# Dynamic Resource Management

## Assignment 2

ECE1779 Introduction to Cloud Computing

## Team Members

Denis Noskov • 1007140666

Sheran Cardoza • 1001070066

Amarpreet Singh • 1002513764

## Date of Submission

March 22nd, 2021

## 1. Project Description

The purpose of this assignment is to extend the Mask Detector user app by implementing dynamic resource management. This consists of a load balancer that evenly distributes incoming HTTP requests to a pool of workers, where each worker is an EC2 instance that is hosting the user app. A manager webapp is provided to control resource management, wherein workers can be manually created or destroyed, or an automatic scaling policy can be set to launch workers based on demand. The manager webapp also allows monitoring workers to ensure healthy operation.

### 1.1. Features

Backend:

- S3 is used to store images, which acts as a common bucket accessible to all workers.
- RDS is used as the database, which is also accessible to all workers.
- ELB (Elastic Load Balancer) is used as the load balancer to distribute HTTP requests to a pool of workers.
- The pool of workers is defined by a single Target Group, which consists of a pool of EC2 machines.
- Custom auto-scaling policy is implemented to create and destroy workers based on CPU utilization. The thresholds and growth and shrink ratios are configurable via the manager webapp.
- CloudWatch is used to log data for monitoring the workers.

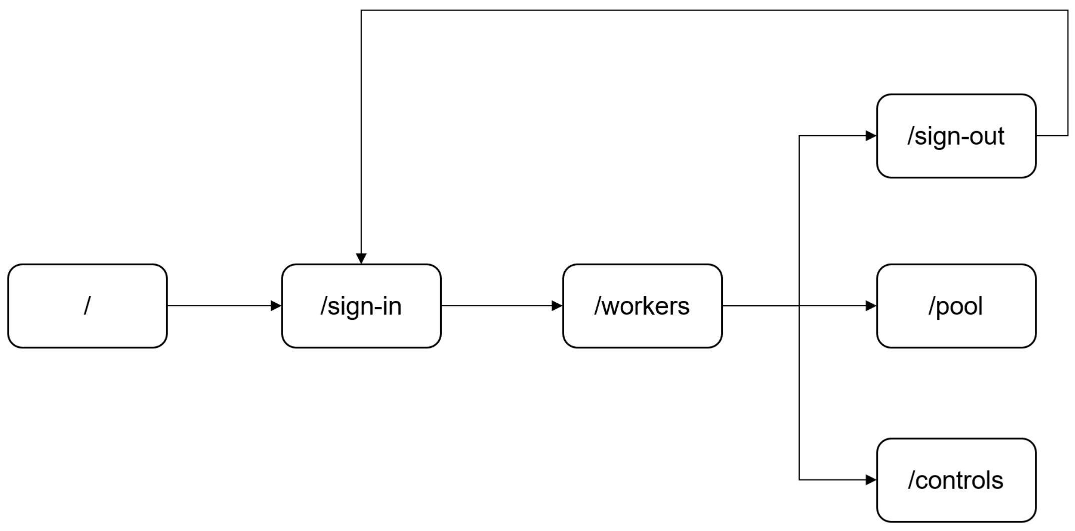
Manager Webapp Frontend:

- Login page, allowing only the admin to login
- Charts displaying the CPU utilization and HTTP requests for each worker
- Chart displaying the number of healthy and unhealthy workers
- Pool controls to manually scale the pool and set an automatic scaling policy
- General controls to clear the database, clear S3, stop workers and stop the manager

User Webapp Frontend:

- Login page, allowing each user to upload their own files
- Upload image via URL or from a local file, and detect faces and masks
- View previously uploaded images, and delete history
- Admin can create users and delete users

### 1.2. Site Map



- /
    - Redirects to /sign-in if user is not signed in, else redirects to /workers
- /sign-in
    - Allows the admin to sign in. All other users are not allowed to sign in.
- /sign-out
    - An intermediate link accessed when signing out of the webapp.
- /workers
    - Main page that shows a chart of the number of healthy and unhealthy workers over time, and details of each worker. Worker details include instance ID, health, and charts showing CPU utilization and HTTP requests.
- /pool
    - Pool scaling policy can be set in this page. Workers can be manually created and terminated. Auto-scaling policy can also be set to automatically create and terminate workers based on demand. The parameters for this policy include minimum and maximum CPU utilization threshold, below which workers are terminated and above which workers are created. It also includes a growth and shrink ratio to control how many workers are added or removed by the auto-scaler.
- /controls
    - General controls to clear S3, clear database, stop workers only, and stop both workers and manager.

All pages display the URL to the Mask Detector user webapp in the navigation bar.

The following end-points serve data requests to update live charts, and return JSON objects containing tuples of datapoint and timestamp:

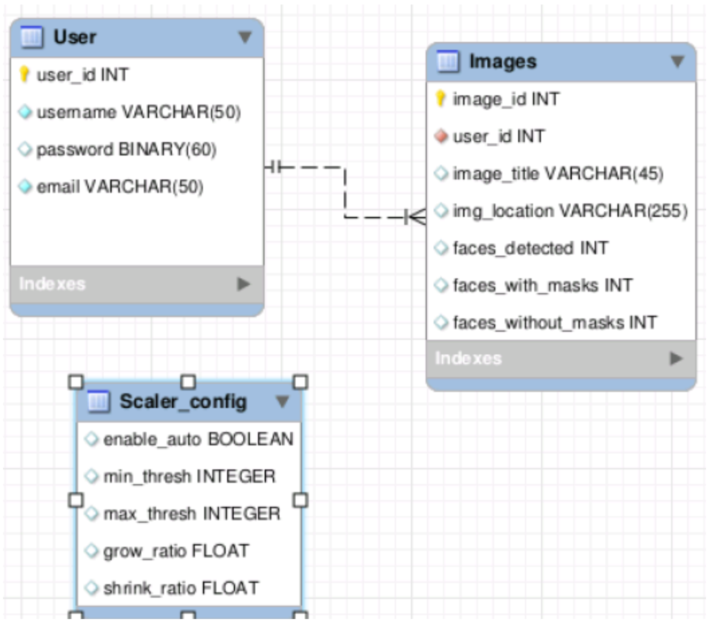
- /cpuutil-data-all/<worker_id>

    - Returns all data within a 30 minute window of CPU utilization for a given worker
- /cpuutil-data-new/<worker_id>
    - Returns the latest CPU utilization for a given worker
- /httpreq-data-all/<worker_id>
    - Returns all data within a 30 minute window of HTTP requests for a given worker
- /httpreq-data-new/<worker_id>
    - Returns the latest HTTP requests for a given worker
- /healthyworkers-data-all
    - Returns all data within a 30 minute window of number of healthy workers
- /healthyworkers-data-new
    - Returns the latest number of healthy workers
- /unhealthyworkers-data-all
    - Returns all data within a 30 minute window of number of unhealthy workers
- /unhealthyworkers-data-new
    - Returns the latest number of unhealthy workers
- /live-num-workers
    - Returns the current total number of workers

### 1.3. Database

**Location**: database-1.cf18wqhdlvtt.us-east-1.rds.amazonaws.com
**Credentials**:
- Username: admin
- Password: pass12345

