

IMT Atlantique, LIRMM

Internship Report

# SEADOGSEA WEBSITE

Montpellier, the 25 August

Hoang Duong,  
2A – Computer engineering student  
[hoang.duong@imt-atlantique.net](mailto:hoang.duong@imt-atlantique.net)

Supervisor: Marc Chaumont, Gérard Subsol,  
Cyril Barrelet, Joel Maizi, Vincent  
Raveneau



## **ACKNOWLEDGEMENTS**

I extend my heartfelt gratitude to CIRAD members (especially, Mr Michel de Garine-Wichatitsky) who provided me with materials, inspiration and the subject for this research, and also I cannot help but mention my esteemed tutors, (Mr Marc Chaumont, Mr Gérard Subsol, Mr Cyril Barrelet, Mr Joel Maizi and Mr Vincent Raveneau) for their exceptional guidance and unwavering support during my internship in the field of website development. Their expert mentorship has been a cornerstone of my success, as they skillfully navigated me through challenges and opportunities inherent to the realm of technology. Their patient explanations, insightful feedback, and encouragement have not only aided in the completion of my internship but have also significantly enriched my understanding of practical applications within the field. I am truly appreciative of the opportunities to contribute to real-world projects under their guidance, which has fortified my skills and bolstered my confidence. This experience at LIRMM has undoubtedly laid a robust foundation for my future endeavors in the dynamic landscape of computer science, and I am sincerely thankful for their role in shaping my growth and accomplishments.

# Contents

- I. Database management guidelines ..... 7**
  - 1. Pipeline of adding new data .....7**
    - a. General workflow ..... 7
    - b. User relevant .....7
    - c. Preparation of dog data .....8
    - d. Insert data to the database (This step must be done by obeying the order of the General workflow)..... 10
    - e. Add avatar data for dogs .....11
  - 2. User permission note ..... 11**
  - 3. Groups ..... 11**
- II. Website management guidelines ..... 13**
- III. User guide ..... 14**
  - 1. Homepage ..... 14**
  - 2. Login page ..... 15**
  - 3. Register page ..... 15**
  - 4. Reset password page ..... 16**
  - 5. Modify personal information ..... 17**
  - 6. First visualization page: Explore Data by Dog ..... 17**
  - 7. Second visualization page: Explore Data by Camera ..... 22**
  - 8. Feedback page .....24**
  - 9. Dog Identification page ..... 25**
  - 10. Admin page (ADMIN PERMISSION REQUIRED) .....26**
- IV. Release notes ..... 28**
  - 1. Side bar .....28**
  - 2. Header bar ..... 28**
  - 3. User&Group management (Admin page)..... 28**
  - 4. Homepage .....29**
  - 5. Explore Data ..... 29**
    - a. Explore Data by Dog ..... 29
    - b. Explore Data by Camera ..... 30
  - 6. Feedback .....31**
  - 7. Dog Re-identification ..... 31**
  - 8. Page access ..... 31**
- V. System doc .....33**

<b>1.</b>	<b>Directory hierarchy (Django based)</b> .....	<b>33</b>
<b>2.</b>	<b>Front-end</b> .....	<b>33</b>
a.	Overall .....	33
b.	Side bar .....	34
c.	Explore Data by Dog .....	35
d.	Explore Data by Camera .....	37
<b>3.</b>	<b>Back-end</b> .....	<b>39</b>
a.	Overall .....	39
b.	Routing controller (HTTPS) .....	40
c.	Routing controller (WSS) .....	41
d.	Django settings .....	42
e.	Homepage .....	43
f.	Login page .....	43
g.	Register and Reset password page .....	45
h.	Modify user personal information page .....	46
i.	Admin page (for user management) .....	47
j.	Explore Data Pages .....	50
k.	Feedback page .....	52
l.	Dog Re-Identification page .....	53
<b>4.</b>	<b>Database</b> .....	<b>53</b>
<b>VI.</b>	<b>Limit and improvements</b> .....	<b>55</b>
	<b>REFERENCES</b> .....	<b>56</b>

## FIGURES

Figure I-1: database schema for dog data (It's only main part of the database).....	7
Figure I-2 Example of setting camera variables in the insert_data script.....	10
Figure III- 1 Homepage UI.....	14
Figure III- 2 Register and sign in button on homepage.....	14
Figure III- 3 Login page UI.....	15
Figure III- 4 Register page UI.....	16
Figure III- 5 Register page UI.....	17
Figure III- 6 Side bar.....	18
Figure III- 7 Explore Data by Dog UI.....	19
Figure III- 8 Result after a click on a bar of the chart.....	20
Figure III- 9 Video timeline, on Explore Data by Dog page.....	20
Figure III- 10 Single-track mode on Explore Data by Dog page.....	21
Figure III- 11 Multi-track mode on Explore Data by Dog page.....	21
Figure III- 12 Path popup showing information after a click on the path.....	22
Figure III- 13 Give feedback.....	22
Figure III- 14 Explore Data by Camera UI.....	23
Figure III- 15 Result after a click on a camera bar on the bottom left chart.....	23
Figure III- 16 Feedback page UI.....	24
Figure III- 17 Dog Identification page UI.....	25
Figure III- 18 Sample result.....	26
Figure III- 19 Admin page UI.....	26
Figure IV-1 Check user's permissions in some pages.....	32
Figure V-1: Side bar flowchart.....	35
Figure V- 2: Initialize the Explore Data by Dog page.....	36
Figure V- 3: Explore Data by Dog page flowchart.....	37
Figure V-4: Initialize the Explore Data by Dog page.....	38
Figure V- 5: Flowchart after a click on the camera bar chart.....	38
Figure V-6: Flowchart of further actions after a click on the camera bar.....	39
Figure V-7: URL routing of the website.....	40
Figure V-8: Implicit URLs.....	41
Figure V-9: the difference between HTTP and websocket (WS).....	42
Figure V-10: Flowchart of login page in server side.....	44
Figure V-11: Flowchart of register page in server side.....	45
Figure V- 12: Flowchart of reset password page in server side.....	46
Figure V-13: Flowchart of admin page in server side.....	48
Figure V-14: Flowchart of rendering admin Page.....	49
Figure V- 15: Flowchart of activating a user.....	50
Figure V-16: Flowchart of deleting a user.....	50
Figure V-17: Example of relational tables.....	51
Figure V-18: Flowchart of Explore Data by Dog page in the server side.....	52
Figure V- 19: Flowchart of communication between front-end and back-end through websocket in Dog Re-Identification page.....	53
Figure V- 20: Example of storing STSD and clustering results.....	54



# I. Database management guidelines

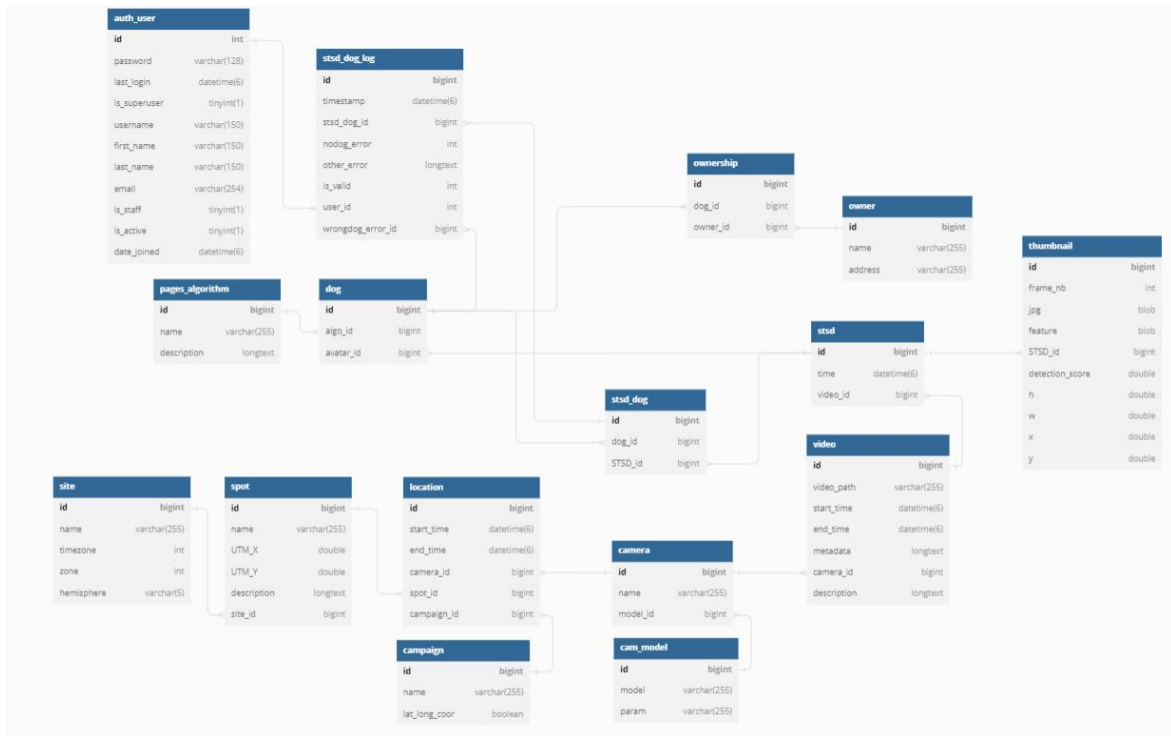


Figure I-1: database schema for dog data (It's only main part of the database)

## 1. Pipeline of adding new data

### a. General workflow

Add Campaign → Add Site (if the campaign is carried out in new site) → Add new spot (Note that there are possibly some old spots) → Add Camera → Add location → Add new algorithms if needed → Use scripts to add data of video, std, thumbnail, std\_dog (where we store the information about stds belong to which cluster (dog)), dog.

### b. User relevant

#### Step 1:

Add campaign: the name of the campaign MUST follow a standard: name-time

**Example:** SIBETAN-2019

#### Step 2:

Add permission that enable user to access the visualization page of new campaign:

Go Permissions → Add Permission → Type name and code name for permission, content type: pages | campaign.

Code name must be consistent with campaign name and comply with this standard: can\_access\_{campaignName}.

**Example:** Create a permission to access Sibtan (2019) campaign

Name: can access SIBETAN 2019

Codename: can\_access\_SIBETAN-2019

### Step 3:

Grant permission to valid groups of users (DO NOT FORGET to grant all permissions to **web admin group** and **database admin group**).

#### c. Preparation of dog data

Before starting, you must ensure the folder organization in which you will process the raw data like this (I've already sorted everything in the repo, and do not change the hierarchy of them):

```
|-- DataProcessing
```

```
    |-- scripts (where all the scripts to process are stored)
```

```
        |-- dogs_tracking
```

```
        |-- process_to_database
```

```
        |-- kmeans_convnext
```

```
        ...
```

```
    |-- campaign (where the results of this step are stored, e.g., you can name it SraeRuessei_ST_Oct2022)
```

```
        |-- dog_tracking (This folder contains only video from client's dataset. Because their dataset includes both images and videos)
```

*\*\*\* You must inspect the folder organization of client's dataset and obey it, in my case, all my folder results are organized by following the one of campaign Srae Ruessei 2022. If you have another folder organization, you must modify the scripts \*\*\**

You can choose to use Yolov5 + StrongSort or Yolov8 + BotSort, and it's vital to follow the guidelines if you don't want something too challenging.

For me, firstly, I ran Yolov5 + Strongsort but I got a huge number of misdetection and ID switching, then, I carried out some tests on Yolov8 + BotSort to limit the problems.

To start running some framework like Yolo or using ML model, it's necessary to *pip install -r requirements.txt*

Step 1: Filter video from raw data



- Create folder campaign and change the name of **camp**, **country**, **time of the campaign** in file **script/rename.py** (It will create dog\_tracking folder which filter only videos from the dataset)
- Execute **script/dogs\_tracking/yolo\_tracking/rename.py**.

Step 2: Use Yolo to detect dogs in videos.

*If using both Yolov5 + Yolov8:*

- After installing all required packages, change name of camp in **script/dogs\_tracking/Yolov5\_StrongSORT\_OSNet / db\_strongsort\_remake.py** and execute it. Output folder **campaign/get\_dog** (cropped image of dogs)
- Due to a lot of misdetections, run Yolov8: change name of camp in **script/dogs\_tracking/yolo\_tracking/filter\_dog\_yolov8.py** and execute it. Output folder **campaign/filter\_dog\_yolov8**.

*If using only Yolov8:*

- Remember to install all required packages, change camp name, and execute **script/dogs\_tracking/yolo\_tracking/yolov8.py**. Output folder **campaign/filter\_dog\_yolov8**.

Step 3: Pre-processing data before the feature extraction process:

- (Optional) Change camp in **script/dogs\_tracking/yolo\_tracking/clean\_dog.py** and run it. This step requires you to check manually the data after detection, to eliminate all the misdetections or ID switching. You can possibly check the data without using this script.
- Change camp and execute **script/dogs\_tracking/yolo\_tracking/filter\_only\_day.py**. Output **campaign/STSDs**, this script returns all valid STSDs [maximum 10 thumbnails (size bigger than 50x50) per STSD, you can modify this number in the script] which are taken during the day, not night, and only from videos that have only 1 dog.

Step 4: Add metadata (start time, end time, date, camera of videos) to the folder **STSDs**:

- Use **script/process\_to\_database/ocr.py** with a change of camp name to extract meta data from the videos.

Step 5: Run clustering:

- Change name of camp, clustering folder and carefully choose the optimal params for the clustering in file **script/process\_to\_database/cluster.py**.
- After that, execute the script. Output: folder contains cluster\_id.txt which points out all stsd belonging to cluster\_id

Note: The website now has only 1 clustering method (Agglomerative Clustering). In the future, it may have more than 1, therefore in case it has more than 1 method, it's better to

update data of all campaigns with the new method and store in the database to avoid **404 error (Explore Data pages)**.

For example: SIBETAN-2019 data is only clustered by Agglomerative Clustering, assume that you add K neighbor method to database. Then, on header bar of Explore Data pages will display this new method and if user clicks on it, it will response 404 error because there is no SIBETAN-2019 data linked to this new method unless you update it.

In case you already add new algorithm to the database, and you want to update data, you can modify the algorithm/clustering method inside the file **script/process\_to\_database/cluster.py**.

d. Insert data to the database (This step must be done by obeying the order of the General workflow)

- As usual, we must change the name of campaign in a script **script/process\_to\_database/insert\_data.py**. Moreover, there are some more steps to finish the process such as setting camera id (Id camera in database) based on its name.

```
camera_id = {  
    'C201': 15,  
    'C202': 16,  
    'C203': 17,  
    'C204': 18,  
    'C205': 19,  
    'C206': 20,  
    'C207': 21,  
    'C208': 22,  
    'C209': 23,  
    'C210': 24,  
}
```

*Figure I-2 Example of setting camera variables in the insert\_data script*

Here is an example, after you add camera traps into database manually (you can create a new script that automatically adds new camera into camera table), note their names and ids, then, change the camera\_id variable with respective values.

- Change database information to connect to the database if needed.

In case of failure when adding data, you can rerun the script by commenting commands 'insertdata(sql, value)' to skip the data which is already put into the database.

For instance, insertdata(sql, value): data of video, insertdata(sql1,value1): data of std, insertdata(sql2, value2): data of thumbnail, insertdata(sql3, value3): data of dog (cluster), insertdata(sql4, value4): data of STSD\_dog.

Note: if you want to update data with new algorithm, comment all insertdata(...,...) except for **sql4, value4** (to insert the result of new clustering method to STSD\_dog table)

Optional: you can also change a **batch** (default:5, that is a number of rows to insert to the database each 0.25s, the higher value will speed the process but more risks to get an connection error)

e. Add avatar data for dogs

Should you understand how the website works, you will know that each dog (cluster) has an avatar (which is a medoid of cluster, and it's used as the representative image of the cluster). Therefore, we need to add them into the database to retrieve data with less effort.

- Go to the main project source, in **pages/views.py**: uncomment the part for identifying the avatar of each dog, change the dog id range (new dogs) and then run server, then the data will be automatically added to the database. After finishing the process, re-comment this part as its initial state to avoid repeating the process.

## 2. User permission note

- **Admin**: can access all pages (including user management page) + give feedback + accept/reject feedback.
- **Staff member**: can access pages based on their group + give feedback.
- **Guest user**: can only access specific pages for guest users (Guest group) and can't give feedback.
- **No active user**: (already in Pending group, when created) the account isn't activated yet, needs permission from administrators.

