**First Iteration**

HGCG :

* Tvisha Gangwani
* Jiawen Li
* Evan Ziebart
* Jiahong He

1. GitHub Repo Link:

<https://github.com/JiahongHe/Personal-Facial-Identification-System>

1. Pre-commit : Personal-Facial-Identification-System/test/[pre\_commit.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/pre_commit.sh)

Which would run the following:

* Style checker: use pylint to automatically run static analysis on all the .py files, and save the output to ./test/reports/[styleCheckReport.txt](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/reports/styleCheckReport.txt)

(we found that this contains a lot of false positives)

* Install all the required Python module
* Run [test\_main.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/test_main.sh) that would run all the tests.

Post-commit CI : Personal-Facial-Identification-System/.travis.yml

Which would:

* Setup the environment (linux operating system, Python)
* Install required Python packages.
* Run [test\_main.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/test_main.sh) that would run all the tests.

1. Unit tests and tests for facial recognition :

The following script would automatically run all the tests:

Personal-Facial-Identification-System/test/[test\_main.sh](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/test_main.sh)

Which include

* Tests for Django server functionalities.
* Tests for the correctly getting data from the server.

The tests include:

* src/server/backendServer/requestHandle/[tests.py](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/src/server/backendServer/requestHandler/tests.py)

Which would test:

1. Single User/Song object creation and query from the database.
2. Multiple User/Song object creation and query from the database.

* src/server/backendServer/userRegistration/[tests.py](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/src/server/backendServer/userRegistration/tests.py)

Which would test:

1. The user registration page can be correctly served with the correct URL.
2. The form in the served registration page is an instance of the form we intended to send.
3. The URL used for the registration page to submit user registration form can’t be accessed by request not sent by the registration page.

* ./test/[getAllUsers\_test.py](https://github.com/JiahongHe/Personal-Facial-Identification-System/blob/master/test/getAllUsers_test.py)

Which would test:

1. It can correctly connect to the server and retrieve user data.
2. The retrieved user data is not corrupted and can be correctly converted into the format later used for face recognizing.

Note:

The facial recognition computation is done by the module Face\_Recognition (Github: [ageitgey](https://github.com/ageitgey)/[face\_recognition](https://github.com/ageitgey/face_recognition)) Which is maintained and tested by other developers, so we didn’t test it ourselves.

The same also applies for the form and password validation in the Django server, all of which come with the Django module and are tested by the developers of Django, so we didn’t test it.

(We tried to write some tests anyway, but found it really hard to do, and then during the search on StackOverflow.com, we found other people with the same question are told that we are not supposed to unit test the off the shelf functions in Django, and no test method is provided by the developers of Django)