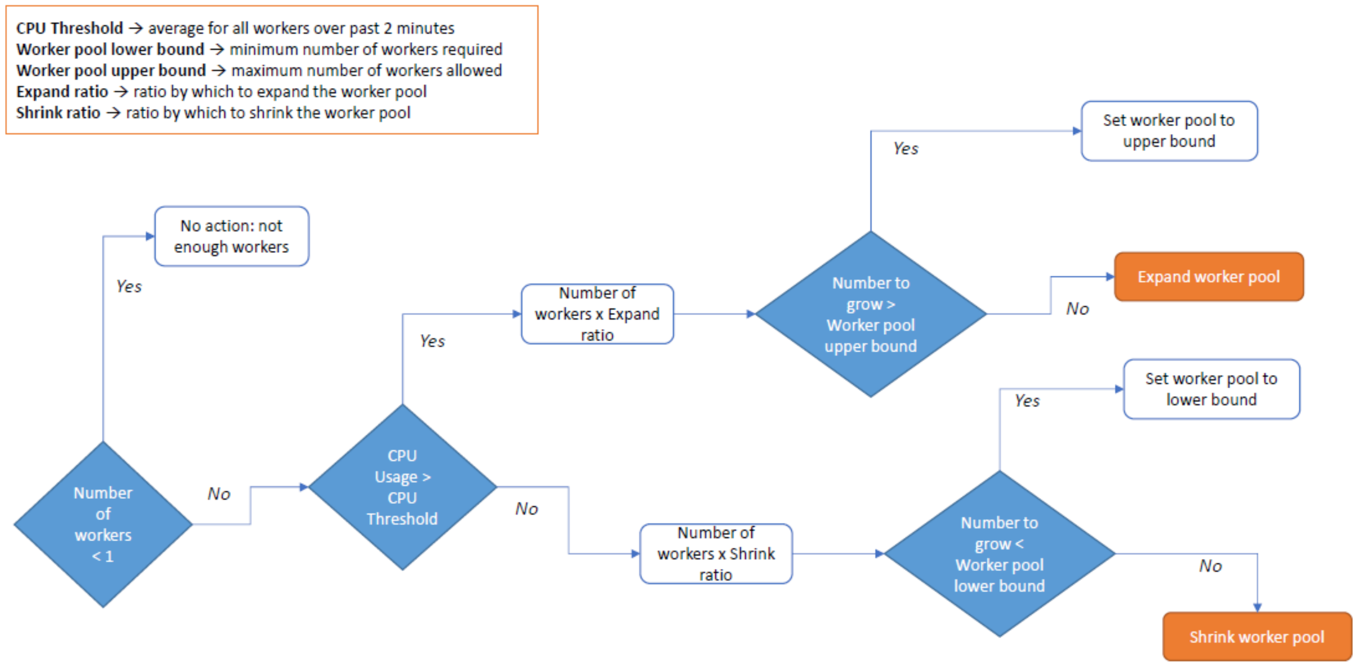
The database connected to the application is stored in RDS and is organized into three tables:
- User: stores information about the registered users
- Images: this table stores information related to images uploaded by the user after having been processed by the mask detection model. The location of the images indicates the S3 bucket in which they are stored
- Scaler_config: stores information relating to the configurations for the auto-scaler



## 2. Auto-Scaler

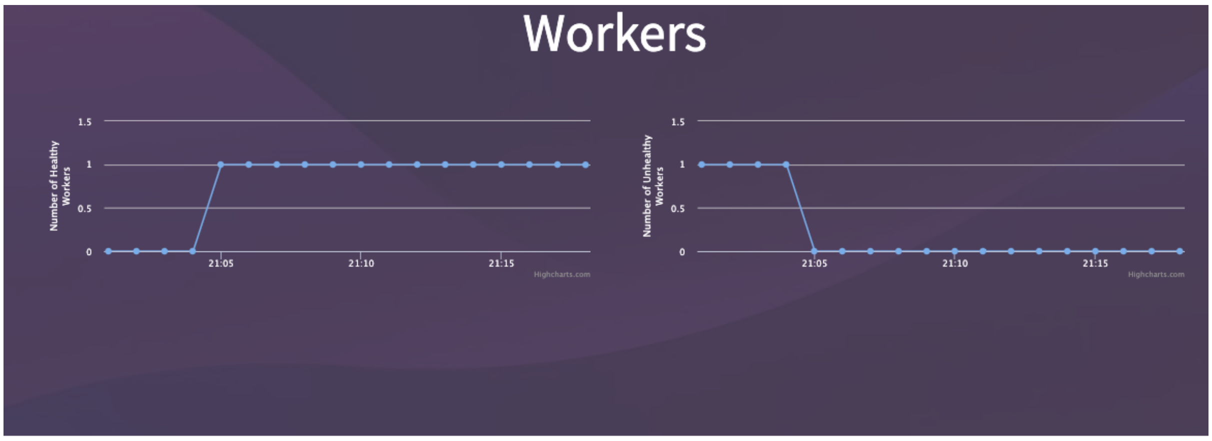
### 2.1. Implementation

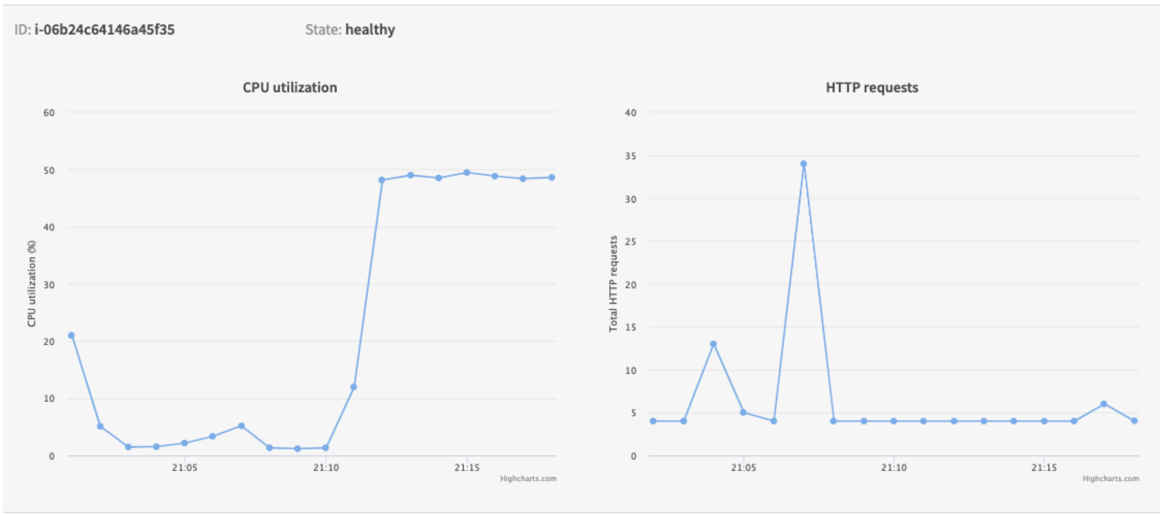
The auto-scaler runs every 60 seconds and adheres to the auto-scaling policy. The auto-scaling policy is defined by: (1) growth cpu threshold, (2) shrink cpu threshold, (3) growth ratio, and (4) shrink ratio. The lower limit of the worker pool size is 1 and the upper limit of the worker pool size is 8. Validation is performed on the auto scaling policy parameters as it is acquired through the UI.


CPU Threshold → average for all workers over past 2 minutes
Worker pool lower bound → minimum number of workers required
Worker pool upper bound → maximum number of workers allowed
Expand ratio → ratio by which to expand the worker pool
Shrink ratio → ratio by which to shrink the worker pool
Number of workers < 1
Yes
No action: not enough workers
No
CPU Usage > CPU Threshold
Yes
Number of workers x Expand ratio
Number to grow > Worker pool upper bound
Yes
Set worker pool to upper bound
No
Expand worker pool
No
Number of workers x Shrink ratio
Number to grow < Worker pool lower bound
Yes
Set worker pool to lower bound
No
Shrink worker pool