## 3. Groups

- Web admin group: those having access to all visualization pages.
- Database admin group: those having all permission.
- (Currently), I divided users into groups based on areas, for each group, the users can access corresponding campaigns in the area:
  - Sibatana group: group of users working on Sibatana campaigns.
  - Cheytouch group: group of users working on Cheytouch campaigns.
  - Srae Ruessei group: group of users working on Srae Ruessei campaigns.Of course, you can define a new group and add respective permissions to the group as you want.
- Guest group: group of users not working on any campaign (guess user)
- Pending group: group for users waiting for validation (**No active user**)

### Note:

- Users of guess group/pending group DO NOT belong to any other group.
- If you want to assign guess user to any campaign (when working directly on database management), you should remove 'guess group' from his current group, add staff status and campaign group afterwards. (**Not recommended**, this should be changed in the adminPage interface: use admin account to login then choose admin tab in the side bar menu)

→ You can modify any of these groups but DO NOT DELETE **PENDING GROUP, WEB ADMIN GROUP AND DATABASE ADMIN GROUP!**

## II. Website management guidelines

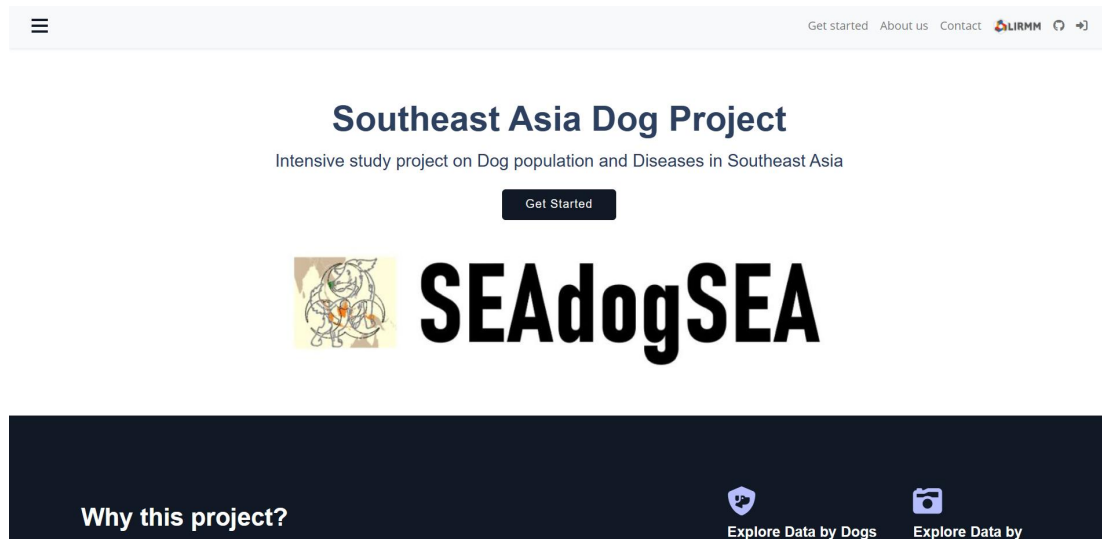
This section will help you to update the code from your local computer to the server.

- Firstly, clone the repo from gitlab: [https://gite.lirmm.fr/icar1/seadogsea/dog\\_project](https://gite.lirmm.fr/icar1/seadogsea/dog_project)
- Switch to the development branch Master
- After implementing new features for the website, push the code to branch Master, or you can create your own branch in the repo
- If everything is fine (IMPORTANT, avoid making some unrepairable mistakes on the server), you can push the code to branch Production
- Ask Mr. Maizi to request access the server through SSH
- Then move to the project directory on the server
- Pull code from repo
- Restart the server

In case you want to make some changes/update the relation, add new tables to the database, you can modify/add models in `pages/models.py` and then using the commands:`python manage.py makemigrates + manage.py migrate` to update the database. DO NOT directly modify the database.


### III. User guide

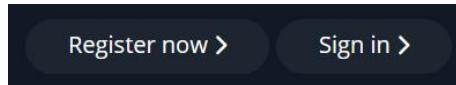
#### 1. Homepage



*Figure III- 1 Homepage UI*

On this page, you can find some general information about the project/website, contact.

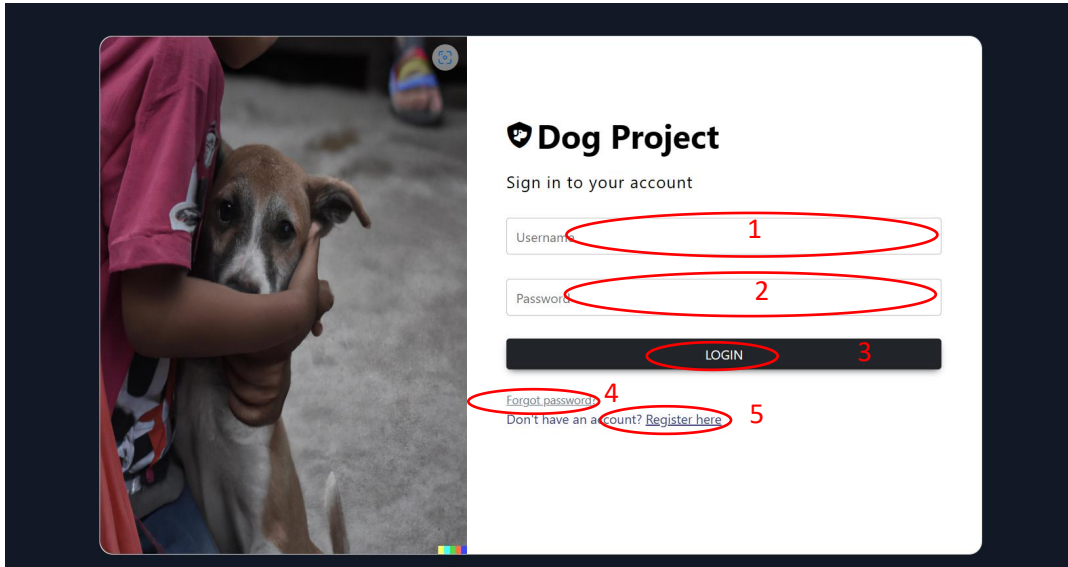
To access more features, you need to login, click  button on top right of the header bar, or you can click on Get Started button, here there are 2 options: register or login.



*Figure III- 2 Register and sign in button on homepage*

When you already signed in, this area in Figure III-2 will become logout button.

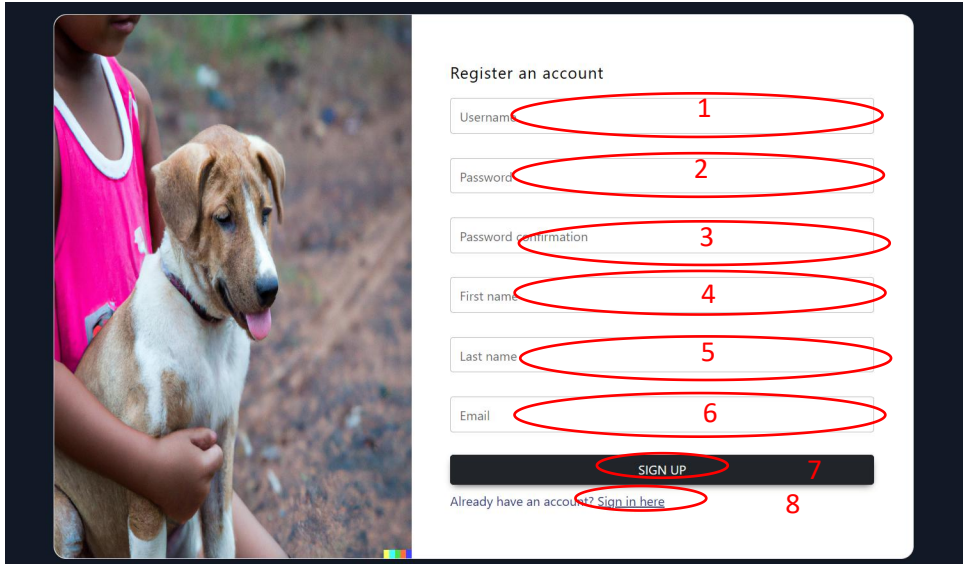
## 2. Login page



*Figure III- 3 Login page UI*

Enter your username [1], password [2] and press login [3] to log in.  
In case you forget your password, click on [4].  
If you do not have an account yet, you can register one by clicking [5].

## 3. Register page



*Figure III- 4 Register page UI*

You must enter all fields ([1],[2],[3],[4],[5],[6]) to be able to register, then, click on [7] to finish the process.

Username, password, and email must meet requirements:

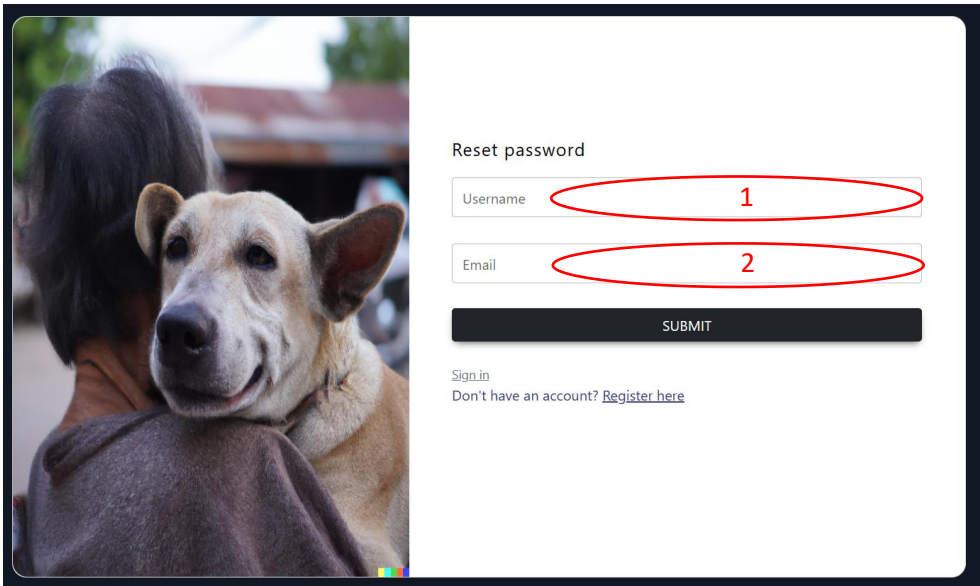
- No space, gap accepted in username
- Password: not a part of other personal information, at least 8 characters, not a commonly used password, not entirely numeric
- Email must comply with the standard form: abc@xyz...

You can also go back to login page by clicking [8].

When registering successfully, an email will be sent to administrators to inform them of the new registration request.

#### **4. Reset password page**






*Figure III- 5 Register page UI*

To reset your password, you must enter your registered username and email ([1], [2]) to recover your account. An email including a new password will be sent to the entered email.


## **5. Modify personal information**

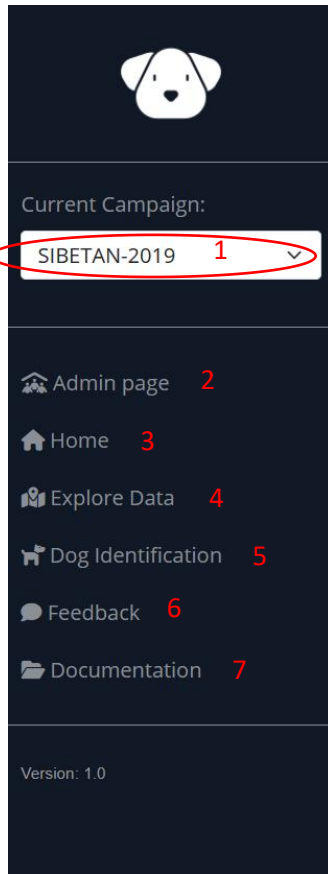
In case you have already logged in, an icon  will be added to the header bar (next to logout button). Click it to modify your email, first and last name or change your password if you have made an error while creating an account.

## **6. First visualization page: Explore Data by Dog**

When you are assigned to specific campaigns by one of administrators, you can access the page to explore data of the campaigns.

Assume that you are permitted to access SIBETAN and SRE RUESSEY data.

Login to your account, from now on, you can access the page by clicking  (top left of the header bar) to expand a side bar (Fig below).



*Figure III- 6 Side bar*

The first thing you need to do to access your desired campaign is selecting the campaign [1]

Only administrators can access [2]

[3]: Homepage

[4]: navigate to Explore Data by Dog page

[5]: Dog Re-Identification page

[6]: Feedback on data by campaign

[7]: Reference document about project/website.

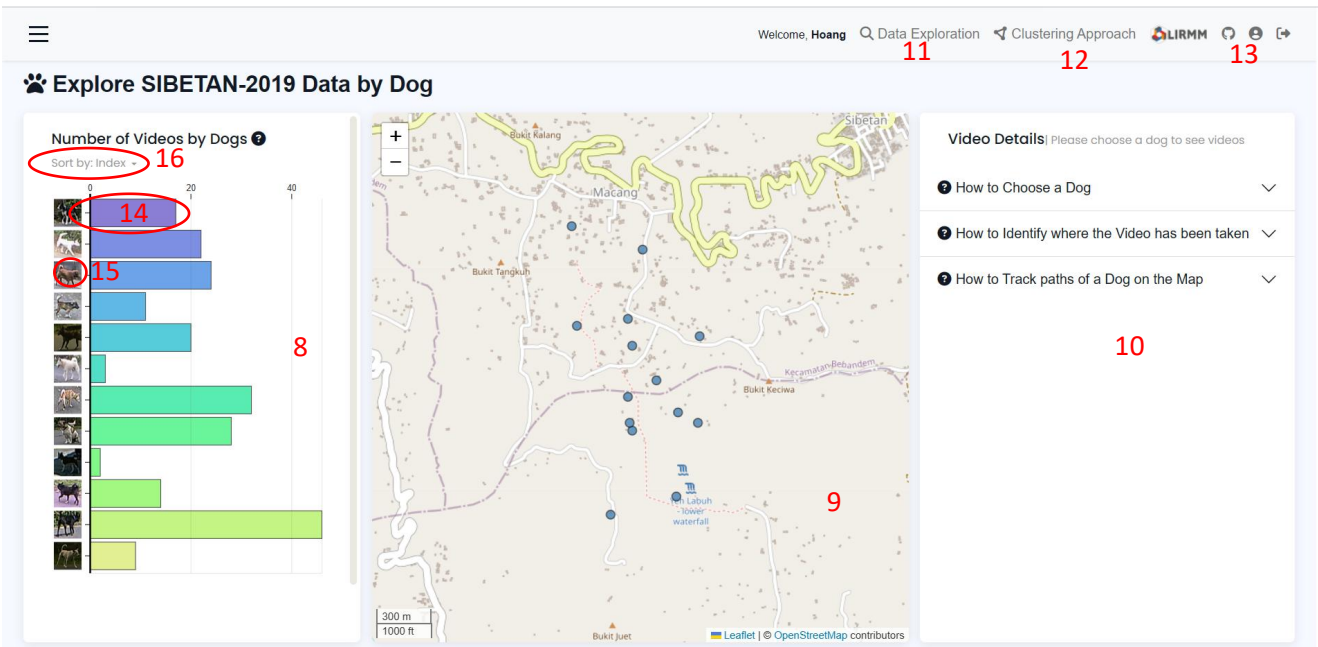


Figure III- 7 Explore Data by Dog UI

By clicking on [4], you can see the interface like the Figure III- 7

[8]: An area shows a bar chart (number of videos by dog, here “dog” means cluster, we use specific algorithms to group videos that may show the same dog, then, set an index to the cluster. Moreover, each dog has an avatar [15] (representation thumbnail)

[9]: A map illustrates trap camera position during a campaign.

If you hover on a bar chart with the mouse, it will give you the exact number of videos for a given dog.

If you hover on a blue spot (map), you will know the camera id.

[10]: A division shows some instructions. It will show video list of a dog when you click on any bar of the bar chart

[11]: It allows user to switch between visualization pages (Explore by Dog or Camera)

[12]: You can change clustering method but for now, we have provided only 1 method: Agglomerative Clustering

[13]: Logout

[15]: Avatar of a dog, when clicking on it, you can see all thumbnails of this avatar

[16]: Sort a chart by index or the number of videos for each dog.

[14]: A bar chart shows the number of videos of a given dog, after a click on a bar, all camera positions where this dog was captured will be displayed on [9], and the video details are also shown on [10] like the figure below.

