Backend Development Doc

Version 1.0.1

* Backend API update
  + There should not be any uuid as parameter to backend server

|  |  |  |  |
| --- | --- | --- | --- |
|  | description | bug | fix |
| /account/ | list all account information | dangerous, information leak, no safety check | deprecate this API |
| /account/<uuid> | list information for one user | dangerous, no safety check | now user should use token based authentication to access this information |

**Table of Contents**

[**Development environment**](#_3act1wi1ksdc) **2**

[**Backend API**](#_hezy386ln69v) **4**

[User Management Module](#_qr2x51bdmice) 4

[Account:](#_gf42lvw4fqv5) 4

[Analysis Creation](#_5vy6ng26uxyh) 7

[Submission Module](#_hv8y5f3t2zci) 9

[Leaderboard Module](#_mhanyrp373mp) 10

## Development environment

1. Cloud 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

1. Local development: 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

## 

## Backend API

### User Management Module

### 

### Account:

POST: /register

**Input**:

1. username (string)

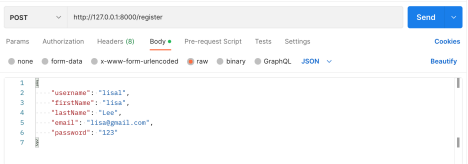
2. email (string)

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)

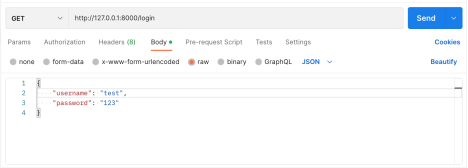


GET: /login

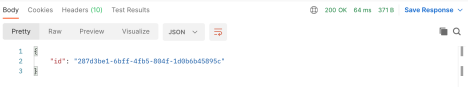
**Input:**

1. username (string)

2. password (string)

**Output:**

1. Id (uuid)



GET: /account/

**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

**Input:**

1. user authentication token in the http header field

3. Any other data that needs to be updated

**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/analys is\_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-e c2-instance-working-or-not

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

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

### Submission Module

/jobs/user\_submission

**Input**:

1. user authentication token in http header

**Output:**

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

/jobs/analysis/<analysis\_id>/user\_submission

**Input**:

1. analysis\_id

2. user authentication token in http header

**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 Module

/jobs/<analysis\_id>/leaderboard

**Input**:

1. Analysis\_id

**Output:**

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