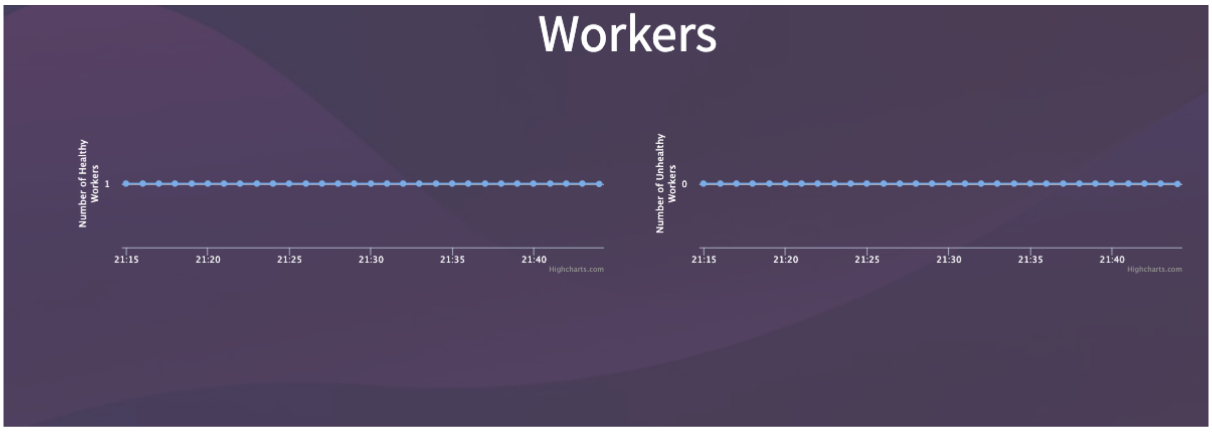
### 2.2. Results

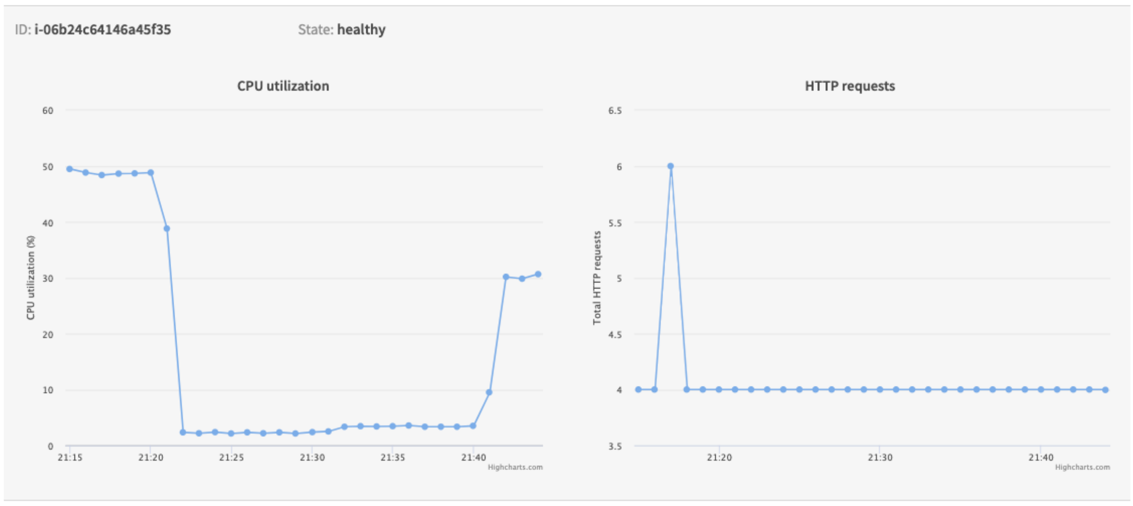
1) Single worker is added upon launch, but it is not loaded with HTTP requests yet



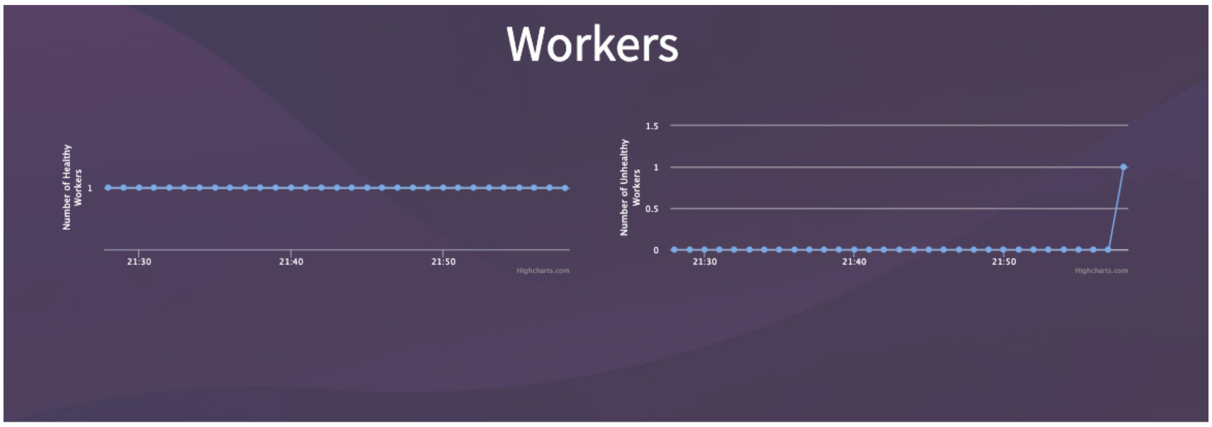


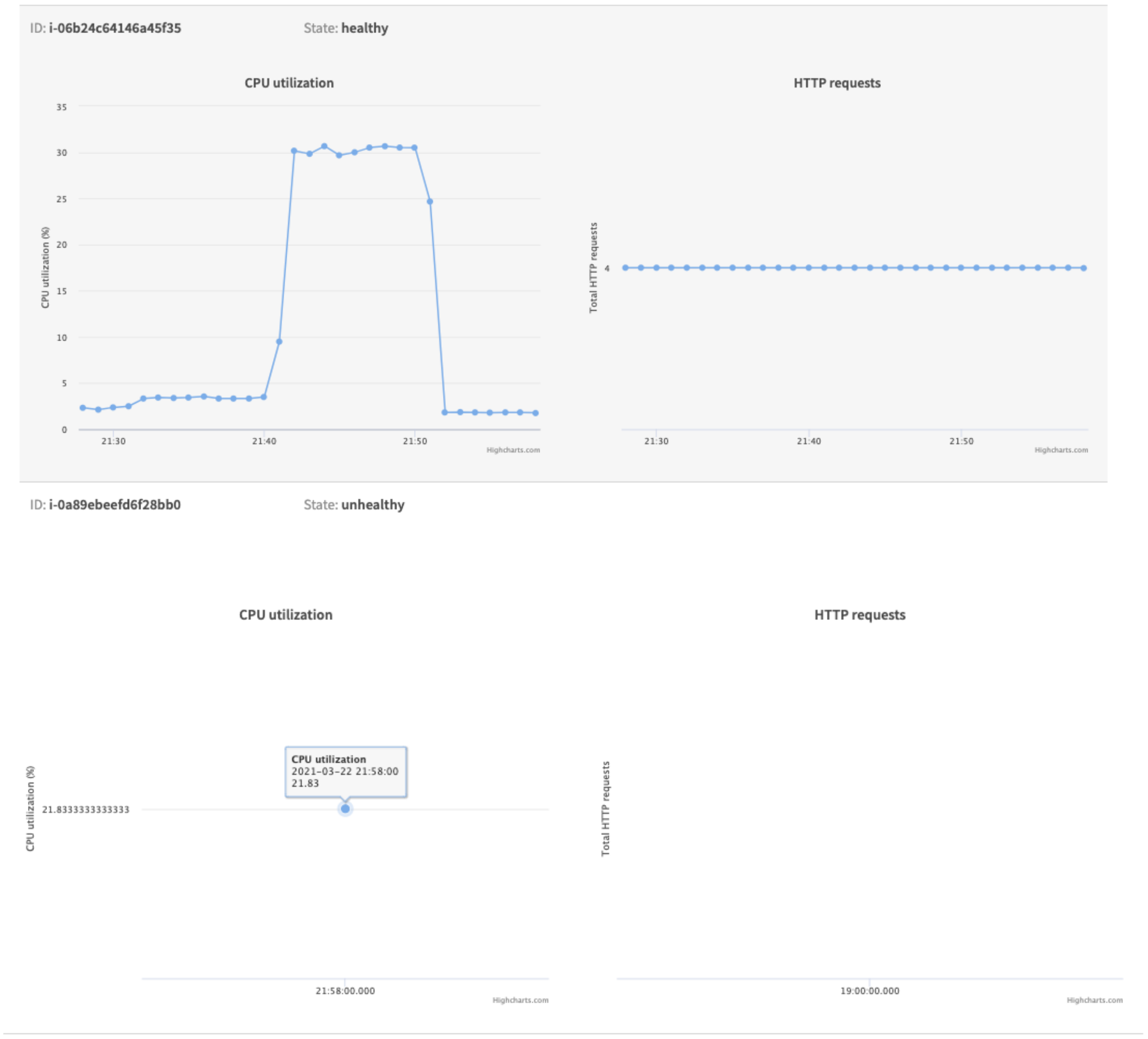
2) Single worker is now loaded with HTTP requests which spikes its CPU utilization