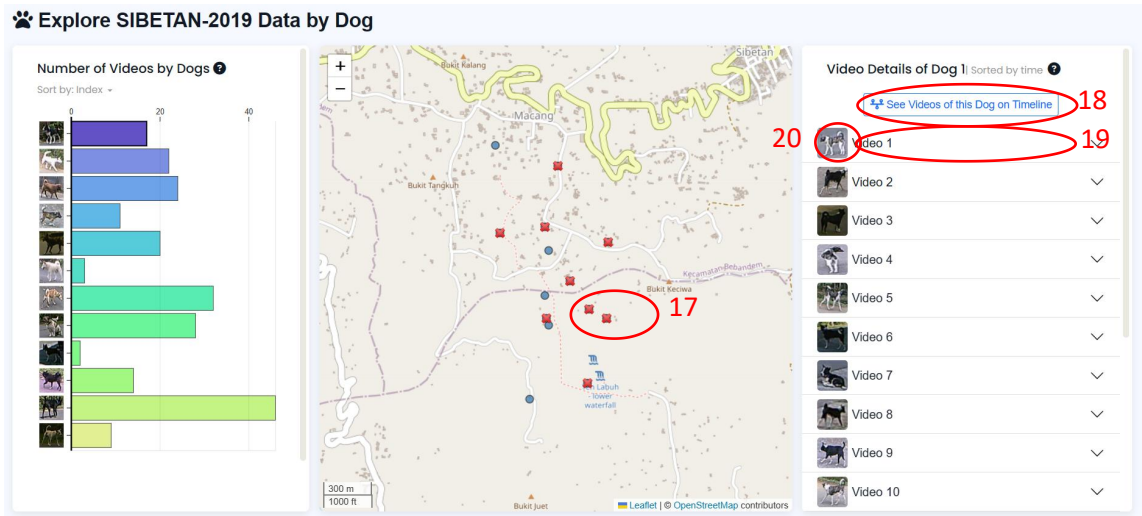


Figure III- 8 Result after a click on a bar of the chart

After a click, a full video list of the clicked dog will appear on the right panel. They are videos recorded by trap camera during the campaign and sorted by time (from the earliest to the latest)

The X symbol [17] indicates the examined dog has not been to this place during the campaign.

By clicking on [18] you will see the videos of the dog displaying on the timeline. It is possible to pan or zoom in/out. The figure below shows how it looks like.

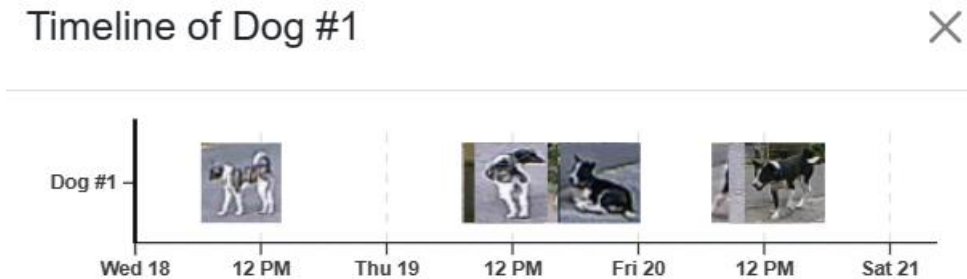


Figure III- 9 Video timeline, on Explore Data by Dog page

[20]: Dog thumbnail, clicking on it will open the modal showing all dog thumbnail of the clicked video.

You can expand the video accordion [19] by clicking on it, showing the video details such as the video is taken by which camera as well as the start time and end time of the video.

From now on, there are 2 modes of usage: single and multi-track on the map [9]:

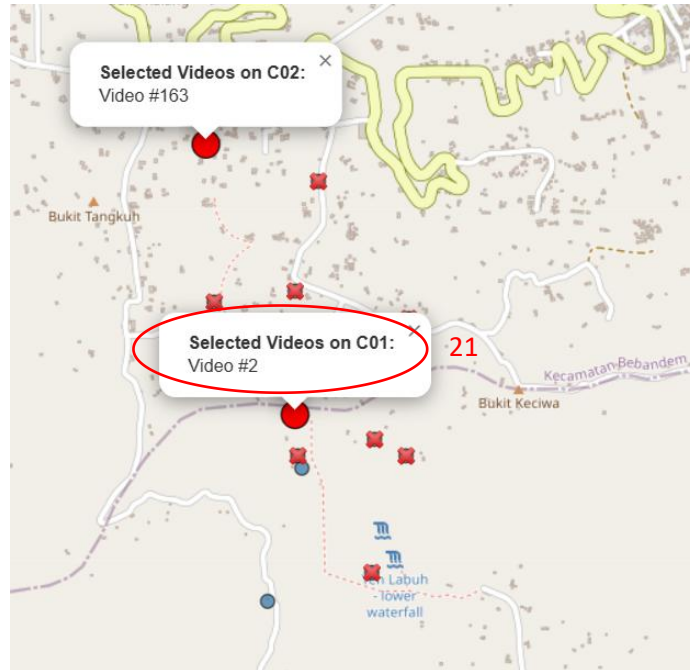


Figure III- 10 Single-track mode on Explore Data by Dog page

- Single-track mode: when you expand UNCONSECUTIVE videos in the video list (right panel) (e.g., video 1,3,5...), it will show you the camera positions where the videos are recorded (Figure III- 10). Note that the video ID shown on the popup [21] is the ID of this video on the database (global ID) not the video order that you can see on the video list.

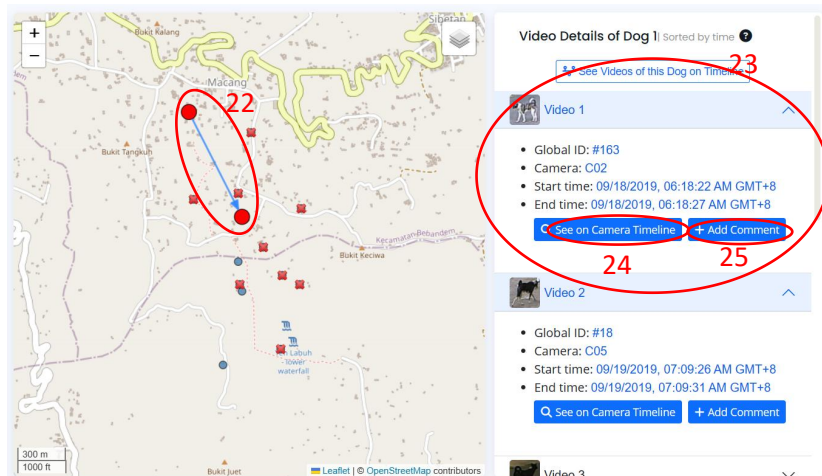


Figure III- 11 Multi-track mode on Explore Data by Dog page

- Multi track mode: it is enabled if you click on CONSECUTIVE videos, then, a path [22] between cameras emerges on the map to estimate the dog's trajectory (Figure III- 11). Clicking on the path will trigger the popup showing the distance between cameras, the time difference between 2 activated videos.

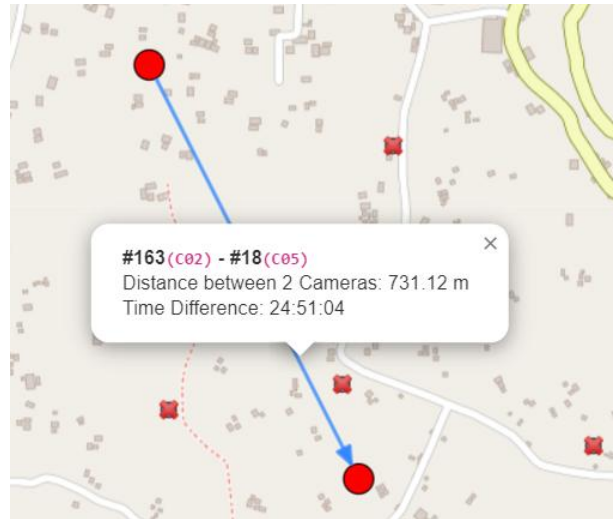


Figure III- 12 Path popup showing information after a click on the path

Look at the expanded video, for example [23], you can find 2 buttons [24][25]:

The first one [24] is to direct to the 2<sup>nd</sup> visualization page (Explore Data by Camera) where you can view the video on the timeline (highlighted with a red outline). This timeline will be explained more later.

The latter button [25] will be useful provided that you find any abnormalities in the video and want to add comments.

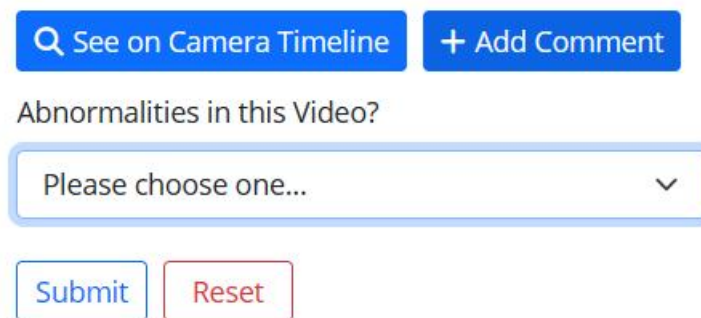


Figure III- 13 Give feedback

After the button (add comment) is clicked, some questions appear to collect your feedback. The feedback will be submitted and saved in our database to help admins correct it. The result of this feedback is shown on the Feedback page.

Note: feedback from one user for 1 specific video can be modified until admins admit or reject it.

## 7. Second visualization page: Explore Data by Camera

To be able to access this page, you need to click on [Data Exploration](#) (on header bar, 1<sup>st</sup> visualization page) then select Explore by Camera.



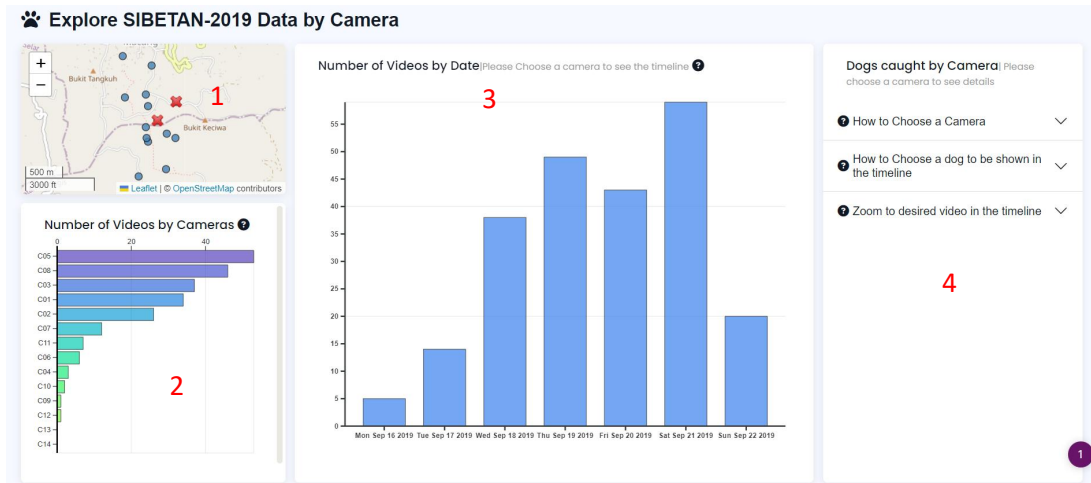


Figure III- 14 Explore Data by Camera UI

You can find here 4 main sections.

[1]: A map shows the position of all cameras. The symbol **X** on the map indicates there was no dog caught at this place.

[2]: The number of videos by camera chart. For example, the camera C05 took over 50 dog videos in the SIBETAN 2019 campaign.

[3]: A chart illustrates the number of dog videos taken by day. This area will be replaced by camera timelines after a click on the bottom left bar chart.

[4]: Some instructions, this area will be replaced by dog details.



Figure III- 15 Result after a click on a camera bar on the bottom left chart

The position of the active camera will be highlighted on the map [5].

When clicking on a bar of a camera, a camera timeline appears. The timeline in the middle of the interface has the time scale as x axis and y axis dedicated for different dogs (clusters) that are caught by this camera.

[7] You can enable a dog to be shown on the timeline or hide it by checking corresponding boxes.

[6]: Video thumbnails, when you click on specific thumbnail, the right panel will show the video details (like what it does in 1<sup>st</sup> visualization page).

[8]: Dog details, in this area, you can expand a dog to see its videos. When you click on any video, the timeline will automatically focus on the video thumbnail.

Note: on the timeline, it is possible to pan left/right, zoom in/out by the time scale axis

## 8. Feedback page

This page displays feedback from users that report abnormalities of dog videos in the 1st visualization page. Return to section III.6, figure I-11 to discover how to give feedback.

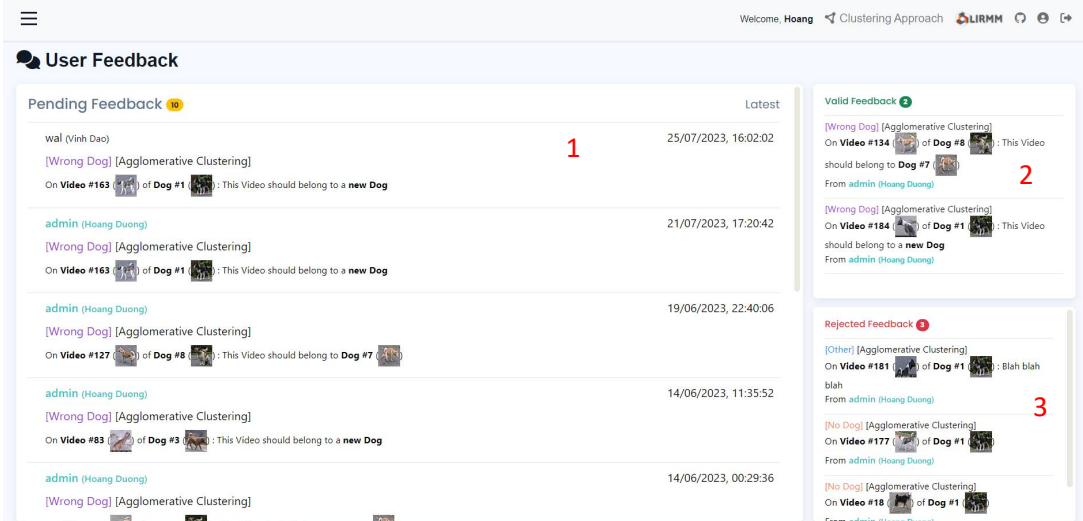


Figure III- 16 Feedback page UI

We have different feedback for different campaigns and algorithms.

[1]: pending feedback area, in which the feedback is waiting to be evaluated by admins. An admin can accept valuable feedback or reject wrong/spam feedback or even delete it by clicking on feedback, these icons will appear afterwards at the end of feedback    (For further study, this kind of feedback will support data correction, improving the confidence of clustering method...). Each feedback complies with a form:

Username of author (first name + last name)

[Type of abnormality] [Clustering method]

On Video #xyz of Dog #abc: feedback of user

Note: your username will be highlighted (cyan) to distinguish your feedback from others.



[2]: valid feedback area shows feedback accepted by admins.

[3]: rejected feedback → rejected by admins.

Note: As mentioned, you can modify feedback until admins validate it.

## 9. Dog Identification page

This page provides the functionality that predicts the most similar dog in specific campaign based on your input dog images.

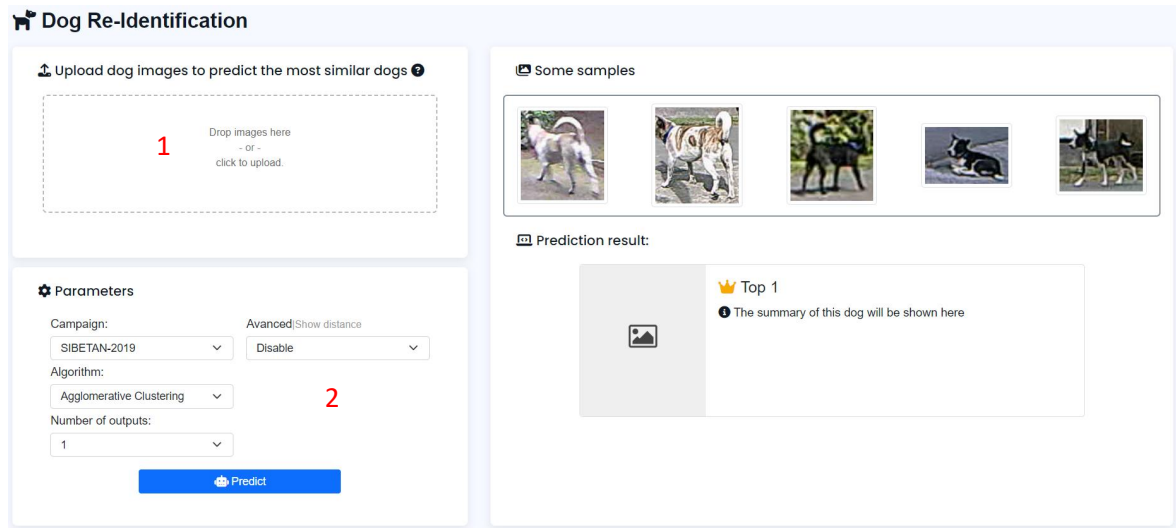


Figure III- 17 Dog Identification page UI

[1]: Drag and drop your images to this zone. Then, you can crop it if the image has a noisy background by hovering on the image. (max images: 12, max size each image: 1920x1080, 3Mb)

[2]: In this section, you can choose a target campaign to find the most similar dogs in this campaign to your dog input.

