Backend Development

To start the backend, run the following commands on the AWS dev instance. The database would be AWS RDS

```
#activate virtual env
source venv-pv-val-hub/bin/activate
cd pv-validation-hub/valhub/

#update database schema
python3 manage.py makemigrations
python3 manage.py migrate

#start server
python3 manage.py runserver 0.0.0.0:8080
```

To run locally, create a virtual environment and install all required packages. The database would be a local sqlite database.

```
#update database schema
python3 manage.py makemigrations
python3 manage.py migrate

#start server
python3 manage.py runserver 8080
```

Reset database

1. remove migrations
   delete all migration files except __init__.py in jobs/migrations, analyses/migrations, accounts/migrations
2. delete and re-create database

```
#in MySQL command line
drop database test;
create database test;
```

3. migrate

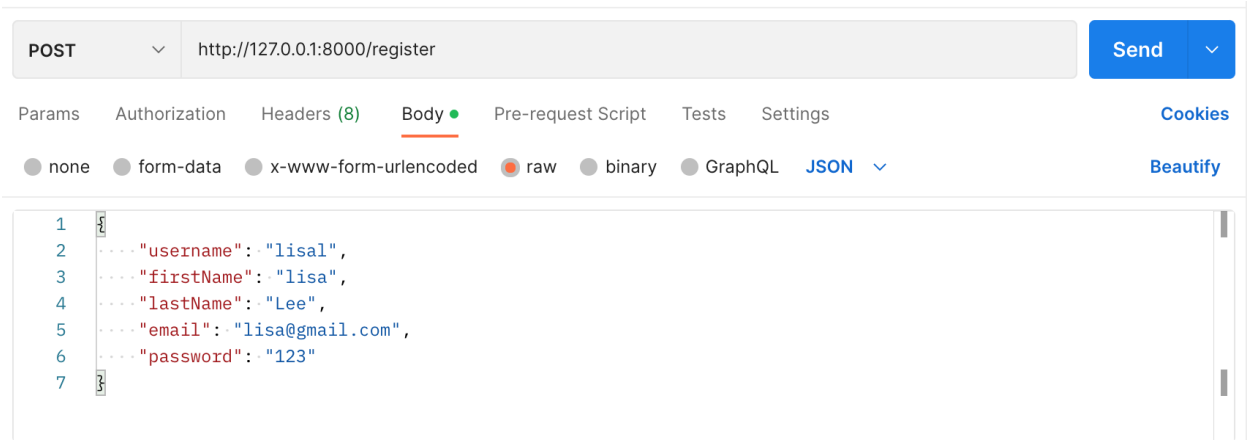
```
#update database schema
python3 manage.py makemigrations accounts
python3 manage.py makemigrations analyses
python3 manage.py makemigrations jobs
python3 manage.py migrate
```

User Account:

POST: /register

Input:

1. username (string)
2. email (string)
3. password (string)
4. firstName (string)
5. lastName (string)
6. githubLink(string, optional)



Output:

1. id(uuid)
2. username(string)
3. email (string)
4. password (string)
5. firstName (string)
6. lastName (string)
7. githubLink(string, optional)

Body Cookies Headers (10) Test Results 200 OK 30 ms 500 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "id": "71a66690-07be-4dfe-a2b4-48670ffc32fe",
    "username": "lisal",
    "password": "123",
    "firstName": "lisa",
    "lastName": "Lee",
    "email": "lisa@gmail.com",
    "githubLink": null
}
```


GET: /login

Input:

1. username (string)
2. password (string)

GET http://127.0.0.1:8000/login Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
    "username": "test",
    "password": "123"
}
```


Output:

1. Id (uuid)

Body Cookies Headers (10) Test Results 200 OK 64 ms 371 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "id": "287d3be1-6bff-4fb5-804f-1d0b6b45895c"
}
```


GET: /account/list

Input: None
Output: A list of all account objects

example:

[{"id": "46417394-c0ef-4184-90a3-294c07e6d063", "username": "lisal", "password": "123", "firstName": "lisa", "lastName": "Lee", "email": "lisa@gmail.com", "githubLink": null}, {"id": "074b2f11-7222-4700-bdc6-1a38d6eec209", "username": "lisax", "password": "123", "firstName": "lisa", "lastName": "Lee", "email": "lisa1@gmail.com", "githubLink": null}, {"id": "71a66690-07be-4dfe-a2b4-48670ffc32fe", "username": "lisal", "password": "123", "firstName": "lisa", "lastName": "Lee", "email": "lisa@gmail.com", "githubLink": null}, {"id": "287d3be1-6bff-4fb5-804f-1d0b6b45895c", "username": "test", "password": "123", "firstName": "test", "lastName": "test", "email": "test@gmail.com", "githubLink": null}]

GET/POST/DELETE: /account/<user_uuid>/

Input:

1. Username
2. Password
3. Any other data that needs to be updated(for put)

Output:

1. Updated Account object

Analysis Creation

/analysis/upload

Input:

1. user_id (int)
2. analysis_name (string)
3. description (string)
4. short_description (string)
5. ruleset (string)
6. dataset_description (string)
7. evaluation_script (.zip file; [example])

Output:

1. a json string containing an analysis object


```
"[{\"model\": \"analyses.analysis\", \"pk\": 1953719668, \"fields\": {\"analysis_name\": \"test\", \"description\": \"test\", \"evaluation_script\": \"https://pv-insight-application-bucket.s3.us-east-2.amazonaws.com/evaluation_scripts/analysis_1953719668.zip\", \"annotation_file_name\": \"test_annotation.txt\", \"max_concurrent_submission_evaluation\": 100}}]"
```


Implementation

1. get user Account object based on user_id
2. save the Analysis object which would generate a unique analysis id automatically
3. upload .zip file to S3 bucket (anaylysis_<analysis_id>.zip)

4. spin up an EC2 instance (worker) with analysis_pk in the tag, which would create an SQS queue and listen to it

Reference
https://github.com/Cloud-CV/EvalAI/blob/09202ad46e31bd721c3cbb0c2fa31e740cd6d19e/apps/challenges/views.py#L860

Debugging

1. Django file uploading
   uploaded files are in request.FILES and would be handled by serializer. Serializer needs to be validated first and then saved (a model instance would be saved if a model serializer is used)

   serializer: create, update, and save
   https://stackoverflow.com/questions/45100515/what-is-the-different-between-save-create-and-update-in-django-rest-fram

   Django MEDIA_ROOT and MEDIA_URL
   https://stackoverflow.com/questions/4820122/django-media-root-and-media-url

2. AWS credentials
   AWS credentials are automatically configured when an IAM is attached to the instance
   https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-metadata.html
3. user data log
   /var/log/cloud-init-output.log
   https://intellipaat.com/community/7865/how-to-check-whether-my-user-data-passing-to-ec2-instance-working-or-not

   note that user data is executed by root and home directory is /root
4.

Analysis Listing

/analysis/home

Input
Output: a json string containing a list of objects (analysis object)

Analysis Detail

/analysis/<analysis_id>

Input
Output: a json string of an analysis object

Submission Creation

/jobs/analysis/<analysis_id>/submission

Input:
1. analysis_id (int)
2. user_id (int)
3. algorithm (.zip file; example)

Output:
1. a json string containing a list of objects (submission object)


```
"[{\"model\": \"jobs.submission\", \"pk\": \"1cd7e0ea-884d-44ce-9e96-5f186bf99904\",
\"fields\": {\"analysis\": 1, \"algorithm\":
\"https://pv-insight-application-bucket.s3.us-east-2.amazonaws.com/submission_files/submissi
on_1cd7e0ea-884d-44ce-9e96-5f186bf99904.zip\"}}]"
```


Implementation
1. check if the analysis exists or not
2. check if the analysis queue exists or not
3. get user Account object based on user_id
4. save the Submission object which would generate a unique submission_id automatically
5. upload .zip file to S3 bucket
6. send a message to the analysis SQS queue

Reference
https://github.com/Cloud-CV/EvalAI/blob/09202ad46e31bd721c3cbb0c2fa31e740cd6d19e/apps/jobs/views.py#L133

User's Submission

/jobs/user_submission/<user_id>

Input:
1. user_id

Output:

1. a json string containing a list of objects (submission object)

/jobs/analysis/<analysis_id>/user_submission/<user_id>

Input:

1. analysis_id
2. user_id

Output:

1. a json string containing a list of objects (submission object)

Submission Detail

/jobs/<submission_id>

Input:

1. submission_id

Output:

1. json string containing submission object

Leaderboard

/jobs/<analysis_id>/leaderboard

Input:

1. Analysis_id

Output:

1. a json string containing a list of objects (submission object)