ID: i-06b24c64146a45f35 State: healthy

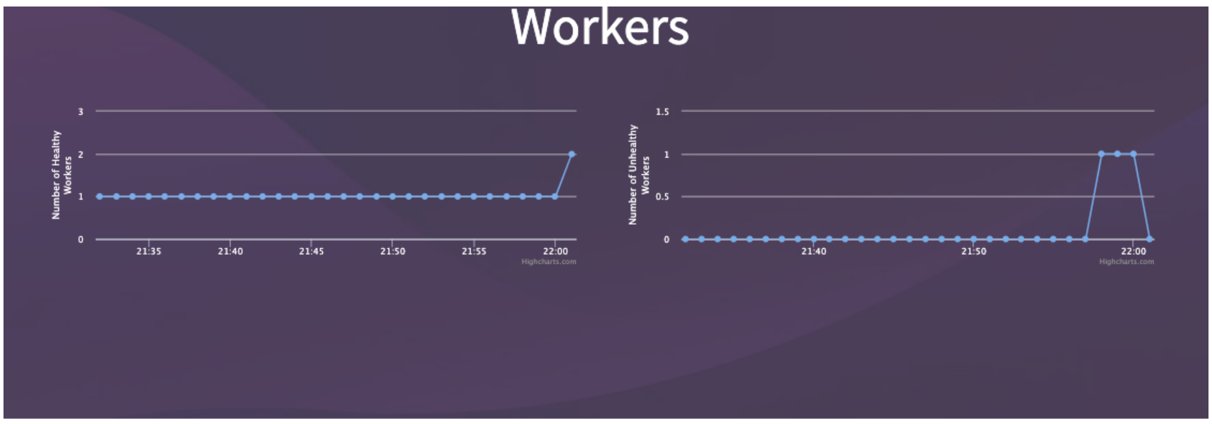
3) Auto-scaler automatically adds a second worker once maximum CPU utilization threshold is exceeded

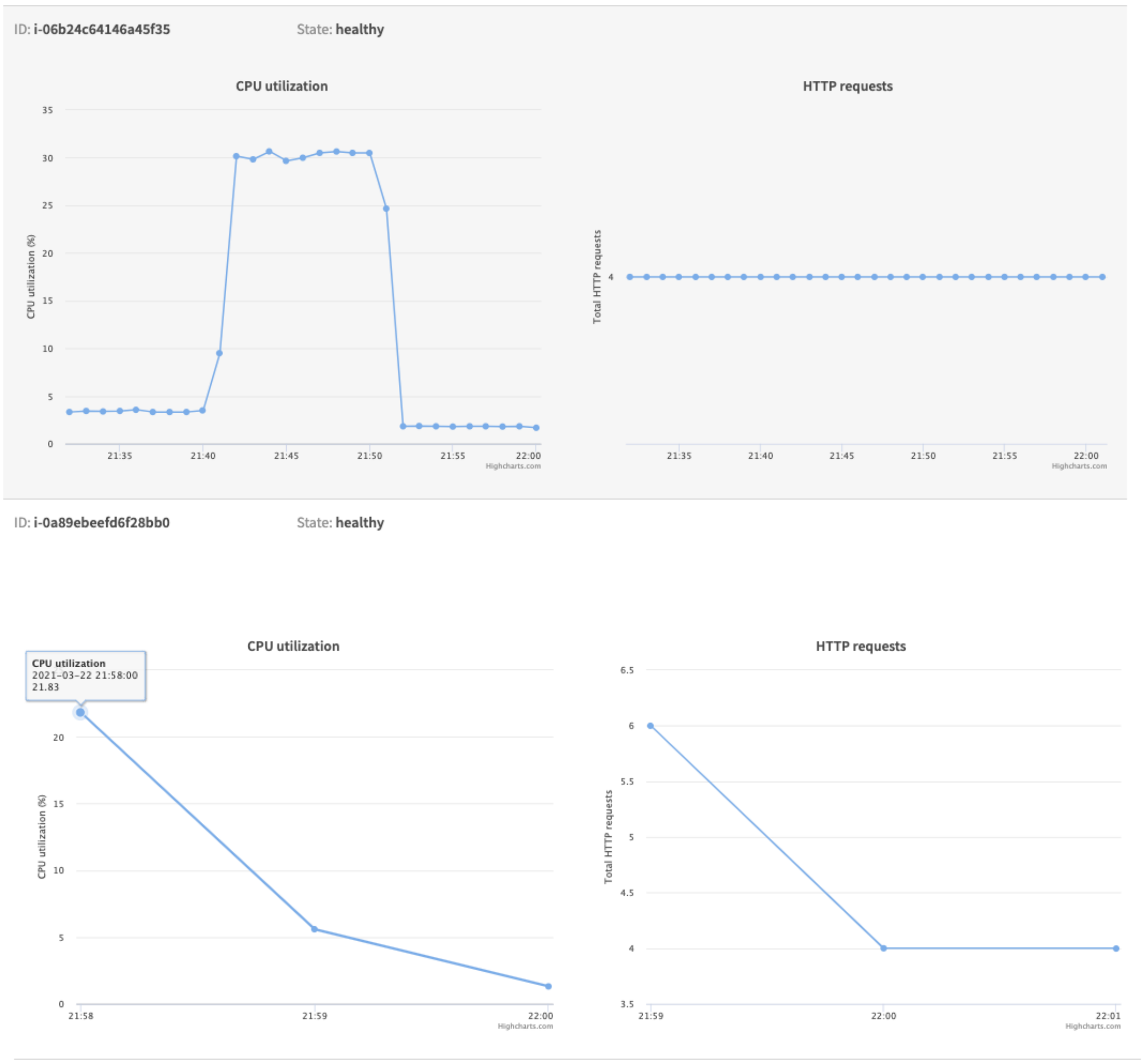



ID: i-06b24c64146a45f35 State: healthy

ID: i-0a89ebeefd6f28bb0 State: unhealthy

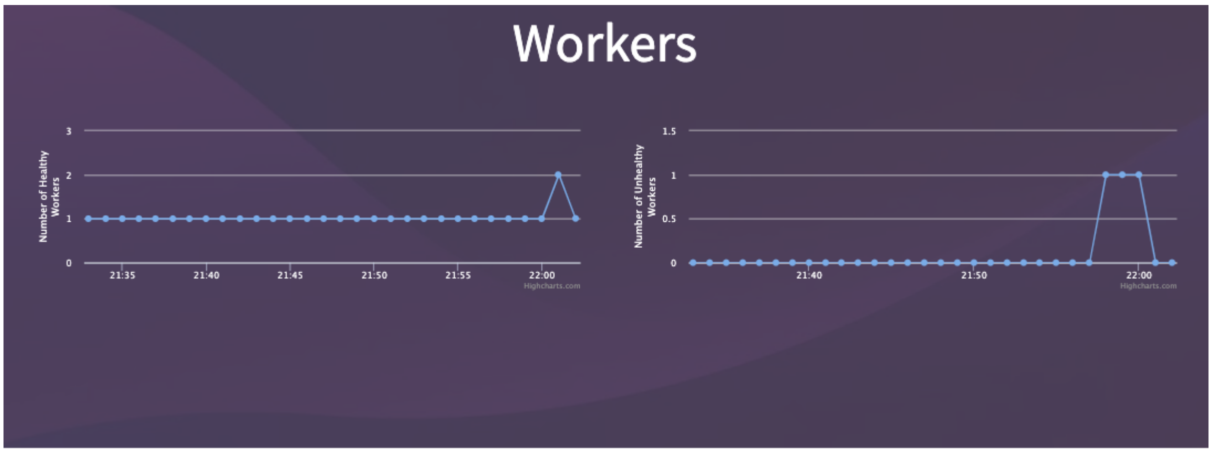
4) Now both workers are stable, and the auto-scaler is not diverging, i.e. it is not spuriously adding or removing workers

