Technical Manual

Infrastructure

UI Webserver / API

In order to power the front-end UI and back-end API on the webserver, we use nginx as the webserver.

To start nginx

```bash
systemctl start nginx
```

To stop nginx

```bash
systemctl stop nginx
```

To get nginx status

```bash
systemctl status nginx
```

We also proxy traffic destined for the API from nginx to the gunicorn application server.

We have added a custom unit and socket file for this implementation at: `/etc/systemd/system/gunicorn.service` and `/etc/systemd/system/gunicorn.socket` respectively.

To stop/start/get-status for gunicorn

Replace `nginx` in the commands above for `gunicorn`

Logs

Log files for nginx are in `/var/log/nginx/` and the gunicorn error log is also in that directory at `/var/log/nginx/gunicorn.error`

SSL

SSL is enabled via Let's Encrypt (free) certificates.

- Your certificate and chain have been saved at: `/etc/letsencrypt/live/app.moveintel.com/fullchain.pem`
- Your key file has been saved at: `/etc/letsencrypt/live/app.moveintel.com/privkey.pem`
- To non-interactively renew your certificates: `certbot renew`

S3

Two buckets are in use

- `f782d2ee-068f-11eb-adc1-0242ac120002` For processed AI videos and frames.Writable only from AI server, readable only from UI server.
- `f782d51e-068f-11eb-adc1-0242ac120002` For videos upload by users from UI. Writable only from UI server, readable only from AI server.

SQS

A single FIFO SQS queue is in use

- `f10edcdd-a593-48cc-8b6c-91481e8c3b40.fifo` in `us-west-1`

To purge the queue: `aws sqs --region us-west-1 purge-queue --queue-url "https://us-west-1.queue.amazonaws.com/83772344077/f10edcdd-a593-48cc-8b6c-91481e8c3b40.fifo"`

Backend

The backend uses python flask as the application server to produce API responses. It uses PynamoDB to add a ORM layer to DynamoDB. It also uses flask-login to mediate ACL to users stored in DynamoDB

Back end environment setup and tutorial

We will assume you will use `/csci577a/backend` as the dev directory but use whatever you prefer.

Getting the backend code

Clone the repository over to your local machine:
cd /csci577a/backend

```
git clone https://github.com/CSCI577A-MoveIntel/ui_backend.git
```

Prerequisites

Currently, the UI backend uses several python modules (boto3, flask, flask-login, and pynamodb)

We will first create a new python virtualenv for our project and activate it:

```
python3 -m venv /csci577a/backend/venv
source /csci577a/backend/venv/bin/activate
```

And now we install the prereqs mentioned above:

```
cd ui_backend
pip install -U pip wheel
pip install -r requirements.txt
```

Running the backend flask server

You are now ready to test that you can run the flask server, though we will still have to configure AWS keys

```
export FLASK_APP=/csci577a/backend/ui_backend/ui_backend
export FLASK_ENV=development
flask run --reload
```

By default, the server will now be available at: http://localhost:5000

APIs

All the APIs and their routes are in `ui_backend/ui_backend/__init__.py`. All API functionality is within the respective controllers under `ui_backend/ui_backend/controllers/`

DynamoDB Setup

To create the DynamoDB tables you can use the `models/initial_setup.py` script: `python3 ui_backend/models/initial_setup.py`

To delete the existing tables: `python3 ui_backend/models/clean.py`

To register new tables, you can add models under `ui_backend/ui_backend/models/` and import them to `ui_backend/models/initial_setup.py` and `ui_backend/models/clean.py` to create and delete tables

Testing

To run pytest: `python3 -m pytest`

References and tutorials

There are several tutorials and references that may be useful to look at for new developers.

- [Flask Tutorial](#)
- [Flask-Login Documentation](#)
- [PynamoDB Tutorial](#)
- [Boto3 AWS SDK Documentation](#)
  - QuickStart
  - DynamoDB
  - S3

Front end environment setup and tutorial.

Environment Preparation

- Install node_modules: `npm install` or `yarn`

Provided Scripts

Ant Design Pro provides some useful script to help you quick start and build with web project, code style check and test.
Scripts provided in `package.json`. It's safe to modify or add additional script:

- Start project with mock data `npm start`
- Build project `npm run build`
- Start project with proxied backend `npm run start:no-mock`
- End the server Press Control-C

Reference and tutorial

For frontend, we use the following libraries, linked with their official website.

- S3
- React
- Ant Design Pro (Overall framework)
- Ant Design (most of our React components are from this)
- Umijs (For page routing)
- Dva (Integrate Redux)
- Redux (Solution for data flow)

Queue Runner

Queue runner software processes queue entries, feeds to AI, and persists results in S3/DynamoDB.

All functionality is in `queue_runner.py` and in the models within the `models` directory. To run manually, first source the appropriate venv: `source /home/sebastian_val_dev/queue_runner_venv/bin/activate` Then run:

```
cd /home/sebastian_val_dev/queue_runner
python3 queue_runner.py
```

We also include a quota resetter at `quota_reset.py`. Whenever this is run it will reset all practitioners quota's.

Transition Tasks

- Auto start the queue runner on AI server start. [AutoStart on Ubuntu 18](#)
- Crontab to run monthly queue runner [Crontab monthly](#)