f        | t         | 2022-08-04 00:45:04.990651
+00
  5 | pbkdf2_sha256$260000$1tzf7NzSKMasvWnq5TLYOj$3Rg1u+nZh69IKhdxg6XpbWsBOYgVmvXVSZ2g/GIpWx4= |                               | f
   | jam@waffle.com             | iulia      | danilov   | jam@waffle.com            | f        | t         | 2022-08-11 03:24:21.469133
+00
  6 | pbkdf2_sha256$260000$3blzn3MfutA9aOCimnHlsT$9sllBrDY/rJRRzEJd0gHlCmWn9e1HvfZZXwPkj0dmLc= | 2022-08-11 04:22:55.494168+00 | f
   | test@capstone.com          | Jack       | Li        | test@capstone.com         | f        | t         | 2022-08-11 04:22:28.684097
+00
  8 | pbkdf2_sha256$260000$SprTdelEGGlYIT7skiWHIZ$ZghqbIYodVeOqe+xez2QiYhQ+eSPVuVs6D1vNU8+6+c= |                               | f
   | shooby@dooby.com           | Julia      | Danilov   | shooby@dooby.com          | f        | t         | 2022-08-11 15:10:17.55697+
00
 11 | pbkdf2_sha256$260000$Wu4BfT0P6t4uW2lnKbtv7z$K8Sc8ye9LP1+xKA2xzZdRd7FkYwovJZYlv4Syfi05MI= |                               | f
   | sdmehtacoc@gmail.com       | smit       | Mehta     | sdmehtacoc@gmail.com      | f        | t         | 2022-08-11 22:48:48.709595
+00
 12 | pbkdf2_sha256$260000$ml7l1PQ4ILnnCuh4LtWszs$FmrwFrb7dhQc74VOwZG5zzvGdFMaFV9bH6OvaV3FCcA= |                               | f
   | ks@gmail.com               | Krupa      | suhagiya  | ks@gmail.com              | f        | t         | 2022-08-12 02:10:09.264324
+00
 13 | pbkdf2_sha256$260000$1qV4VzjgymfMFr9DhpCHfB$L/4kBEukfVk5u/nvDAiF4HQ7XwBRGDy48SrMAMCudJ0= |                               | f
   | sm@gmail.com               | smit       | Mehta     | sm@gmail.com              | f        | t         | 2022-08-12 02:11:11.140126
+00
```

Photos: save photo url and position. The position saved with binary point format, which can be convert to (lat,lon) Double pair

```
ioscapstone=# select * from photos;
 id |            title             |     description      |               keywords               |         ts_created          | ts_changed |
author_id |                                  url                                  |                   position                   |
                          thumb_url
----+------------------------------+----------------------+--------------------------------------+-----------------------------+------------+-