You can increase the number of outputs to show multiple prediction outcomes (for example, set this param to 3 to show 3 dogs that look like your input the most)

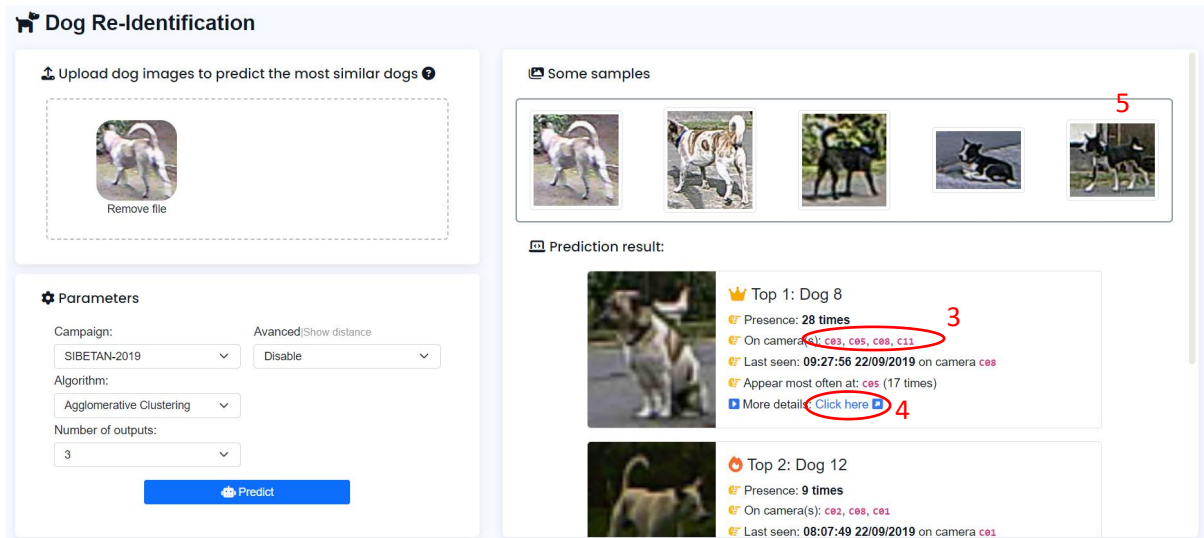


Figure III- 18 Sample result

The figure above shows the result with 3 outputs. We can see some summarized information about the dogs.

[3]: you can hover on these cameras to see their positions during the campaign.

[4]: click it will navigate to Explore Data by Dog page (with the right panel showing the dog details)

[5]: some samples, you can drag these images to the drop zone to test it.

## 10. Admin page (ADMIN PERMISSION REQUIRED)

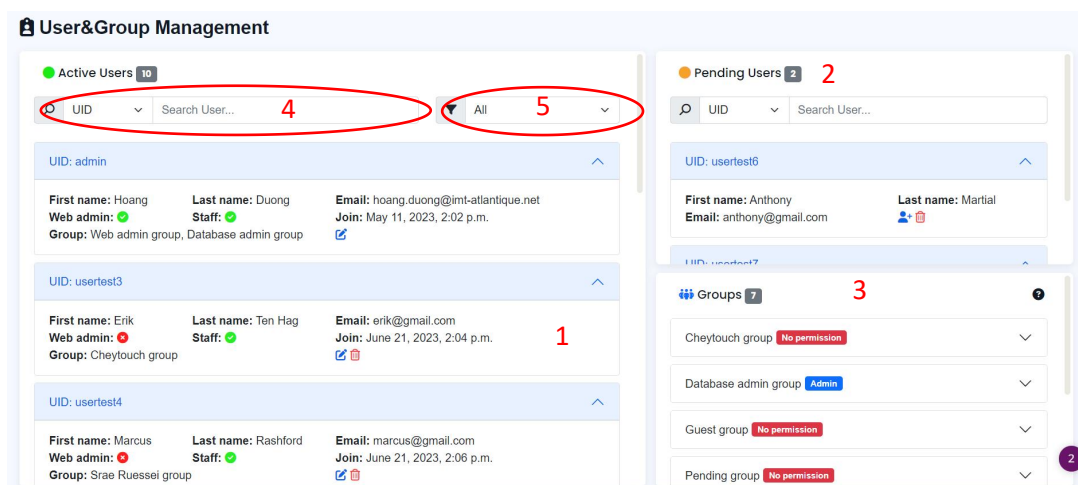




Figure III- 19 Admin page UI

This page can be used to manage, authorize, activate or assign user to specific groups.

[1]: active users. Modify a user by or delete by .

Note: in this page,

- It is impossible to delete Database admin account or change their group.
- Can not grant Database admin role to any user.
- Can not delete your account by yourself.
- If you assign a user to the guess group, other groups (if any) will be ignored.

[2]: pending users, waiting for activation to be able to access pages. Activate a user by  or delete by 

[3] user groups, containing permissions to access pages. For the moment, groups are divided based on available campaigns so that users belonging to that group can access only pages that visualize corresponding data. For example, users of Sibetan group can only view Sibetan data. On the other hand, this area will show group permissions. Web administrators cannot create new groups, nor modify permissions of groups. It is mandatory to carry out these tasks by Database administrators.

Note: 1 user can belong to many groups.

[4] Search bar for searching user by username (UID), last name, first name or email.

[5] Filter user by group

## IV. Release notes

### 1. Side bar

- Campaign selector (use cookie to store campaign name)
- Admin page (if the user has the **Can access admin page** permission)
- Home
- Explore Data – Area – Campaign
- Dog Identification
- Feedback – Area – Campaign
- Documentation
- Reference

### 2. Header bar

In general:

- Welcome + user first name (if signed in)
- LIRMM icon → direct to [LIRMM website](#)
- Github icon → direct to [my Github](#)
- User modification
- Logout (if signed in)/Login

Based on the page on which user is, there would be some more elements that user can interact with.

In visualization pages, there are 2 other tabs:

- Data Exploration: to switch between exploring data by dog or camera.
- Clustering Approach: to switch between clustering algorithms. For the moment, it has only 1 algorithm: agglomerative clustering. (This tab will be available in feedback page)

### 3. User&Group management (Admin page)

- Only Database/Web admins or groups/users that have the **Can access admin page** permission can access this page.
- One access the page will be able to:
  - Active, assign pending users to groups.
  - Change user group/personal information (name, email)
  - Delete users.
  - View group permissions (at this time, this page can not modify/add permissions to groups or create a new group. This can be done by Database admin through **admin Django page**)
  - Filter group/find user by UID/First name/Last name/Email.
- Web admin cannot delete **himself/database/other web admin user**, cannot change his group and the groups of **database admin user**.
- Web admin cannot assign Database admin role to any users. (This role can only be assigned by Database admin).

- Only Database admin can modify group/permissions through **admin Django page. (<http://host/admin>)**

#### 4. Homepage

- Title section, introduction section, team section (about us), contact section: present and summarize general information, functionalities about the project/website.
- Header bar:
  - Can view user role (if signed in)
  - Get started (navigate to introduction section)
  - About us (navigate to team section)
  - Contact (navigate to contact section)
- Button up (emerge bottom left of the page when scrolling down → back to the top)
- In introduction section:
  - If not signed in → Register/Login button
  - If signed in → Logout button
- Team section: display image and contact information of members in the team when hovering

#### 5. Explore Data

- Explore Data by Dog
- By selecting desired campaign, then navigating to Explore Data, both on the side bar, will direct user to this page.
  - Header bar:
    - Users can change to Explore Data by Camera by switching mode in Data Exploration tab.
    - User can change to other available Clustering method by clicking Clustering Approach
  - There are 3 divisions in total: a bar chart, a map, and the last one shows instructions/video details.
  - On 1<sup>st</sup> division (left), number of videos by dogs (clusters) chart:
    - Clickable, hoverable bars
    - Clickable dog thumbnails
    - Sort bars by index/number of videos
    - Active a bar (highlighted) when clicking it. Only 1 bar is activated at the same time to show dog details.
    - Change activated bar by clicking another one. Double click on the same bar will deactivate it.
    - ? popup to show instructions.
  - On 2<sup>nd</sup> division (middle), the map show position of trap cameras:
    - Hoverable dot (position of trap cameras, blue → red and showing the camera's name)
    - When clicking a bar in the 1<sup>st</sup> division, a symbol X may be shown on some positions in which the camera did not catch the dog during the campaign.

- On the last division (right), initially shows some instruction:
  - Show a list of videos of the dog of the activated bar (already sorted by time).
  - The thumbnails of videos are clickable.
  - Each video in the list is expandable to see details (global id, camera, start stop time), giving feedback button, see video in timeline (Explore by Camera, highlight the border of the thumbnail of the video, tooltip displays video information)
  - When a video is expanded, the corresponding camera position where the video is captured will be highlighted on the map.
  - When multi-mode is activated (consecutive videos are being expanded), paths which connect camera positions emerge to estimate the tracks of the dog.
  - Display videos on timeline button.
  - ? popup
- b. Explore Data by Camera
- There are 4 divisions in total: a map, a chart shows the number of videos captured by each camera, the biggest one illustrates a chart (number of videos by date)/timeline of each camera, instructions/dog details are displayed in the last one.
- The map shows position of trap cameras:
  - Hoverable dot (position of trap cameras, blue → red and showing the camera's name)
  - A dot will be highlighted if the corresponding bar in the number of videos by camera chart is clicked (activated).
- The number of videos by camera chart:
  - ? popup
  - Clickable, hoverable bars
  - Active a bar (highlighted) when clicking it. Only 1 bar is activated at the same time to show dog details.
  - Change activated bar by clicking another one. Double click on the same bar will deactivate it.
- The number of videos by date chart/timeline:
  - The chart:
    - ? popup
    - Hoverable bars
    - It will be replaced by a timeline of a camera when the camera bar chart is clicked.
  - The timeline:
    - X-axis is timescale, Y-axis is ordinal scale show dogs caught by the camera.
    - Panable, zoomable
    - Enable/Disable to show or hide specific dogs.
    - Hoverable, clickable thumbnails (each thumbnail represents a video)

- When clicking on the thumbnail, the dog whose video belongs to will be expanded to blink the corresponding video.
- The last division is for instructions/dogs caught by the clicked camera:
  - The dog list contains all dogs caught by the camera.
  - Clicking on any dog shows all videos of this dog recorded by the camera.
  - Clicking on any video will focus the video thumbnail on the timeline + highlight.

## 6. Feedback

- Show user feedback from Explore Data by Dog page in specific campaign.
- 3 divisions: pending, valid, reject feedback.
- Header has clustering approach tab allowing switch between feedback of different algorithms.
- Only admin can evaluate (accept, reject, or delete) feedback.
- Admin can return the evaluated feedback back to pending state.
- After the feedback is examined, the user who give the feedback can no longer modify it.

## 7. Dog Re-identification

- Drag and drop multiple image input (max files: 12, max size per file: 3mb, 1920x1080, accepted file: jpg, jpeg, png)
- Provide some image samples.
- Options to choose campaigns, algorithms, number of output (max: 3), disable/enable distance (between feature vectors of input and output)
- Websocket to communicate between frontend and backend (track progress)
- Show some information of the output dog (using some queries in the database)
- Can hover on camera name to show the camera position in the selected campaign.
- More details → navigate to the Explore Data by Dog page but activate the dog initially.

## 8. Page access

For the moment, it allows the user to access specific campaign page based on their group (the group has permission to access page, e.g., Sibatana group has the permission **'Can access SIBETAN 2019'**, code name: **'can\_access\_SIBETAN-2019'**) then I use `has_perm` to check user permissions.

For the future work, if there are more and more campaigns, it's better to modify a database such as add foreign key that refer campaign id in Group table, so we don't need to restrict the way to name permission or campaign as I mentioned in section **User relevant**, then, instead of checking permission based on its name (to access some restricted pages), we check the campaign id/name through user group.

For example, we have a function that checks user's permissions in the figure below:

```

@cache_control(no_cache=True, must_revalidate=True, no_store=True)
@login_required(login_url='/login')
def ExploreDataByCamera(request, area, campaignName, algoId):
    if not request.user.has_perm(f'pages.can_access_{campaignName}'):
        messages.error(request, "Sorry, you aren't the member of this campaign so you won't be able to access it. Ask admin for more informa")
        return redirect('home')
    return getDataforExplorePages(request, area, campaignName, algoId, 'ExploreDataByCamera.html')

```

*Figure IV-1 Check user's permissions in some pages*

It can be changed to (after modifying the database):

```

for group in request.user.groups.all do
    if group.campaign.name = campaign name do
        return page
    end if
end for
return error

```



## V. System doc

### 1. Directory hierarchy (Django based)

dog\_project (root)

|- dog\_project

    |- settings.py (setting for the system)

    |- urls.py (routing)

    |- asgi.py (Websocket config)

    |- ...

|- pages

    |- models.py (model, ORM)

    |- views.py (controller)

    |- routing.py + consumer.py (websocket implementation)

    |- ...

|- static (static files such as images, js, css files)

    |- css (contains all css files of the website)

    |- data (json, images)

    |- js (javascript files)

|- templates (all html files, view)

|- manage.py

### 2. Front-end

#### a. Overall

In front-end part, it is useful if we leverage some available powerful frameworks like Bootstrap, D3

The website is designed using these frameworks:

- Bootstrap v5.0.2 → Support in frontend responsive design with less code
- Font awesome v6.4.0 → Icon
- D3js v7 → Data driven and visualization.
- Leaflet v1.9.3 → Map
- JQuery v3.3.1 → Traverse and manipulate the HTML DOM tree

Besides that, I also use supplementary library to decorate the website:

- D3 color
- D3 interpolate
- D3 scale chromatic
- Popper.js v1.14
- JQuery Custom Scroller v3.1.5
- Leaflet geometryutil (Leaflet plugin)
- Leaflet arrowhead (Leaflet plugin)
- Cropper.js
- Dropzone v6.0.0

Most of them, I use CDN to import to the project except Leaflet plugins and Cropper (download and integrated directly into the project, in static/js/lib)

Next, I'll explain the frontend implementation of pages that have sophisticated code structure in terms of frontend part, simple pages or backend-biased pages will be skipped in the next sections.

#### b. Side bar

Required data from back-end/database:

- All sites
- All campaigns

This data is used for listing all available sites/campaigns. The flow chart below explains how the side bar works when changing the campaign (in case a user logs in, otherwise, direct to login page).