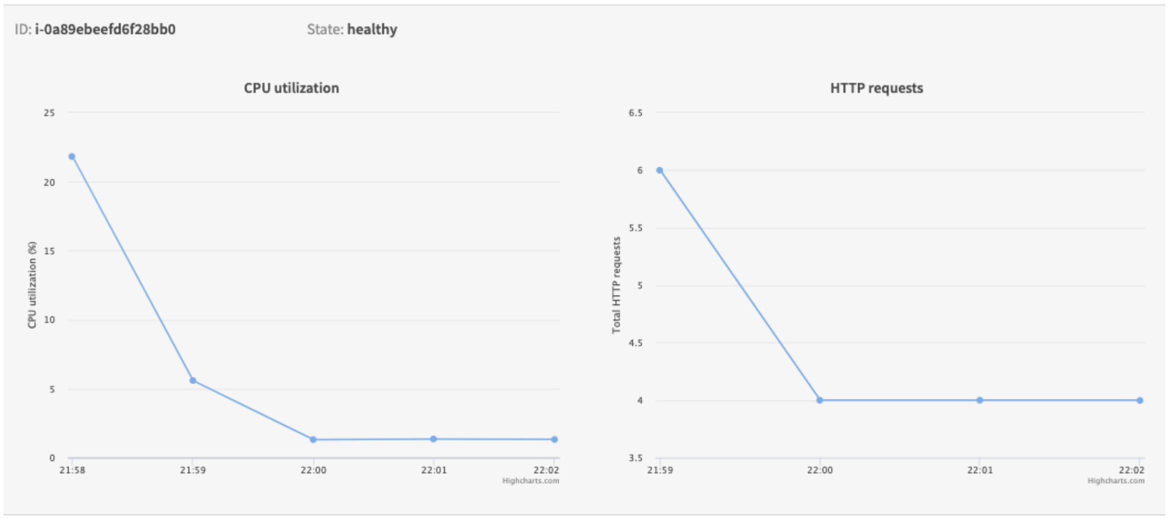


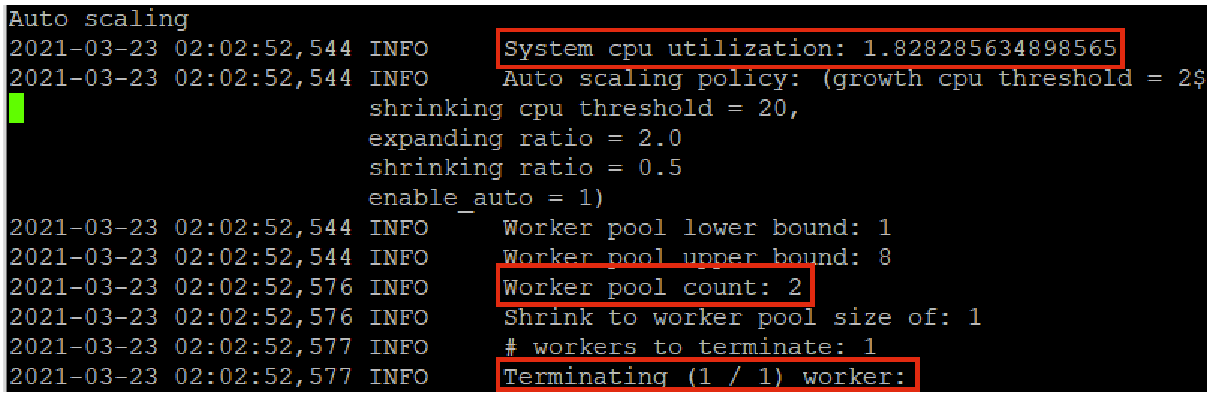

ID: i-06b24c64146a45f35 State: healthy

ID: i-0a89ebeefd6f28bb0 State: healthy

5) Step 5 - Since HTTP request load has dropped, the average CPU utilization has dropped below the minimum threshold causing the auto-scaler to automatically remove the second worker

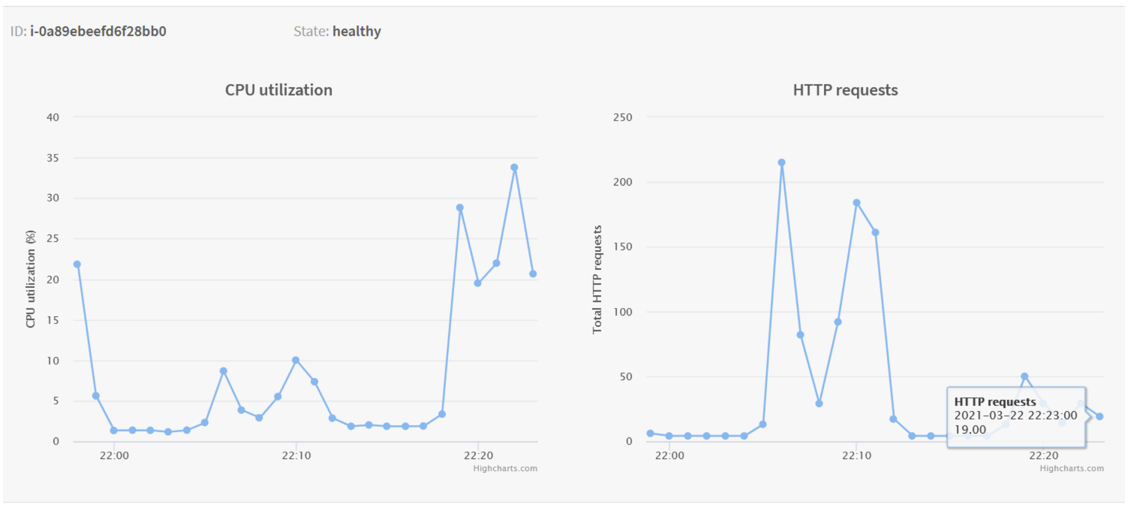


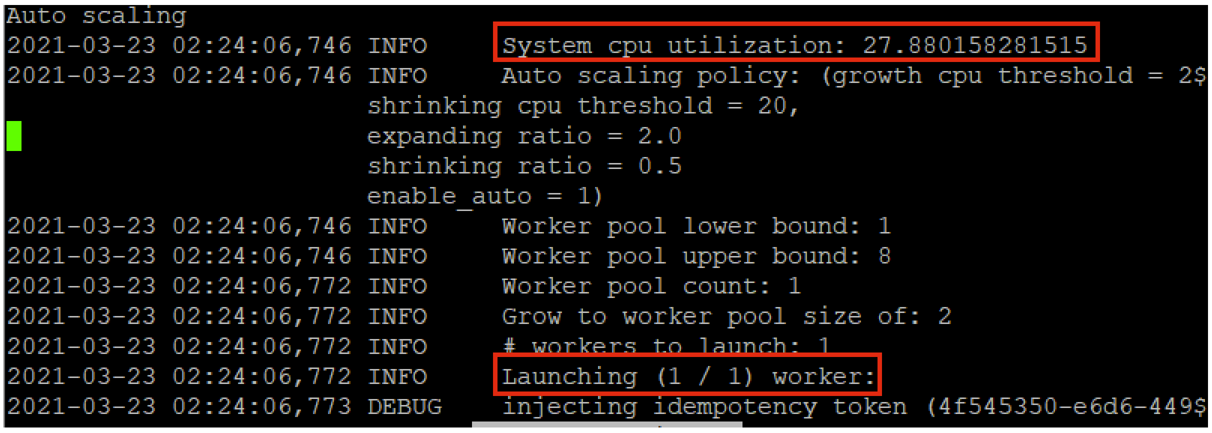




```
Auto scaling
2021-03-23 02:02:52,544 INFO     System cpu utilization: 1.828285634898565
2021-03-23 02:02:52,544 INFO     Auto scaling policy: (growth cpu threshold = 2$
                        shrinking cpu threshold = 20,
                        expanding ratio = 2.0
                        shrinking ratio = 0.5
                        enable_auto = 1)
2021-03-23 02:02:52,544 INFO     Worker pool lower bound: 1
2021-03-23 02:02:52,544 INFO     Worker pool upper bound: 8
2021-03-23 02:02:52,576 INFO     Worker pool count: 2
2021-03-23 02:02:52,576 INFO     Shrink to worker pool size of: 1
2021-03-23 02:02:52,577 INFO     # workers to terminate: 1
2021-03-23 02:02:52,577 INFO     Terminating (1 / 1) worker:
```

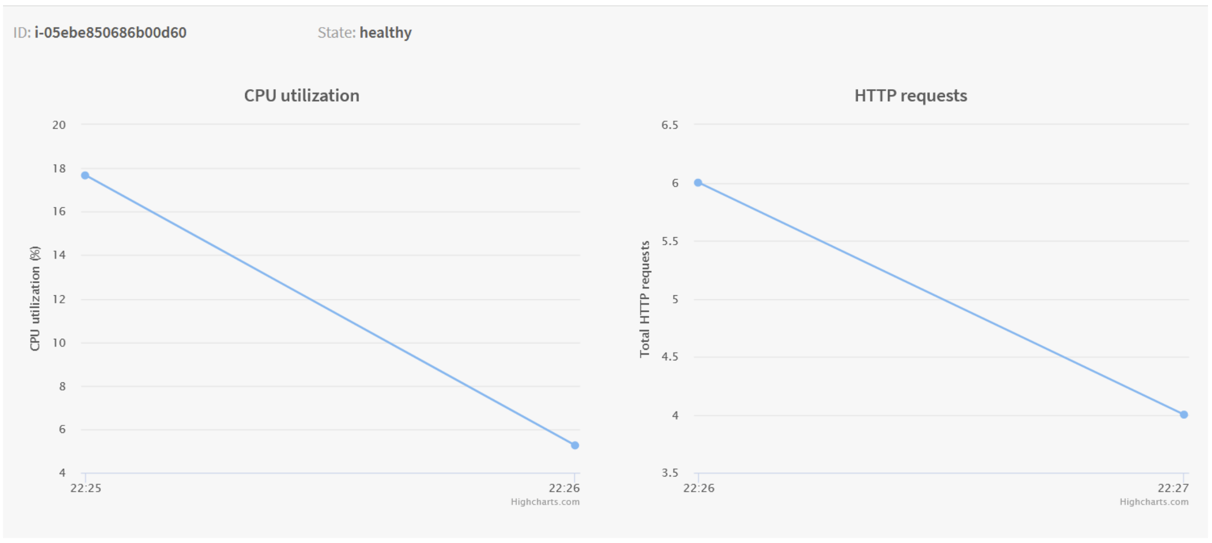
6) Increasing CPU Usage: another test case was conducted to evaluate the auto-scaler capability to handle increased workload by uploading a range of images in a row. As the CPU usage increased over the Growth Threshold of 25%, auto-scaler added another worker