----------+-------------------------------------------------------------------------+----------------------------------------------+------
----------------------------------------------------------------------------
  4 | Grand Canyon                 | Grand Canyon         | grand canyon                         | 2022-08-09 21:58:48.799267+00 |            |
         | https://capstone.freeyeti.net/media/9c412b24-fc91-44e1-b87a-fdb439994b2b.jpg | 0101000020E6100000DB615D26D4595CC086819D51A90B4240 | https
://capstone.freeyeti.net/media/9c412b24-fc91-44e1-b87a-fdb439994b2b_thumb.jpg
  5 | Seal Beach                   | Seal Beach           | seal beach                           | 2022-08-10 00:25:36.818878+00 |            |
         | https://capstone.freeyeti.net/media/c03a9091-97ad-4137-ae3c-e51ff9276cd0.jpg | 0101000020E610000082470A1B54875DC0E1C27064D4DE4040 | https
://capstone.freeyeti.net/media/c03a9091-97ad-4137-ae3c-e51ff9276cd0_thumb.jpg
  6 | Wayfarers Chapel (glass chapel) | Glass Chapel         | glass chapel                         | 2022-08-10 00:32:30.559219+00 |            |
         | https://capstone.freeyeti.net/media/150c3ed9-2b63-4bd2-9571-0b231a269391.jpg | 0101000020E6100000F4925A8352985DC0E12F1CBE3BDF4040 | https
://capstone.freeyeti.net/media/150c3ed9-2b63-4bd2-9571-0b231a269391_thumb.jpg
  7 | Santa Monica Beach           | Santa Monica Beach   | Santa Monica Beach                   | 2022-08-10 00:38:57.220577+00 |            |
         | https://capstone.freeyeti.net/media/69e96d86-5e6e-42d4-baa9-17f0229bc4e2.jpg | 0101000020E6100000398FD48CD69F5DC09A643FE65E014140 | https
://capstone.freeyeti.net/media/69e96d86-5e6e-42d4-baa9-17f0229bc4e2_thumb.jpg
  8 | Griffith Observatory         | Griffith Observatory | Griffith Observatory building mountain | 2022-08-10 00:42:57.338581+00 |            |
         | https://capstone.freeyeti.net/media/b51bc693-151f-42c4-876f-d8365f58cd63.jpg | 0101000020E61000006FCE2D196C935DC049B65FE39F0E4140 | https
://capstone.freeyeti.net/media/b51bc693-151f-42c4-876f-d8365f58cd63_thumb.jpg
 10 | Golden Gate Bridge           | Golden Gate Bridge   | bridge                               | 2022-08-10 00:49:52.809032+00 |            |