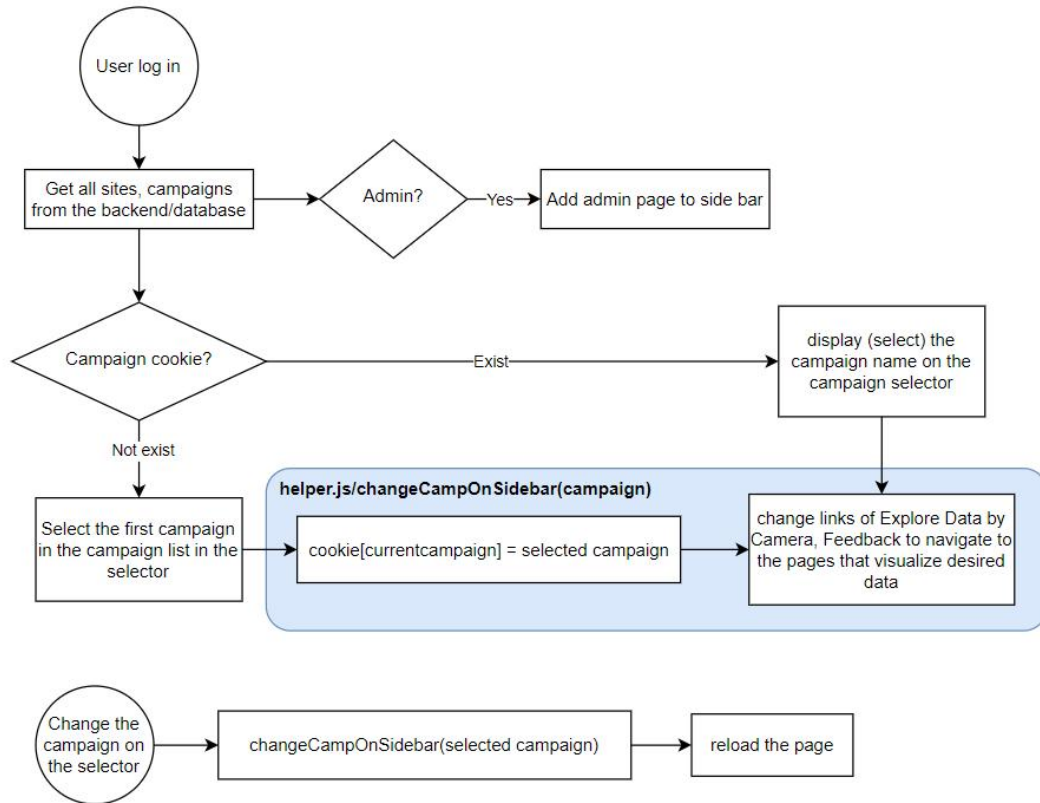


Figure V-1: Side bar flowchart

By default, the cookie will expire after 7 days or when users log out.

### c. Explore Data by Dog

URL: `area(site)/campaignName/algoId/ExploreDataByDog`

Required data from back-end/database:

- Cluster data: which STSD (video) belongs to which dog based on the algorithm (id = algoId)
- Avatar: representative STSD (medoid) of each cluster (dog)
- Camera: location data (of all cameras during the campaign)
- Stsd: all STSD
- pageData: (area, campaignName, algoId of the current page) indicates the current area, campaign name and algo id. Actually, this data can be retrieved directly by frontend not strictly got from backend.
- All sites
- All campaigns
- All algorithms, these 3 fields serve creating the list of available sites, campaigns and algorithms in the side and header bar.
- timezoneData: to display start time and end time of each STSD, the stored time in database is GMT +0, so we need this data to retrieve the proper time.

- staffStatus: check if a user is a staff member or not (in case guest user can access some campaign data) → allow a staff member to give feedback while guest user cannot.

The flowchart below shows how this page works (front-end):

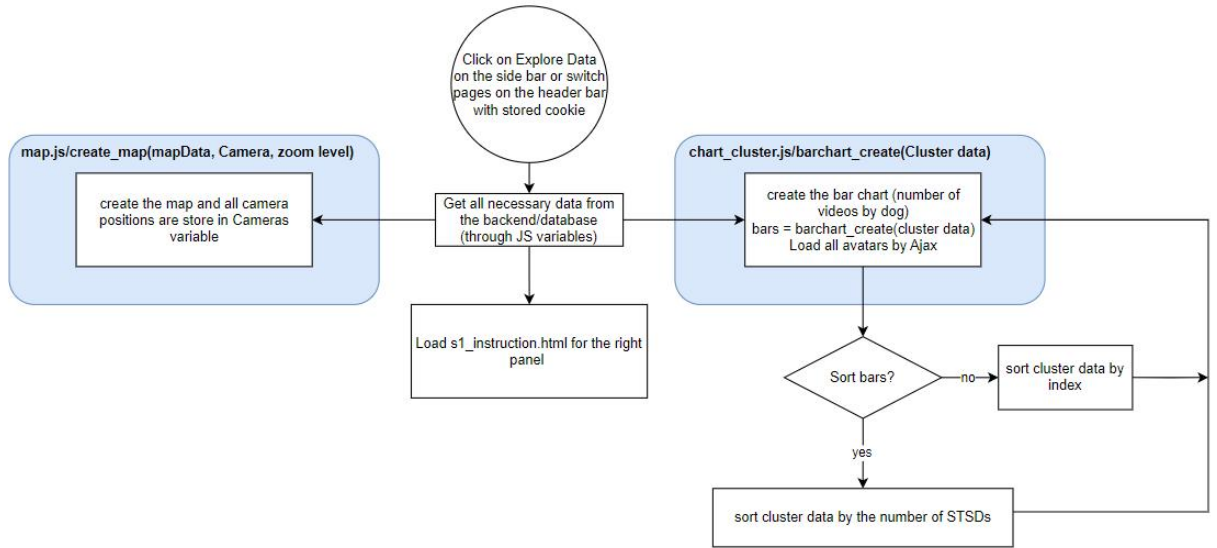


Figure V- 2: Initialize the Explore Data by Dog page

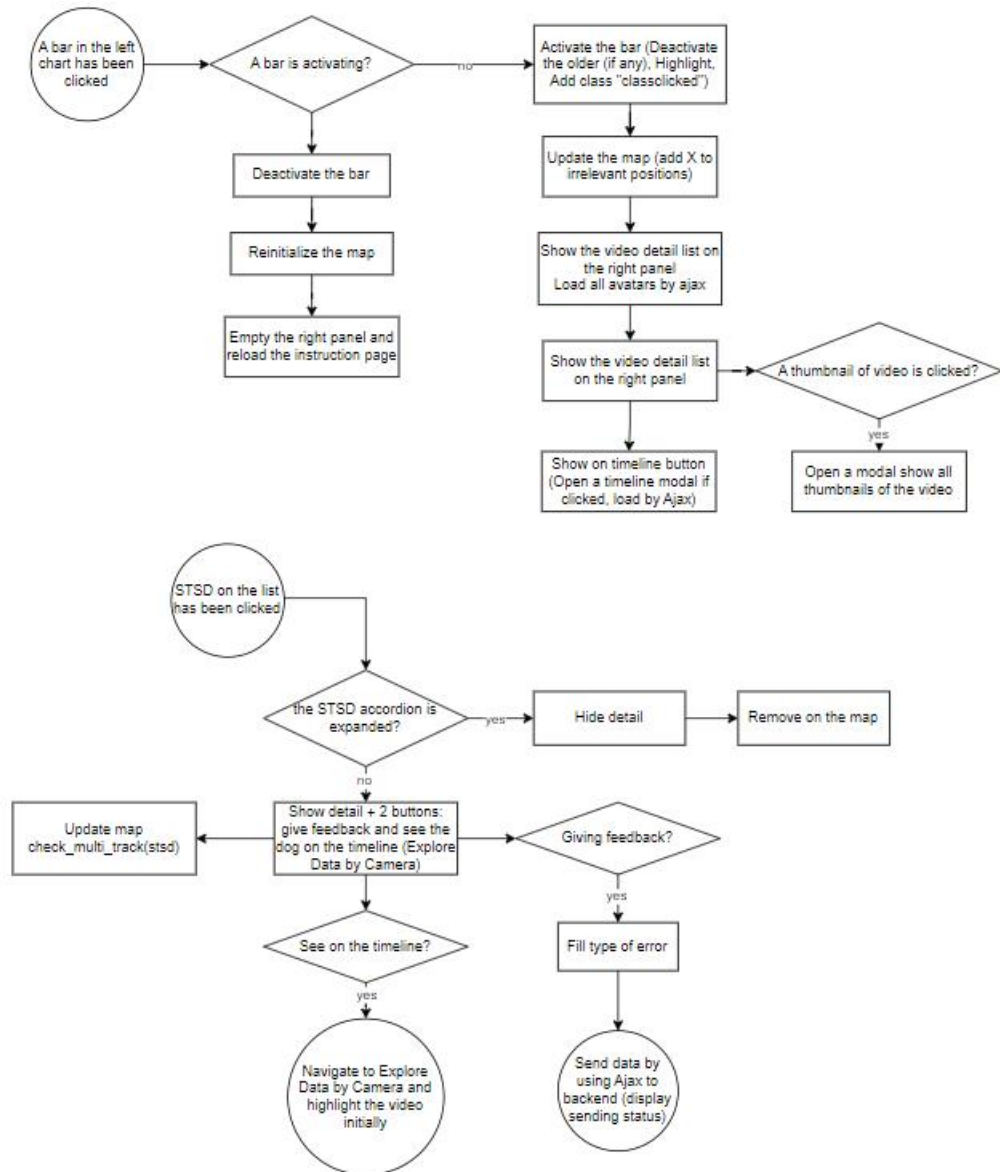


Figure V- 3: Explore Data by Dog page flowchart

This page also has another URL: [area\(site\)/campaignName/algId/ExploreDataByDog/dogId](area(site)/campaignName/algId/ExploreDataByDog/dogId)

This URL will head to the Explore Data by Dog page, but it will be initialized with the activation of the dogId bar (same as the result of clicking the dogId bar). I will explain this URL in more detail in the **Back-end** section.

#### d. Explore Data by Camera

URL: [area\(site\)/campaignName/algId/ExploreDataByCamera](area(site)/campaignName/algId/ExploreDataByCamera)

Required data from back-end/database is similar to Explore Data by Dog page.

The flowchart below shows how this page works (front-end):

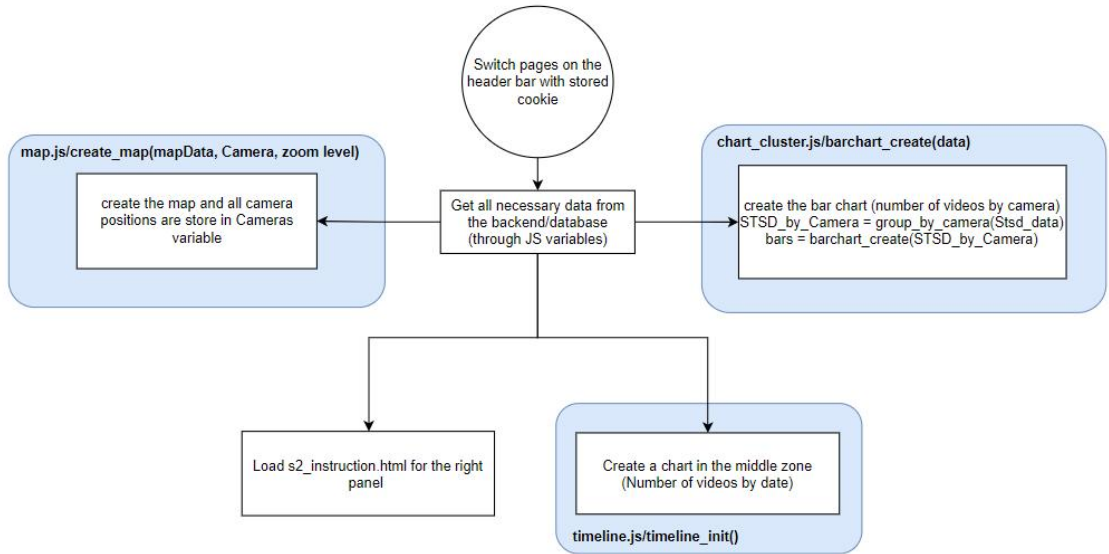


Figure V-4: Initialize the Explore Data by Dog page

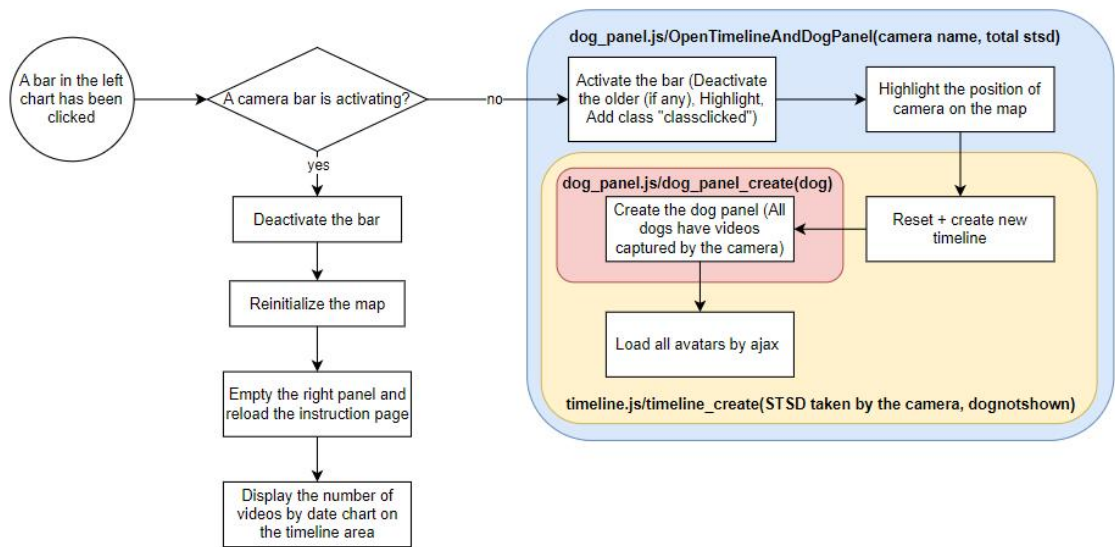


Figure V-5: Flowchart after a click on the camera bar chart

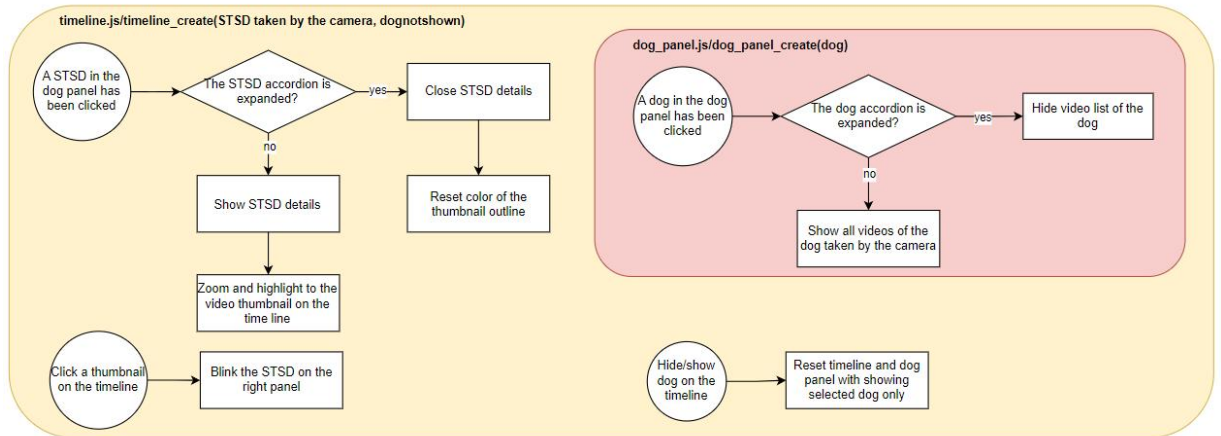


Figure V-6: Flowchart of further actions after a click on the camera bar

Similar to the Explore Data by Dog, this page also has another URL: [area\(site\)/campaignName/algoId/ExploreDataByCamera/dogId](area(site)/campaignName/algoId/ExploreDataByCamera/dogId)

It will be initialized with the activation of the camera bar (same as the result of clicking the camera bar).

### 3. Back-end

#### a. Overall

Among some cutting-edge backend frameworks, Django is chosen to be in charge of connecting the database and the frontend because of its detailed documentation and large community, especially, it's flexible to integrate ML models which are mostly written in Python into the website.

This framework follows MVC design pattern and ORM (Object relational mapping) is also applied in the framework. It means that we just need to create a database schema and the rest of the task like creating a class schema for OOP (Oriented object programming) is carried out by Django. Therefore, the class schema of this website is similar to the database schema (Figure I-1).