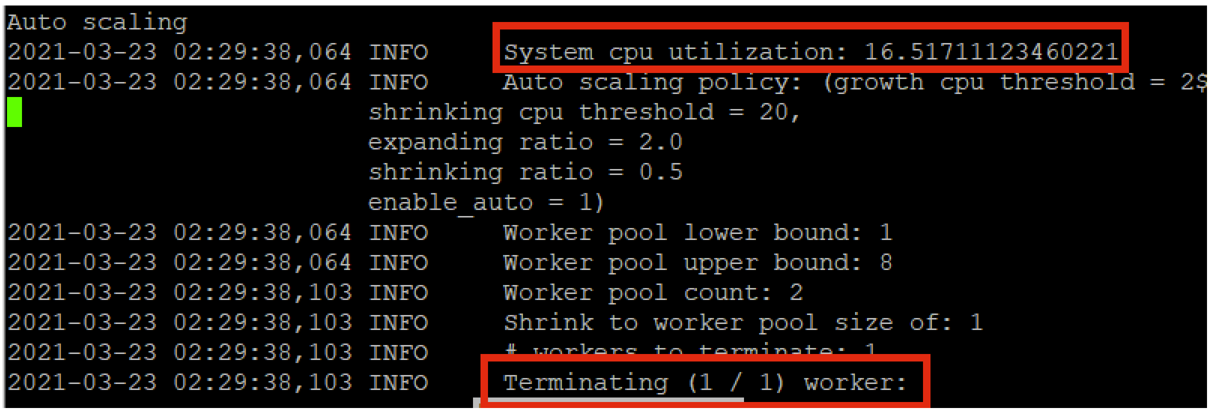




```
Auto scaling
2021-03-23 02:24:06,746 INFO     System cpu utilization: 27.880158281515
2021-03-23 02:24:06,746 INFO     Auto scaling policy: (growth cpu threshold = 2$
                                 shrinking cpu threshold = 20,
                                 expanding ratio = 2.0
                                 shrinking ratio = 0.5
                                 enable_auto = 1)
2021-03-23 02:24:06,746 INFO     Worker pool lower bound: 1
2021-03-23 02:24:06,746 INFO     Worker pool upper bound: 8
2021-03-23 02:24:06,772 INFO     Worker pool count: 1
2021-03-23 02:24:06,772 INFO     Grow to worker pool size of: 2
2021-03-23 02:24:06,772 INFO     # workers to launch: 1
2021-03-23 02:24:06,772 INFO     Launching (1 / 1) worker:
2021-03-23 02:24:06,773 DEBUG    injecting idempotency token (4f545350-e6d6-449$
```

After a new worker was added and reached a healthy state, the CPU has decreased to 16.5%, passing the Shrinking Threshold of 20%, triggering the auto-scaler to terminate one of the working, bringing it to a new total of 1 worker



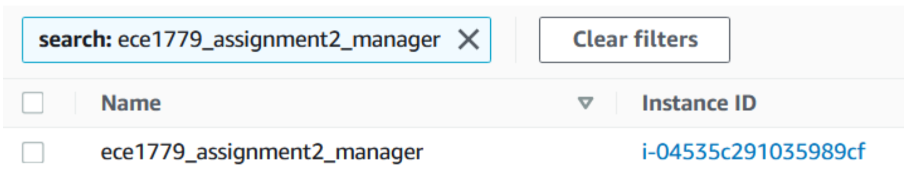

```
Auto scaling
2021-03-23 02:29:38,064 INFO     System cpu utilization: 16.51711123460221
2021-03-23 02:29:38,064 INFO     Auto scaling policy: (growth cpu threshold = 2$
                        shrinking cpu threshold = 20,
                        expanding ratio = 2.0
                        shrinking ratio = 0.5
                        enable_auto = 1)
2021-03-23 02:29:38,064 INFO     Worker pool lower bound: 1
2021-03-23 02:29:38,064 INFO     Worker pool upper bound: 8
2021-03-23 02:29:38,103 INFO     Worker pool count: 2
2021-03-23 02:29:38,103 INFO     Shrink to worker pool size of: 1
2021-03-23 02:29:38,103 INFO     # workers to terminate: 1
2021-03-23 02:29:38,103 INFO     Terminating (1 / 1) worker:
```

## 3. Usage Instructions

### 3.1. Manager web application

a. Start the Manager EC2 instance - which is created on the AWS education account
   i. Username: **denis.noskov@mail.utoronto.ca**
   ii. Password: **EYF5*%QNu7**
   iii. EC2 instance name: **ece1779_assignment2_manager**
   iv. EC2 instance keypair: **keypair.pem**

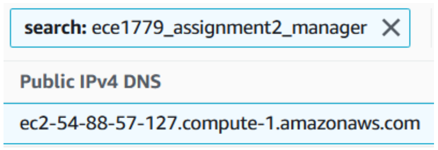
b. Search for the name "**ece1779_assignment2_manager**" in the list of instances in EC2 Services and start instance ID i-04535c291035989cf. This will automatically start the manager app on port 5000, by initiating the script that runs the application
   i. Script path: **/home/ubuntu/Documents/ECE1779/a2/start_manager_main.sh**



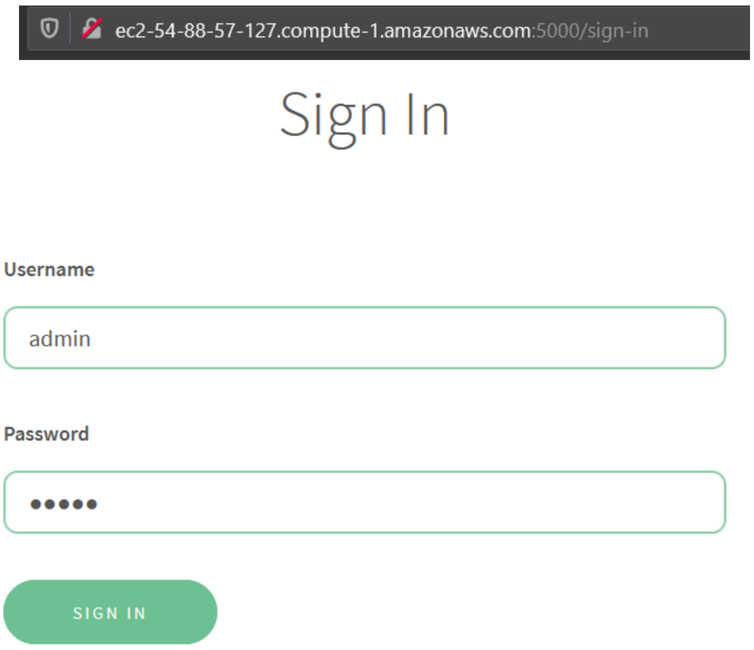
| Name | Instance ID |
|---|---|
| ece1779_assignment2_manager | i-04535c291035989cf |

c. Note the Public IP and navigate to the manager app website once the instance has fully initiated: **<Manager EC2 Instance IP>:5000**

*Example:*



| Public IPv4 DNS |
|---|
| ec2-54-88-57-127.compute-1.amazonaws.com |



d. Login to manager (Note: same credentials also apply to the user app):
   Username: **admin**
   Password: **12345**

e. To view the log associated with the auto scaler, run the following command in the terminal of the manager app:
   **sudo nano /home/ubuntu/ECE1779/a2/auto_scaling.log**

### 3.2. User web application

Refer to {userapp_instructions.pdf} attached in the submission tar file.