         | https://capstone.freeyeti.net/media/55aef9c5-1fcb-4a40-a273-a1c99be6a79c.jpg | 0101000020E610000029136D7DC7A05EC0736C98FCAAEA4240 | https
://capstone.freeyeti.net/media/55aef9c5-1fcb-4a40-a273-a1c99be6a79c_thumb.jpg
```

5.2 APIs

5.2.1 Photo upload API:

URL: https://capstone.freeyeti.net/api/photos/upload

Method: POST

Parameters:

photo={the photo data}

Description: require submit with multipart/form content

Return example:

```
{"image_url":"/media/9c412b24-fc91-44e1-b87a-
fdb439994b2b.jpg","thumb_url":"/media/9c412b24-fc91-44e1-b87a-
fdb439994b2b_thumb.jpg"}
```

5.2.2 Post new photo API:

URL: https://capstone.freeyeti.net/api/photos/

Method: POST

Parameters:

{

"title": "",

"description": "",

"keywords": "",

"url": "",

"thumb_url": "",

"position": null

}

Description: require submit with application/json content

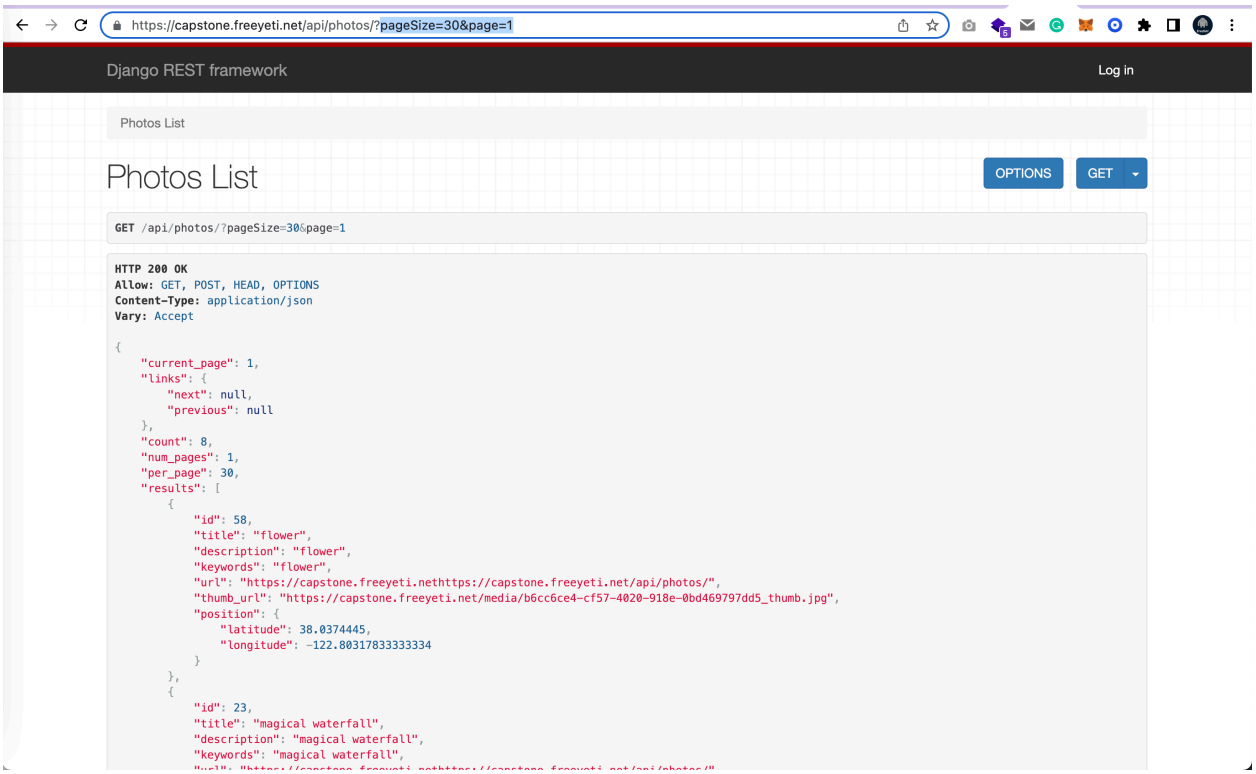
5.2.3 Photo List

URL: https://capstone.freeyeti.net/api/photos/

Method: GET

Parameters:

pageSize=30&page=1



5.2.4 Login API:

URL: https://capstone.freeyeti.net/api/account/login/

Method: POST

Parameters:

```
{
    "email": “String”,
    "password": “String”
}
```

Description: require submit with application/json content

5.2.5 Signup API:

URL: https://capstone.freeyeti.net/api/account/signup/

Method: POST

Parameters:

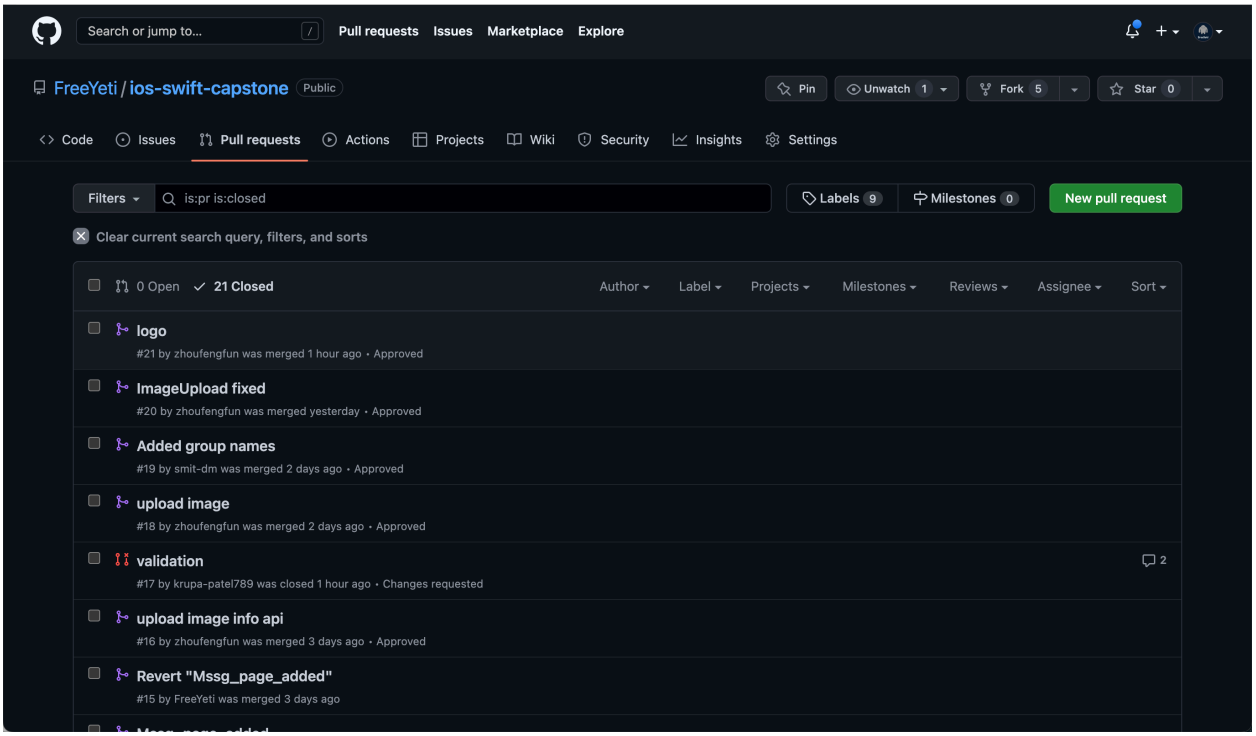
```
{
    "email": “String”,
    "password": “String”,
    "first_name": “String”,
    "last_name": “String”
}
```

Description: require submit with application/json content

6 Out of code

Git repo: We use github manage our git repository, and everyone in group use pull requests to merge our code. We have 21 PRs in the last version. In our development process, we solve conflict and make sure everyone’s work has been merged in every step.

Final Project Report



DevOps:

Because of we create backend APIs by ourself, so we need to do devops works by ourself. We are using Github Actions and Kubernetes to reduce our devops time. And in the end we have 14 versions of release for our backend.

Project analysis:

Our project had achieved most crucial part of our design, however the plan is still not fully finished. Good thing is our learned about team work on this project. And improved skills in Swift.