Besides Django, I use several other libraries/frameworks to support ML integration or Websocket. Below is their version:

- Backend + Database:
  - Django v3.2.18
  - Daphne
  - Pymysql v1.0.2
  - Requests v2.25.0
  - Channels v4.0.0
- ML:
  - Scikit-learn v0.24.2

- Pandas v1.1.5
- Numba v0.53.1
- Hausdorff v0.2.6
- Matplotlib v3.5.3
- Opencv-python v4.5.1.48
- Cryptography v38.0.4
- Pillow v9.1.1
- Pytorch lightning v1.6.3
- Torch v1.11.0
- Torchmetrics v0.8.2
- Torchvision v0.12.0

Note that the Python version on the server to operate the website is 3.7.

b. Routing controller (HTTPS)

All URLs are detailed in pages/urls.py. The figure below summarizes the routing controller:

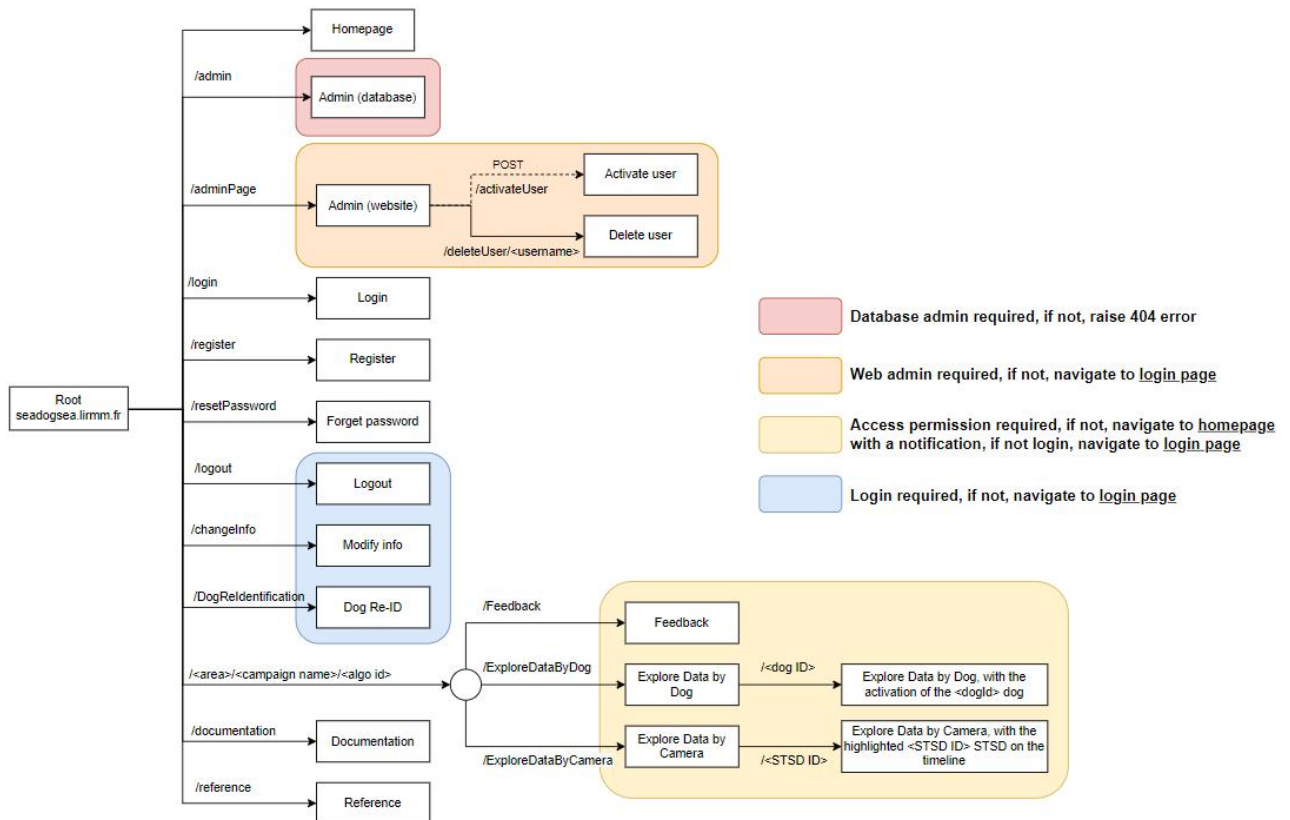
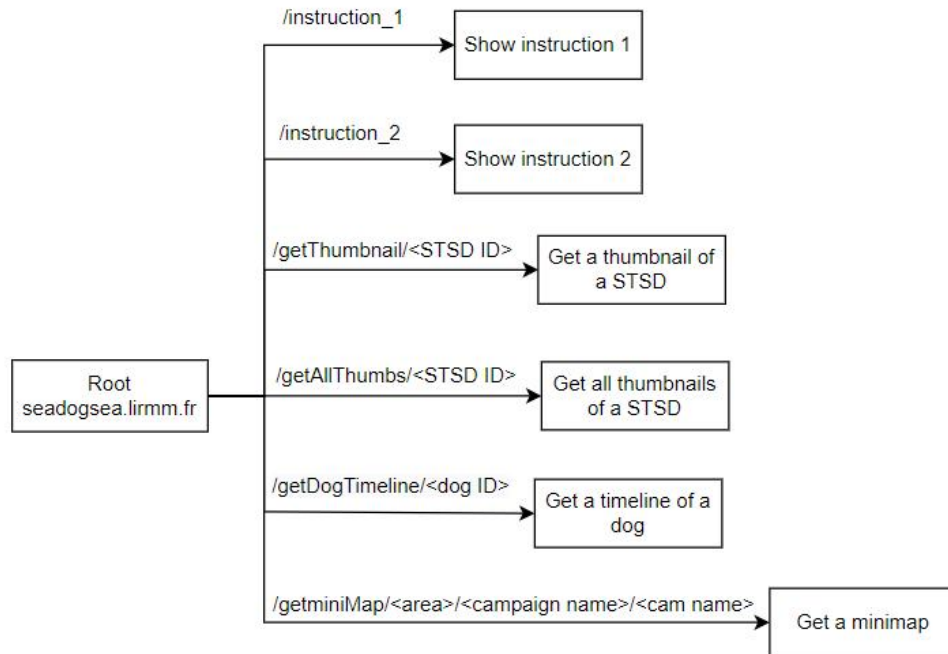


Figure V-7: URL routing of the website



For some pages that require specific permissions to be able to access, I use decorators `@login_required`, `@user_passes_test` (provided by Django) or hard code to redirect users to other pages the users can access such as login page or homepage with notifications.

Besides the URLs mentioned in the figure V-7, there are other URLs that are called implicitly by Ajax to display on main pages without refreshing the pages.



*Figure V-8: Implicit URLs*

For now, I only set a login required constraint for these pages, that means that any user after logging in will be able to access to see data by a URL request, for example, a user types `seadogsea.lirmm.fr/getAllThumbs/1`, then, he can see all thumbnails of the STSD ID 1. Despite the data leak, it's not straightforward to find these URLs because they are only called implicitly by Ajax. For better data protection, it's essential to implement functions that check user permissions to prevent users from requesting this data through URL. Note that HTTPS must be used in production.

c. Routing controller (WSS)

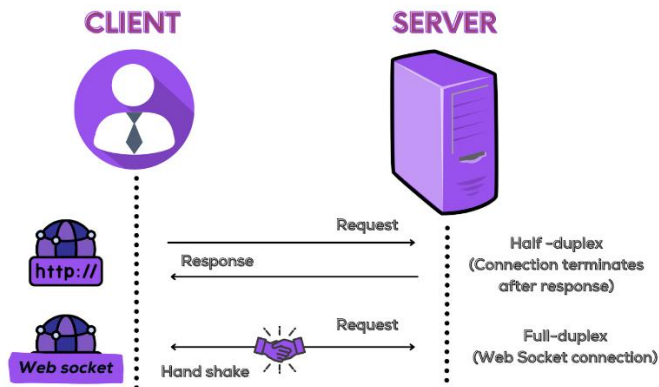


Figure V-9: the difference between HTTP and websocket (WS)

Unlike HTTP, Websocket will establish and remain a connection with the server after a request of client to exchange data continuously. I leverage this feature in the Re-Id page, allowing to track the progress of feature extraction and prediction.

To do that, it is required to implement a consumer to fetch data from client side, process, and response back to the front-end. I detail it in the **Re-ID page** section.

URL: `wss//seadogsea.lirmm.fr/ws/reIdprocess/`, `wss` will be replaced by `ws` in development mode.

#### d. Django settings

It can be found in root → `dog_project` → `settings.py`

The setting file in the repository is different from the one on the server in these points:

- `DEBUG = True` (for development) and `False` (for production), when this variable is `True`, it allows developers to debug while programming.
- `ALLOWED_HOSTS = []` (for development) and `['seadogsea.lirmm.fr']` (for production)

Some information such as database password, email, email password (for sending notifications to administrators) will be hidden in **credentials.py**. To develop the website on your computer, you need to create this file containing all information to connect to your local database or to assure the email functionalities of the website works properly.

Compared to the initial setting (the setting when you have created a project by Django), I add some other configurations:

- In `INSTALLED_APPS`:
  - `Daphne`
  - `Channels` (with `daphne` to enable websocket)
  - `Pages` (the main application, Django is the powerful framework so you can develop many applications in only 1 project)

- In TEMPLATES, add templates to DIRS (the directory stores all html files)
- Enable ASGI (asynchronous programming), it allows the website to handle multiple requests at once, for now, it only supports websocket and if in the future, the website scales and serve more and more users, this feature will be valuable to process user requests asynchronously, taking less time to response.
- Add the directory in which static files are stores (js, css, images) to STATICFILES\_DIRS.
- Add email configurations.
- Set SESSION\_EXPIRE\_AT\_BROWSER\_CLOSE to automatically delete user session variables (log out).

#### e. Homepage

Data sent to front-end:

- areaData: name of all distinct sites in Site table
- campaignData: all campaigns in Campaign table

This data will be used on almost all pages which have the side bar (to choose the desired campaign to work on).

This page leverages Django template mostly for the front-end instead of JS.

Django template: a template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted, more details: <https://docs.djangoproject.com/en/4.2/topics/templates/>

#### f. Login page

This page has a form containing 2 fields: username and password. POST with validation methods in the front-end (such as check empty field before the submission) is used to send the form to server side. Note that, in the client side, it is mandatory to embed csrf token into the delivered form to prevent the Cross-Site Request Forgery.

The figure below is the flowchart for login implementation:

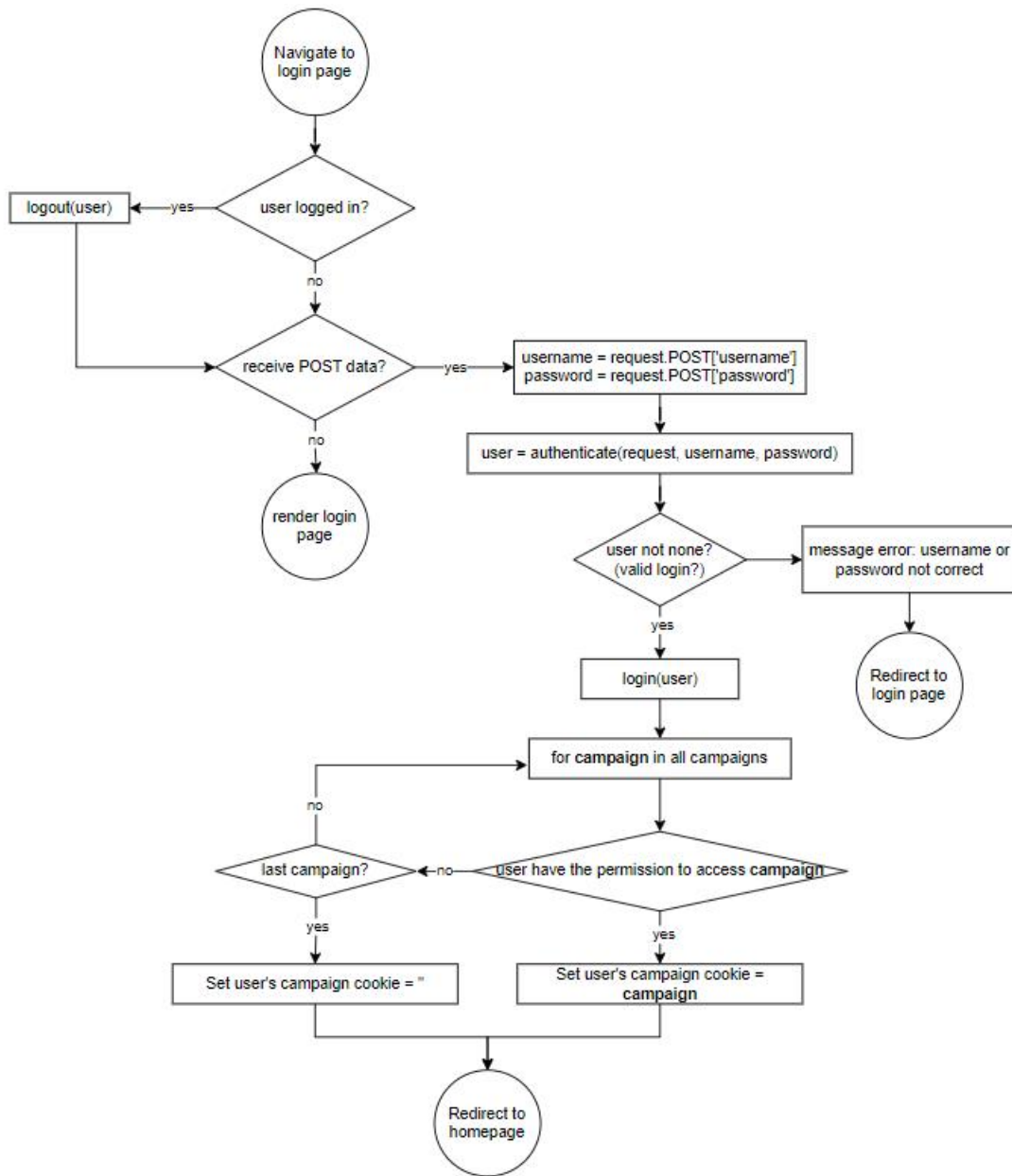


Figure V-10: Flowchart of login page in server side

In the implementation, there are some built-in functions that are provided by Django to avoid hard code programming:

- `logout(request)`: it will log user out if a user is logging in by scanning session variables.
- `authenticate(request, username, password)`: verify password based on username in the database. If the password is valid, the function will return the user object, otherwise, return none.
- `login(request)`: add user information to session variables, the user is now logging in successfully.