### 3.3. Test API

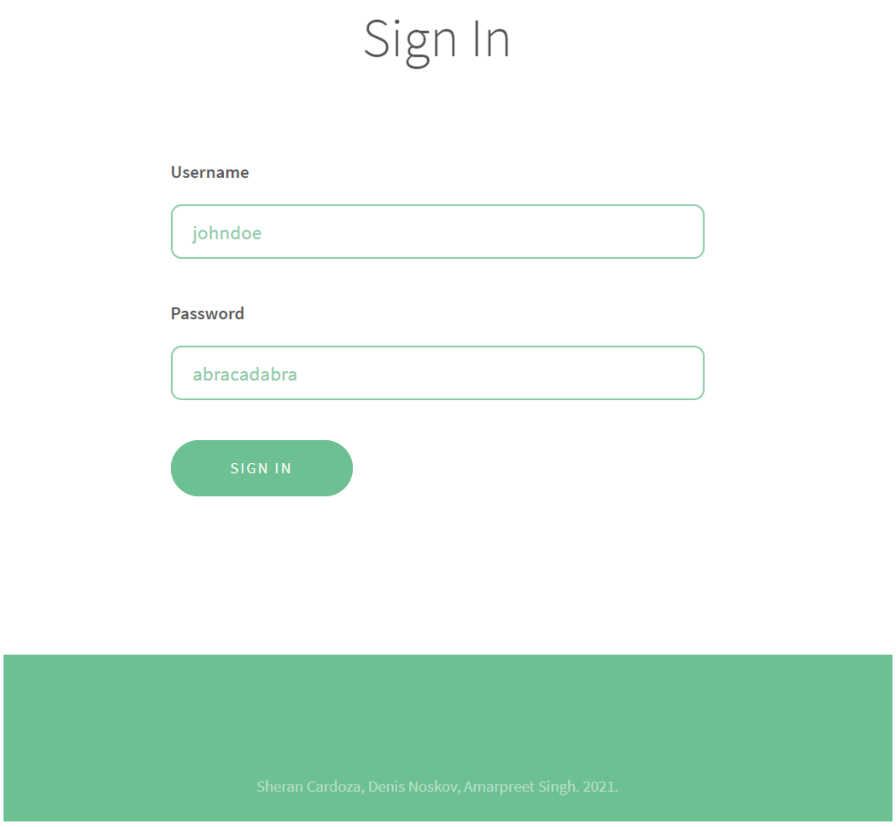
URL endpoints are provided for automated testing of the user webapp's login and upload functionality. Each endpoint accepts form data via a POST method and returns a JSON object indicating success or containing error messages.

- /api/register
  - This URL accepts form data with 'username' and 'password' strings to register a new user.
- /api/upload
  - This URL accepts multipart form data with 'username' and 'password' strings, and a 'file' object pointing to a local image file to upload.

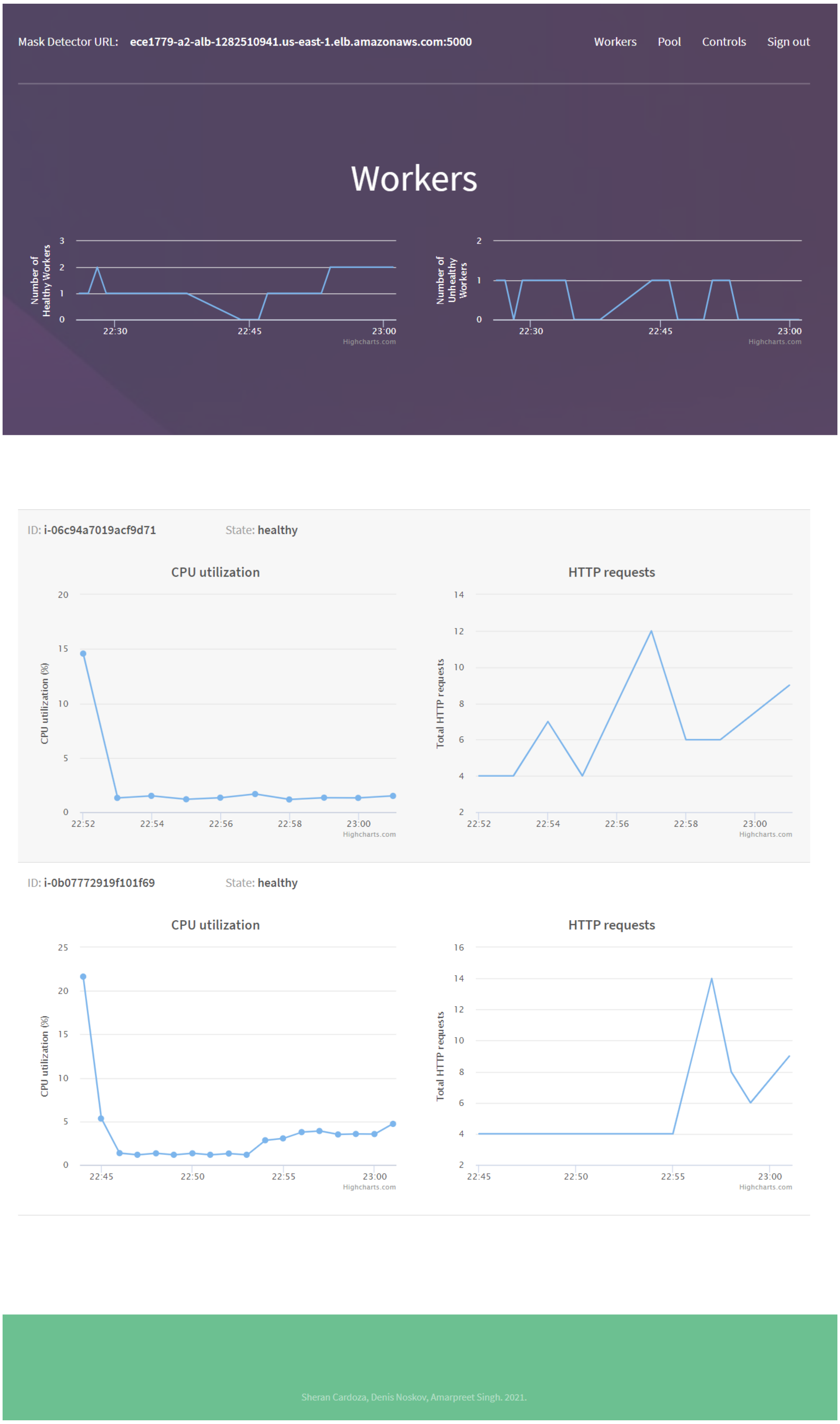
## 4. Screenshots

### 4.1. Manager webapp

Login page:



Workers page:


Mask Detector URL: **ece1779-a2-alb-1282510941.us-east-1.elb.amazonaws.com:5000**

