




COS 301 PHASE
3 – GROUP II



MERLOT FACIAL RECOGNITION SUBSYSTEM



MEET THE TEAM

EMMA COETZER, HERBERT MAGAYA, KOKETSO MOLAWA, JUSTIN ROSS



EMMA COETZER



- Responsible for creating the MongoDB database to store the client's photographs used for the facial recognition. I created methods to interact with the database. Also responsible for the landing page and the PowerPoint presentation.

HERBERT MAGAYA

- Responsible for communicating with the rest of Merlot teams, most specifically with the authentication team. Also implemented the UPDATE function responsible for updating and inserting into the databank. Also set up hosting on Heroku.



KOKETSO MOLAWA



- Responsible for integrating the facial recognition library provided by node.js, into the overall subsystem. This involves writing methods to also loop through the entire JSON databank for do the facial recognition and return the ClientID.

JUSTIN ROSS

- Responsible for writing the GET function responsible for interfacing with the databank module as well as providing the Authentication Subsystem with an interface to access the Facial Recognition. Also responsible for creating the LOGGING subsystem within the facial recognition subsystem.





LIVE HOSTING



HEROKU



FOR THE
DATABASE, API,
AND LOG



UNIT TESTING


```
main.js x database.js x
1 var dataset = require("../database.js");
2 var http = require("http");
3
4 http.createServer(function(req, res){
5   res.writeHead(200, {'Content-Type': 'text/html'});
6   var json = {"1" : "sueifhoadoiada", "2" : "iqohhfascnczjzdfd", "3" : "oefheodahcjsdnfushdf", "4" : "ioadhfdsnjvnjjsrjf", "5" : "iaofsdjcnxkvsrshg"};
7   dataset.Insert("1234", json);
8   res.end();
9 }).listen(8000);
```

```
main.js x database.js x
1 const MongoClient = require('mongodb').MongoClient;
2 const uri = "mongodb+srv://admin:admin123@facialrecdataset-umcor.mongodb.net/facialrecdataset?ssl=true&authSource=admin";
3 const client = new MongoClient(uri, { useNewUrlParser: true });
4 //var exports = module.exports = {};
5
6 exports.Insert = function(client_id, images_json){
7   client.connect(err => {
8     if(err) throw err;
9     const collection = client.db("FacialRecDataSet").collection("FacialRecTable");
10    var obj = {clientId : client_id, photos : images_json, activated : true};
11    collection.insertOne(obj, function(error, result){
12      if(error) throw error;
13      console.log("Inserted");
14      else return client_id;
15    });
16    client.close();
17  });
18 };
19
```

CONTEXT

Project 0

EMMA'S ORG - 2019-03-26 > PROJECT 0 > CLUSTERS

FacialRecDataSet

VERSION
4.0.6

REGION
N. Virginia (us-east-1)

PROJECT

Clusters

Alerts 0

Backup

Access

Settings

Stitch Apps

Charts

Docs

Support

Overview

Real Time

Metrics

Collections

Command Line Tools

1 DATABASES 1 COLLECTIONS

+ Create Database

Q NAMESPACES

FacialRecDataSet

FacialRecTable

FacialRecDataSet.FacialRecTable

COLLECTION SIZE 192B

TOTAL DOCUMENTS 1

INDEXES TOTAL SIZE 16KB

REFRESH

+ INSERT DOCUMENT

Find

Indexes

FILTER

{"filter":"example"}

Find

Reset

QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("5c99e33a45ee0b2a08fb4561")
clientId: "1234"
> photos: Object
  activated: true
```

Additional DB Testing File:

<https://github.com/ul7079544/Merlot-Facial-Recognition/blob/master/test/database/database-test.js>

Facial Recognition System

Database

Insert(client_id, images)

```
[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
```

Activate(client_id)

```
[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
>[] inserts the record into the database
```

Update(client_id, images)

```
[] updates the record in the database
>[] updates the record in the database
>[] updates the record in the database
>[] updates the record in the database
>[] updates the record in the database
```

Delete(client_id)

```
[] deactivates the record in the database
>[] deactivates the record in the database
>[] deactivates the record in the database
>[] deactivates the record in the database
>[] deactivates the record in the database
```

25 passing (40ms)

```

var request = require('request');
var expect = require('chai').expect;

describe('post()', function(){
  it('should authenticate an image and return a clientID', function(){

    //Arrange
    var sent={
      type: "authenticate",
      image: "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA";
    };
    var expected={clientid: 1234};

    //Act
    var result;
    request.post(
      'http://localhost:3000/',
      { json: sent },
      function (error, response, body) {
        if (!error && response.statusCode == 200) {
          result=body;
          //Assert
          expect(result).to.be.equal(expected);
        }
      }
    );
  });

  it('should add an array of images and return a boolean for success', function(){

    //Arrange
    var sent={
      type: "update",
      images: [
        "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA",
        "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA",
        "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA",
        "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA",
        "/9j/4AAQSkZJRgABAgEAYABgAAD/7gAOQWRvYmUAZAAAAAAAAAB/+EUI0V4aWYAAE1NACoAAAAIAAcBMgACAAAFAAAAAGIBOwACAAABwAAAHZHRgADAAAAAQAEABHSQADAAAAQA/AACcnQABAAAADgAAAADqHAAHAA"
      ]
    };
    var expected={success: true};
  });
});

```

```

post()
  [ ] should authenticate an image and return a clientID
  [ ] should add an array of images and return a boolean for success
  [ ] should return an error
  [ ] should return an error

4 passing (77ms)

```

Link to Log Test: <https://github.com/u17079544/Merlot-Facial-Recognition/blob/master/test/log/LogTest.js>

```
add
  [] should return a JSON array containing the added element (568ms)
  [] should return an error
  [] should return an error
  [] should return an error
  [] should return an error

get
  [] should return a JSON array containing a set of elements within the specified
boundaries (524ms)
  [] should return an error
  [] should return an error

8 passing (1s)
```



LINKS TO THE
TECH THAT WE
USED



GITHUB,
ZENHUB AND
LANDING PAGE



PROJECT MANAGEMENT TOOL:

[HTTPS://GITHUB.COM/U17079544/MERLOT-FACIAL-RECOGNITION/BLOB/MASTER/VIEWS/PAGES/INDEX.EJS#ZENHUB](https://github.com/U17079544/MERLOT-FACIAL-RECOGNITION/blob/master/views/pages/index.EJS#ZENHUB)



LANDING PAGE:

[HTTPS://CS.UP.AC.ZA/TEAMS/PAGES/SITE_VIEW/36](https://cs.up.ac.za/teams/pages/site_view/36)



GITHUB RESPOSITIORY:

[HTTPS://GITHUB.COM/U17079544/MERLOT-FACIAL-RECOGNITION](https://github.com/U17079544/MERLOT-FACIAL-RECOGNITION)