This page leverages Django template mostly instead of JS.

g. Register and Reset password page

Like the login page, these pages also use the POST method to send data form from client side to server side. The figures below are the flowcharts showing how they work:

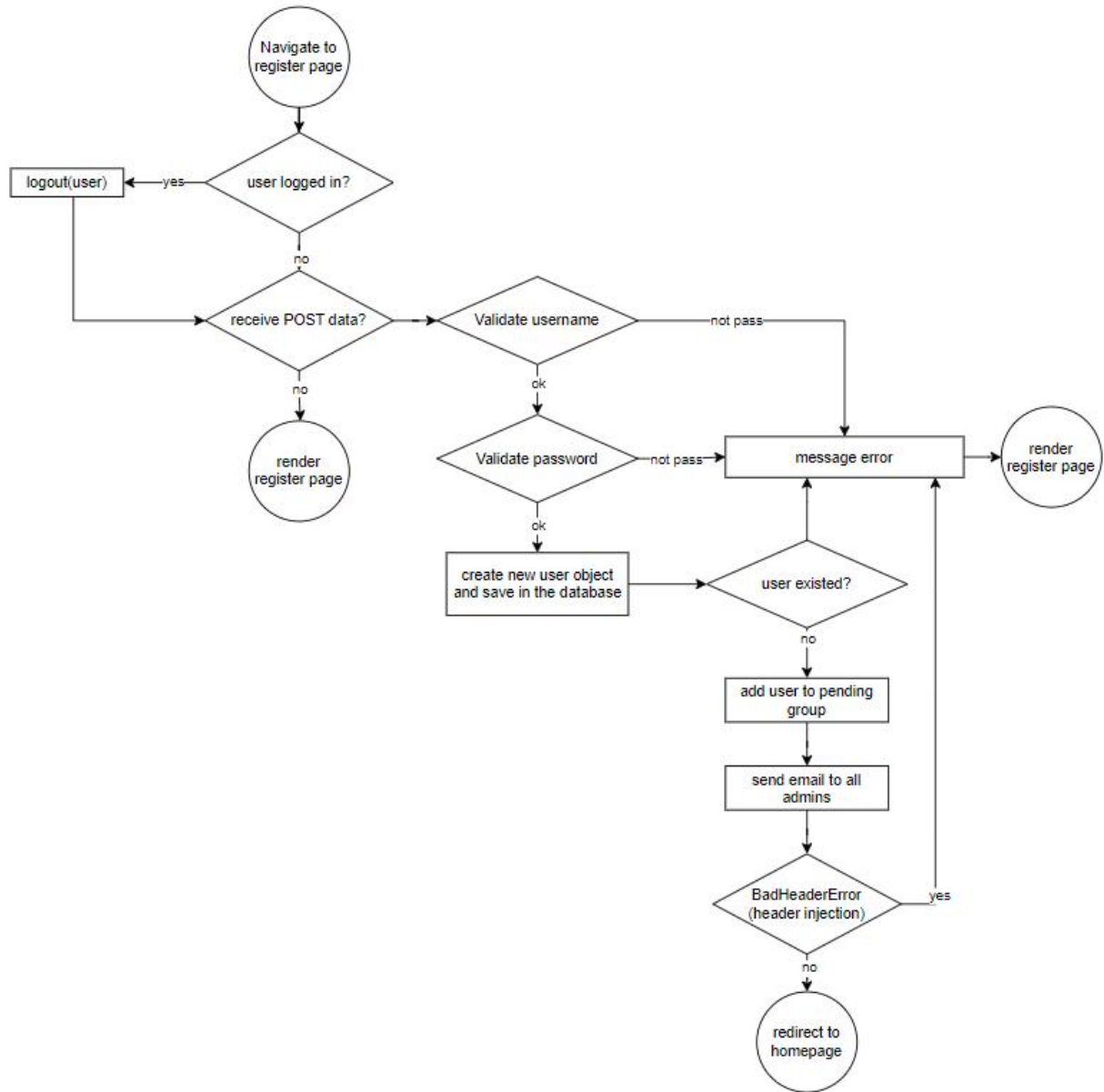


Figure V-11: Flowchart of register page in server side

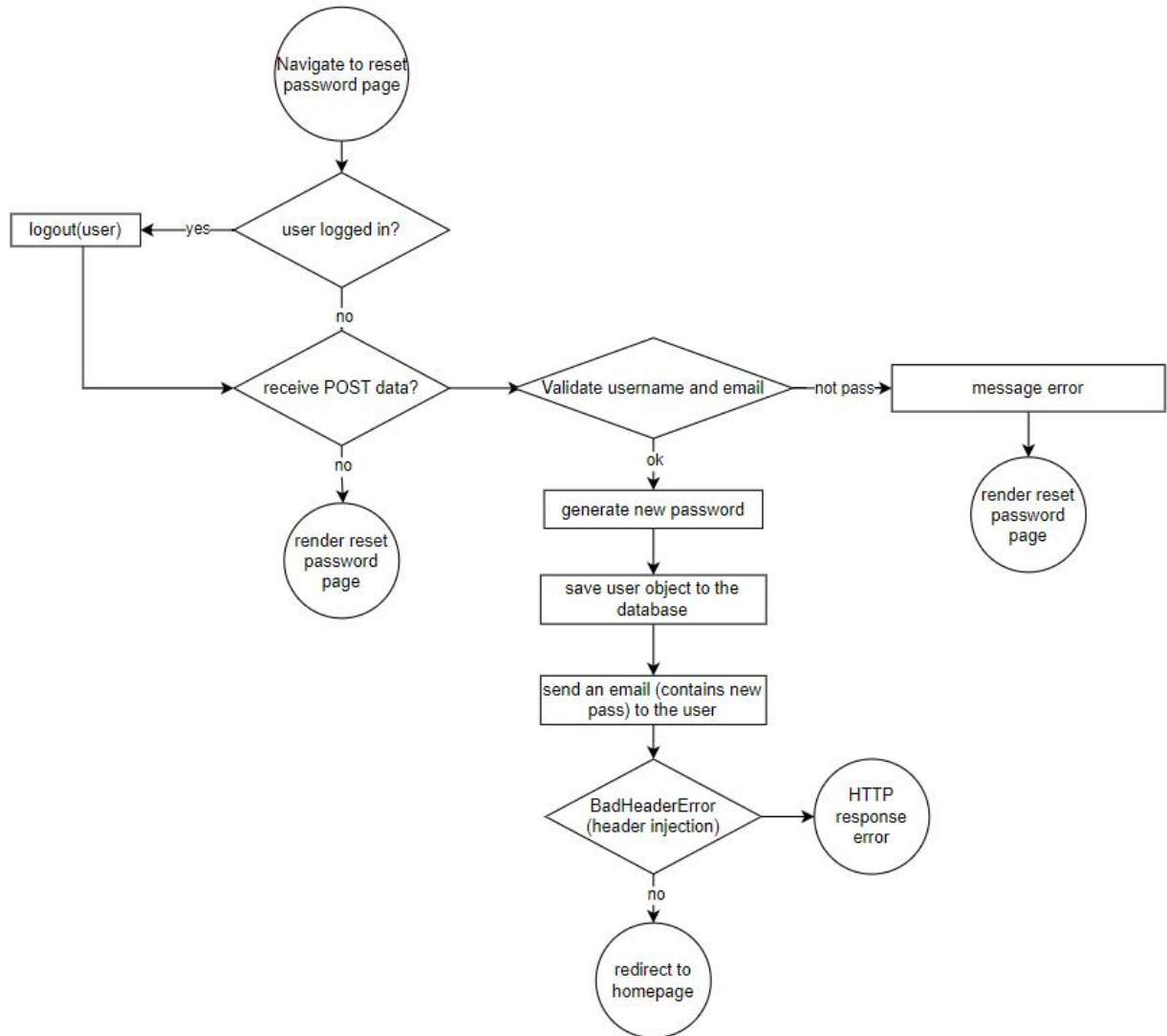


Figure V- 12: Flowchart of reset password page in server side

Built-in functions:

- `validate_password(password)`: it will verify the password to assure that the password meets the requirements (not a part of other personal information, at least 8 characters, not a commonly used password, not entirely numeric)
- `send_email(subject, message, from_email, recipient list,...)`: send an email to designated people in recipient list by using the setup email in settings.py. This function will raise an exception `BadHeaderError` to prevent the header injection attack.

These pages leverage Django template mostly instead of JS.

h. Modify user personal information page

After a user log in, he can modify his account such as change name, email, or password.

There are 2 separate pages allowing the user to commit the actions: modify user information and change password page.

Like login, register or reset page, these pages consist of the form to acquire changed information from user, the new data will be updated in the database afterwards if it is valid. For the change password page, there would be some steps that required a user to verify his old password first to be able to update his new password.

i. Admin page (for user management)

This page helps website admins manage their users like activate, modify information, or assign users to specific groups or even delete users.

To be able to access this page, a user must be a website admin. I implement the function (`adminCheck(user)`) that checks permissions of user, it will return True if the user belongs to Database admin group or Web admin group or if the user has the permission Can Access Admin.

Then, the decorator `user_passes_test` is applied to approve the access request or direct a user to login page if he does not have the valid permission.

This page will display all users in the database; therefore, it is vital for the back-end to send some data to the front-end:

- `areaData`: name of all distinct sites in Site table
- `campaignData`: all campaigns in Campaign table (as usual, pages that have the side bar requires `areaData` and `campaignData`)
- `users`: all user in User table
- `groups`: all groups in Group table
- `userActiveCount`: the number of active users
- `userPendingCount`: the number of pending users

For demonstrating how this page works, 4 flowcharts will be provided in the figures below because there is a solid link between the functionalities of back and front-end and other pages like `activateUser` and `deleteUser` (note that `activate` and `deleteUser` page do not have a html template, only back-end implementation)

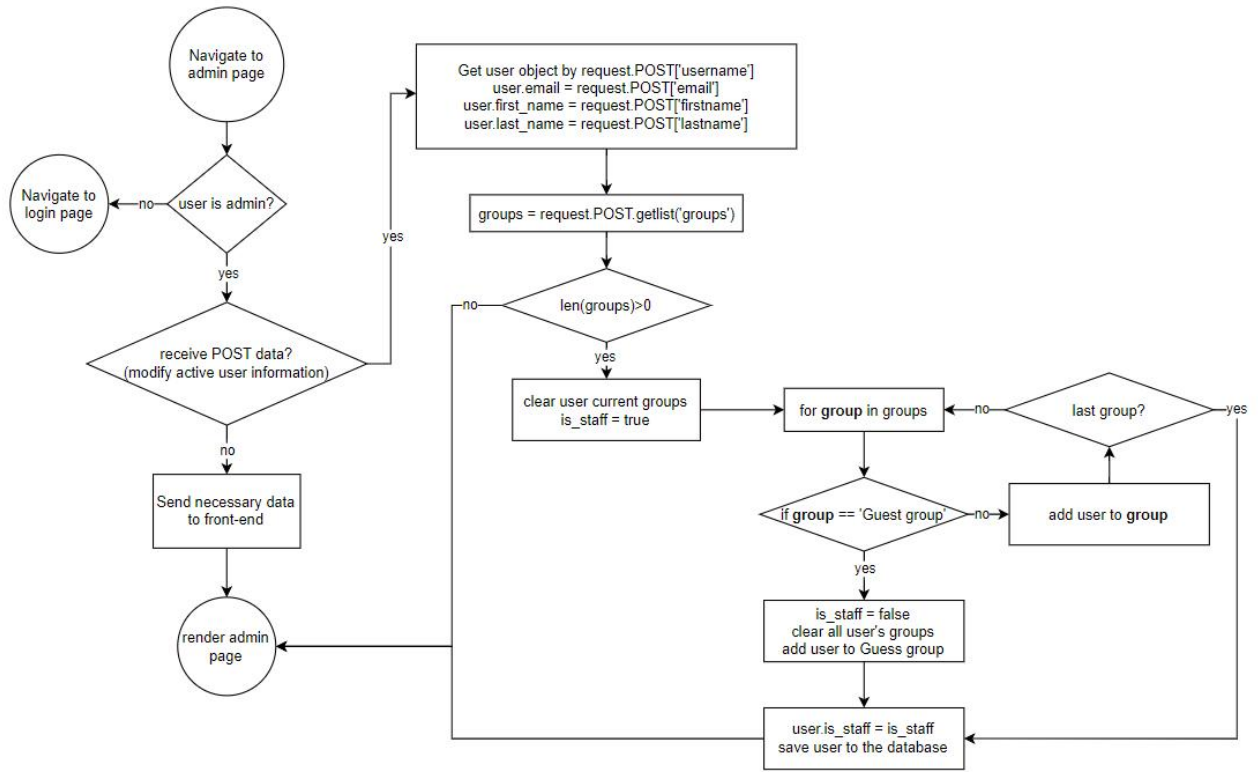


Figure V-13: Flowchart of admin page in server side



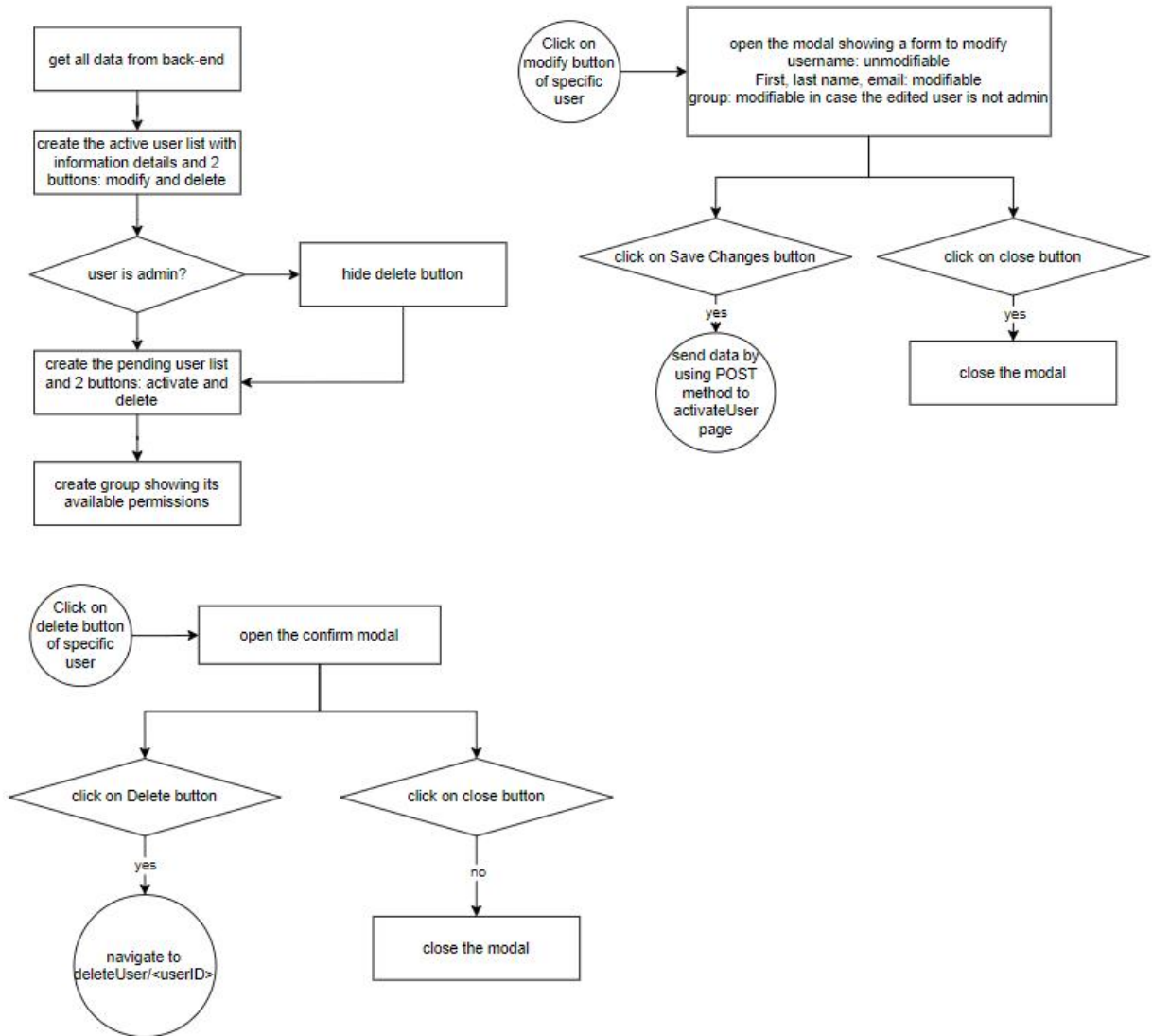


Figure V-14: Flowchart of rendering admin Page

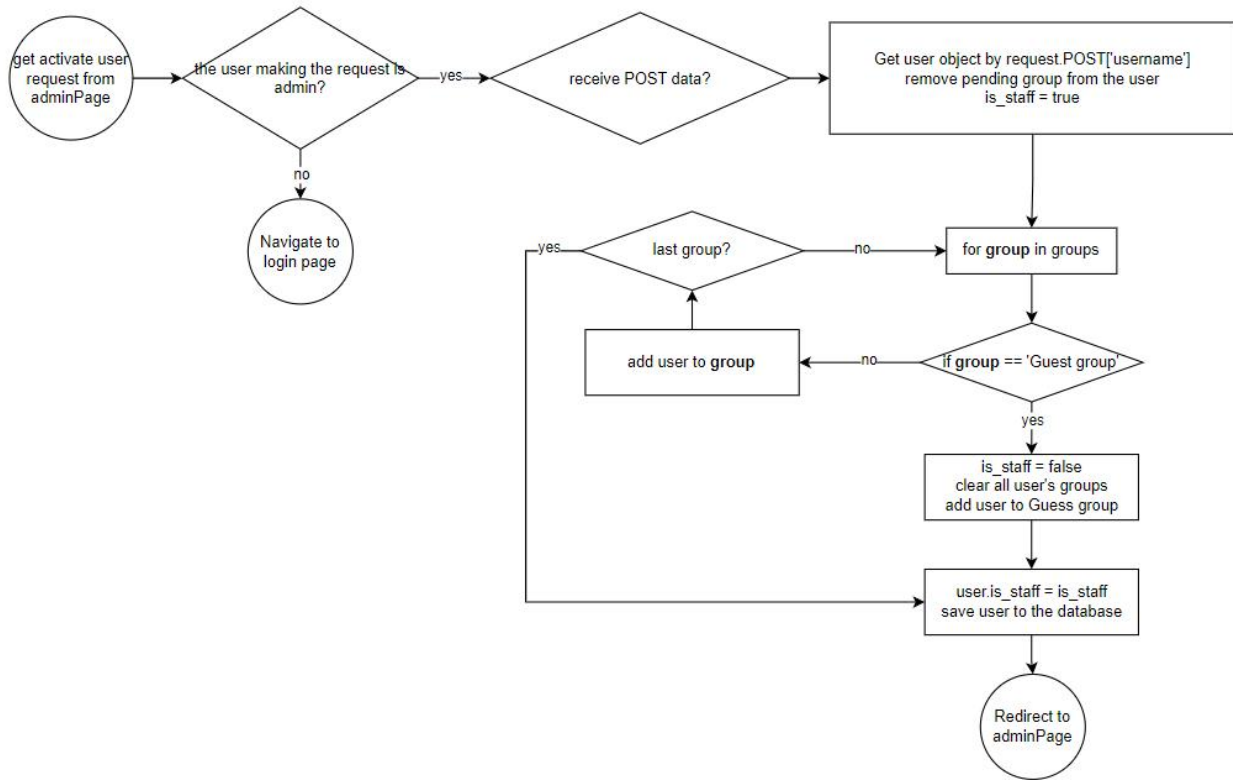


Figure V-15: Flowchart of activating a user

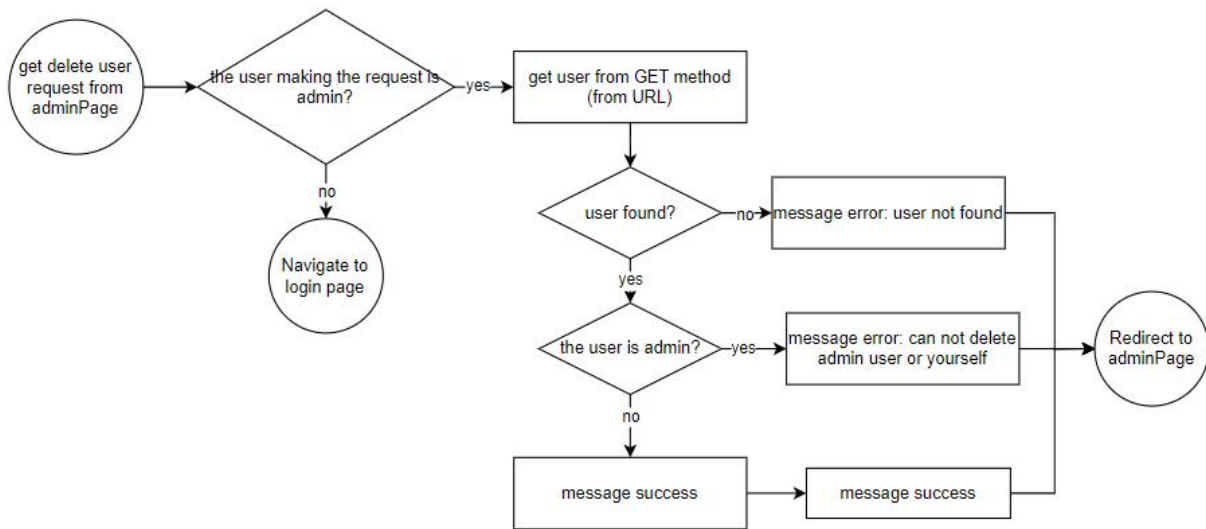


Figure V-16: Flowchart of deleting a user

#### j. Explore Data Pages

Unlike other pages, these pages are programmed mostly by JS, namely D3.js, to create interactive charts on the front-end, making it more intuitive with moderate efforts. But it is a bit

sophisticated to transfer data from back-end to JS so the data can be leveraged to visualize on the interface.

To begin with the idea, firstly we need to retrieve all relevant data from the database. I explain why using JS is more challenging than Django templates by considering this example:

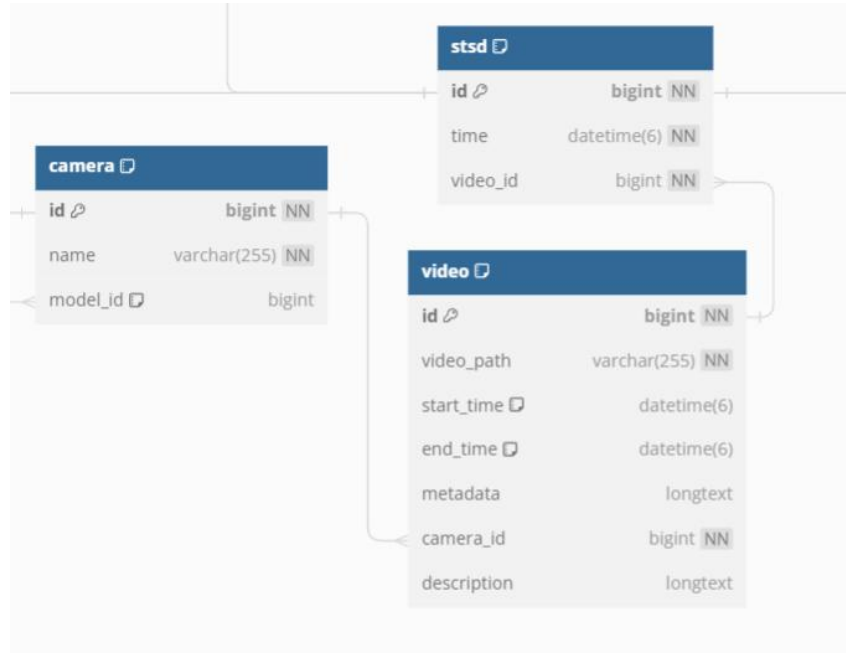


Figure V-17: Example of relational tables

I have 3 relational tables like the figure above, indicating the relation between STSD, video and camera. Then, I'd like to retrieve which camera took the STSD ID 1 in the front-end. For Django template, the first thing is preparing data from the back-end: `StsdData = Stsd.objects.all()`, in the front-end side, to retrieve the desired data we only need: `{% StsdData.0.video.camera.name %}`. JavaScript, in contrast, can not retrieve the data directly like that, it only allows to read JSON-like data. Thus, we need to serialize data before transferring to the front-end. Plus, when serializing the data, we can not receive the full relational objects. For instance, the content of serialized stsd objects is `id`, `time`, and `video_id`, not video objects like the data for Django template. Therefore, we have to transfer data separately from different tables but to avoid the unnecessary data, we can choose only the desired fields (not entire table) then add it to the serialized data. To make it clearer, I assume that we need to get `start_time` and `end_time` of STSD, we just need to add `set_related('video')` to our query, but it's required one more step: add natural key into the video model, this key contains `time` information (more information: <https://docs.djangoproject.com/en/4.2/topics/serialization/>). By doing that, the `video_id` in STSD will be replaced by `start_time` and `end_time`.

There are 2 main visualization pages: Explore Data by Dog and Explore Data by Camera. The pages require the same data from back-end (locate in `getDataforExplorePages` function)

based on campaign and algorithm (this information is obtained by method GET), I've already mentioned in **Explore Data by Dog** (section V.2.c)

The only difference between these pages is that in Explore Data by Dog, it has a functionality that permits a user to give feedback.

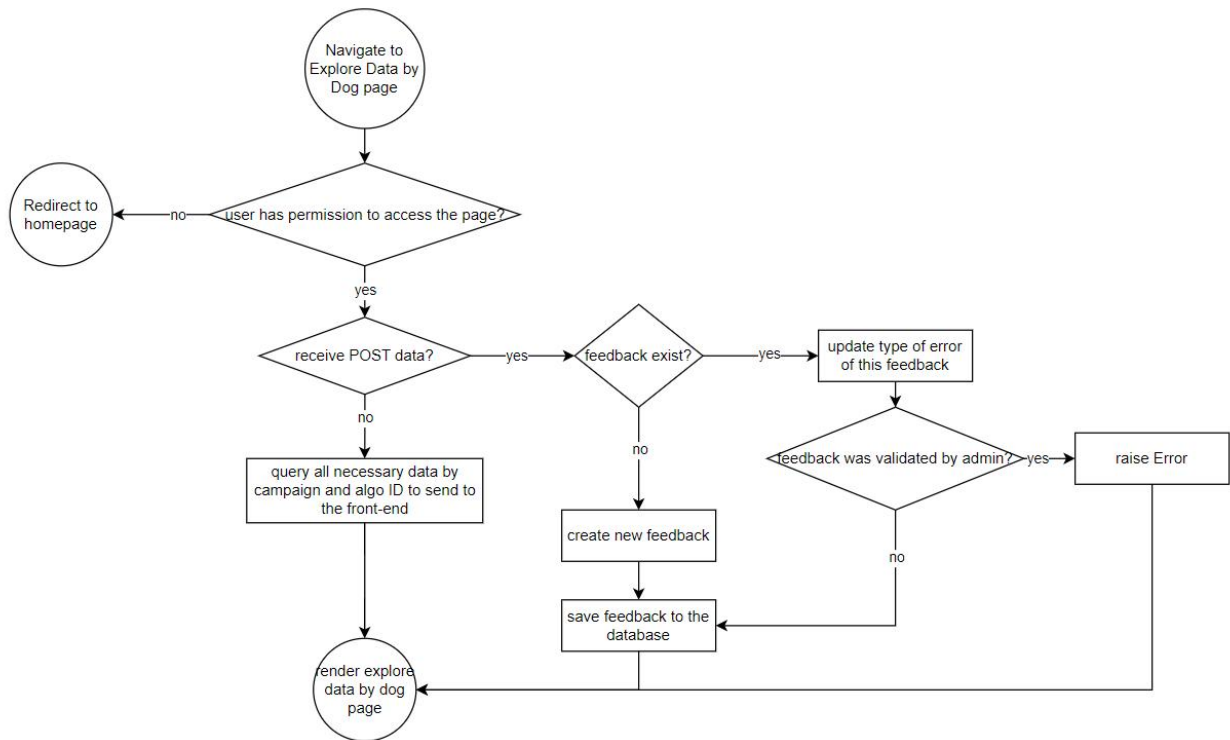


Figure V-18: Flowchart of Explore Data by Dog page in the server side

Note that the POST data is delivered to the back-end through Ajax, so the page won't be refreshed, allowing the user to continue interacting with it. When the back-end raises an Error, Ajax is designed to catch the error, so we just display the error on the front-end if there is something wrong in while the back-end is processing data.

In the code, you will find StsdDogLog, it means feedback which is identified by user id and StsdDog id (StsdDog table store the link between dog and stsd showing which stsd belongs to which dog and dog also means cluster).

#### k. Feedback page

This page requires the same data as data required in Explore Data pages.

The mechanism of this page is quite simple: the Django template (front-end) will check if a user is admin or not, if yes, the user will be able to accept, reject or delete feedback. The template also renders all feedback in the pending area, the accepted feedback and rejected feedback as well. When an admin evaluates feedback, for example, accept it, then the feedback in pending area will be hidden and the one in the accepted area will be shown simultaneously.

When the feedback is evaluated, a POST form is sent to the back-end to update the feedback in the database.

### 1. Dog Re-Identification page

Unlike other pages using the POST method to send data to the server side, this page applies websocket technique to fetch data from the front-end, process and return the result of ML model back to the front-end.

The communication between front-end and back-end will be illustrated on the figure below:

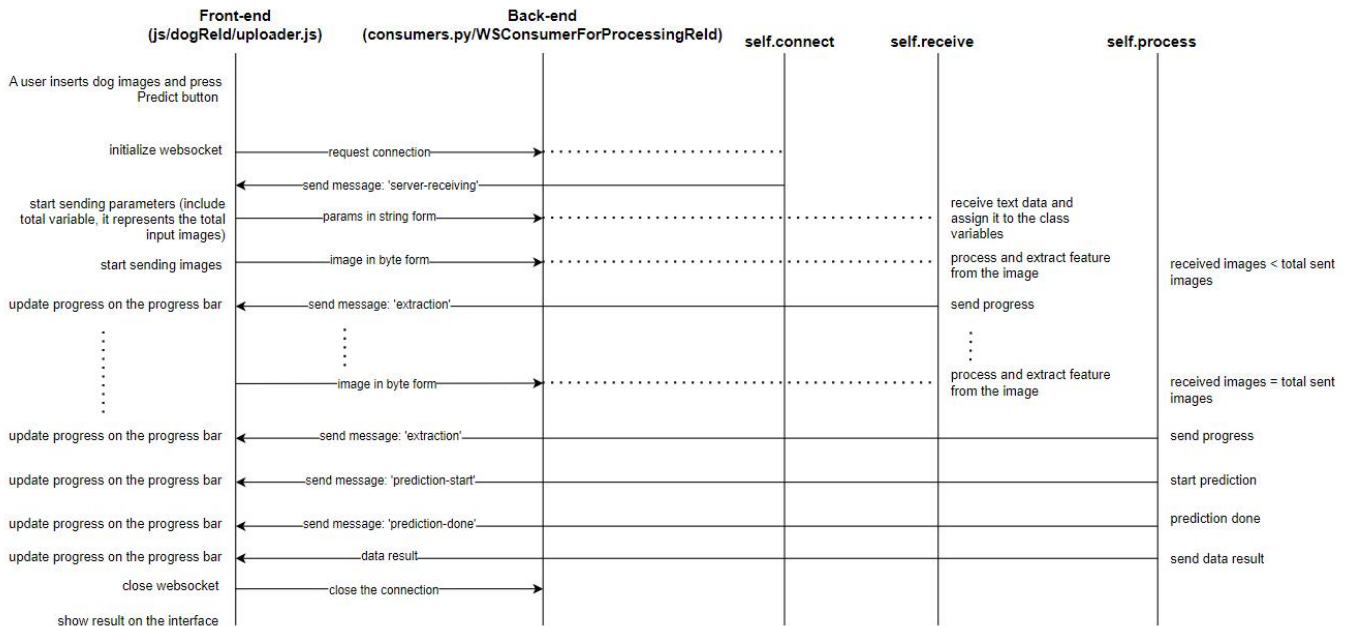


Figure V-19: Flowchart of communication between front-end and back-end through websocket in Dog Re-Identification page

## 4. Database

Database: MySql >= 8.0

As I explained in the section I (**Database management guidelines**), the database consists of 2 main parts, one for user data and another for project data (STSD, video, camera, ...).

There are still some empty columns (like detection score in thumbnail table, or meta data) and some empty tables (such as ownership, owner...). In the future, these tables will be helpful for further advanced features.

Some tables may seem confused because of their name:

- Dog table: this table is to storing clusters along with the table StsdDog. Note that although the cluster results from different algorithms may be identical, they have

to be stored in different clusters in the database. The figure below shows how STSDs are stored when they are clustered by different algorithms:



*Figure V- 20: Example of storing STSD and clustering results*

As we can see, there would be 5 dogs stored in the database by using 2 different algorithms. By doing this, we can leverage all information without any data loss.

- **StdDogLog**: identified by user id and StdDog id. The data in this table represents user feedback. One user can only give one feedback on STSD in a specific cluster (result of specific algorithm).

I recommend that user data should be monitored/modified (if any) in [seadogsea.lirmm.fr/adminPage](http://seadogsea.lirmm.fr/adminPage), not directly in [seadogsea.lirmm.fr/admin](http://seadogsea.lirmm.fr/admin) except some cases of force majeure.

Other data can be accessed in [seadogsea.lirmm.fr/admin](http://seadogsea.lirmm.fr/admin) but remember to avoid deleting data as much as possible.

## **VI. Limit and improvements**

The drag and drop feature does not work on Firefox.

For the moment, the website only uses 1 server on 1 machine and is not distributed yet, if in the future, the project gets more scale or more users, it's better to apply distributed system with asynchronous techniques to assure the stability of the whole system.

New proposed features in Re-Identification page:

- The current page is just a demo. For production, it would be upgraded with more functionalities
- Add dog image with its owner in the sample area
- Feedbacks given by users can be tested on this page
- Merge the campaign parameter to the campaign selector in the sidebar. So, user can only interact with the data in the selected campaign (which the user is authorized to access) on this page

## REFERENCES

1. Django: <https://docs.djangoproject.com/en/4.2/>
2. D3js: <https://d3js.org/>
3. Leaflet: <https://leafletjs.com/>
4. JS: <https://www.w3schools.com/js/>
5. Dog re-identification from videos in non-controlled environment: Eugênio DIAS RIBEIRO NETO, Cyril BARRELET, Marc CHAUMONT(Senior, IEEE), Gérard SUBSOL, Barandi Sapta WIDARTONO, Najib Arung PETANA, Wayan Tunas ARTAMA, Sopheak SORN, Sowath LY, Etienne LOIRE, Michel DE GARINE-WICHATITSKY