Workers Pool Controls Sign out

Workers

ID: **i-06c94a7019acf9d71** State: **healthy**

CPU utilization

HTTP requests

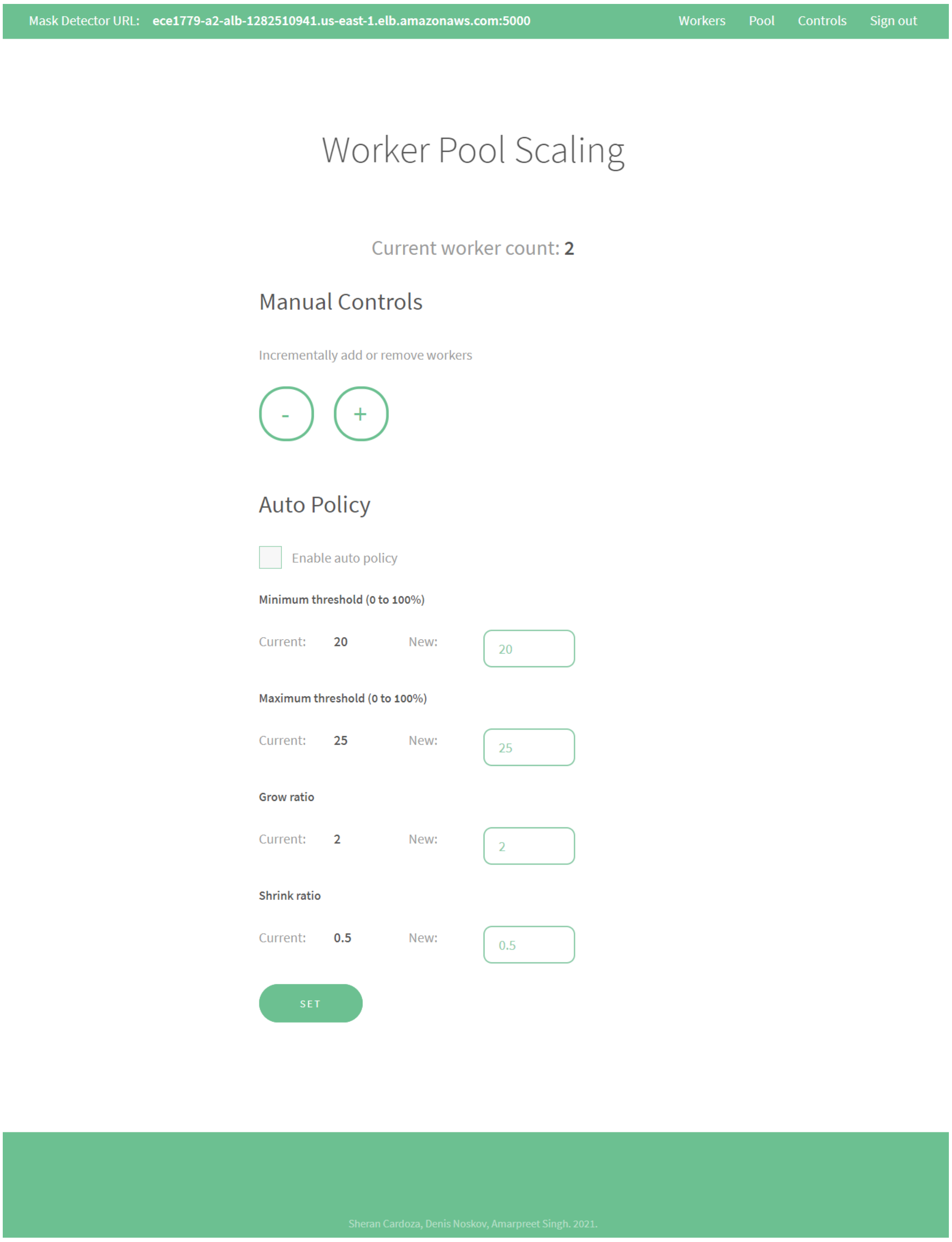
ID: **i-0b07772919f101f69** State: **healthy**

CPU utilization

HTTP requests

Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

Pool scaling policy page:



Mask Detector URL: ece1779-a2-alb-1282510941.us-east-1.elb.amazonaws.com:5000

Workers Pool Controls Sign out

Worker Pool Scaling

Current worker count: **2**

Manual Controls

Incrementally add or remove workers

- +

Auto Policy

☐ Enable auto policy

**Minimum threshold (0 to 100%)**

Current: 20 New: 20

**Maximum threshold (0 to 100%)**

Current: 25 New: 25

**Grow ratio**

Current: 2 New: 2

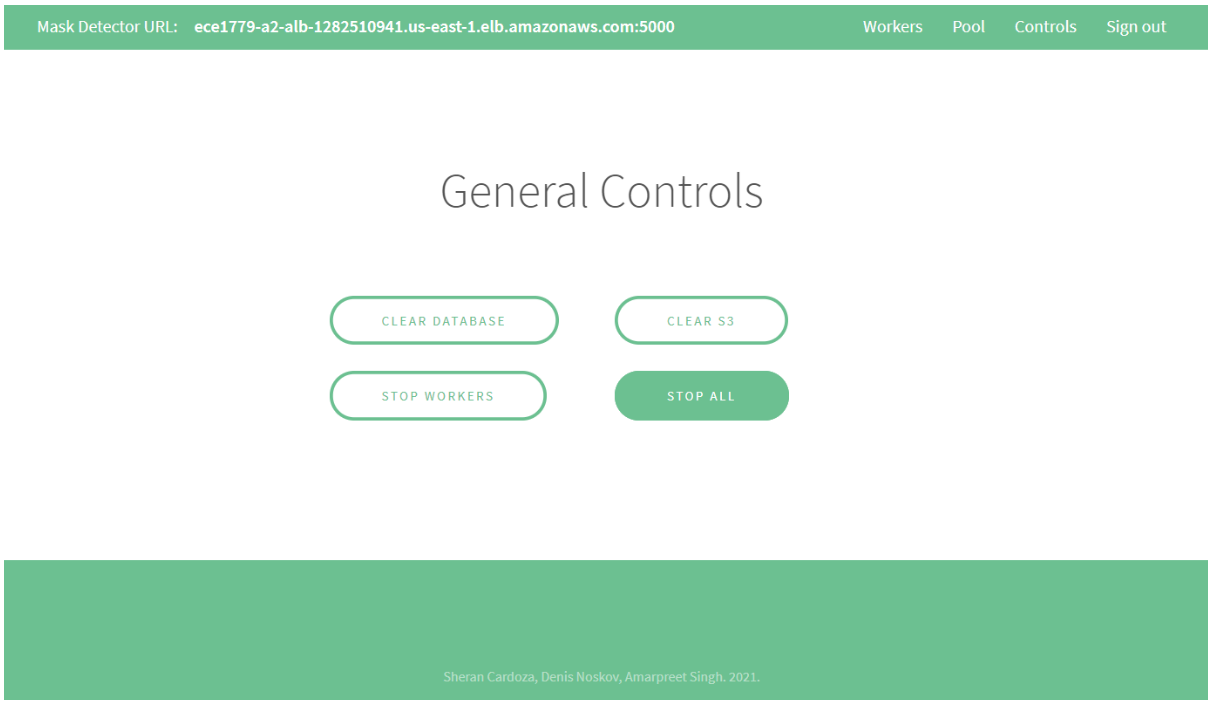
**Shrink ratio**

Current: 0.5 New: 0.5

SET

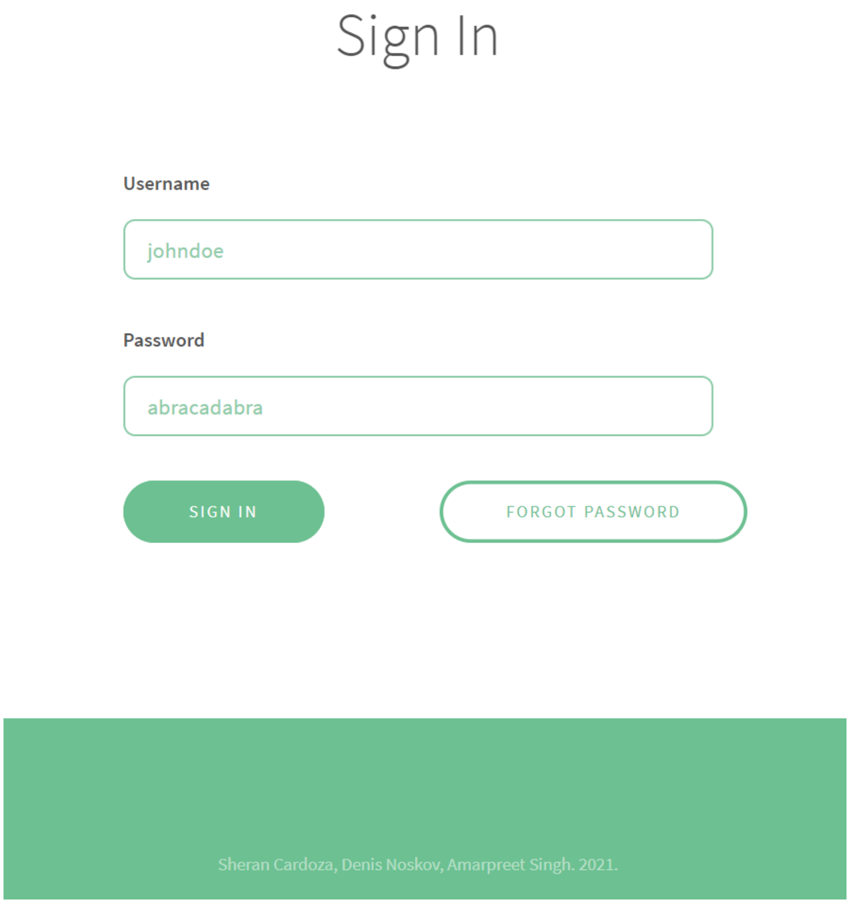
Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

General controls page:

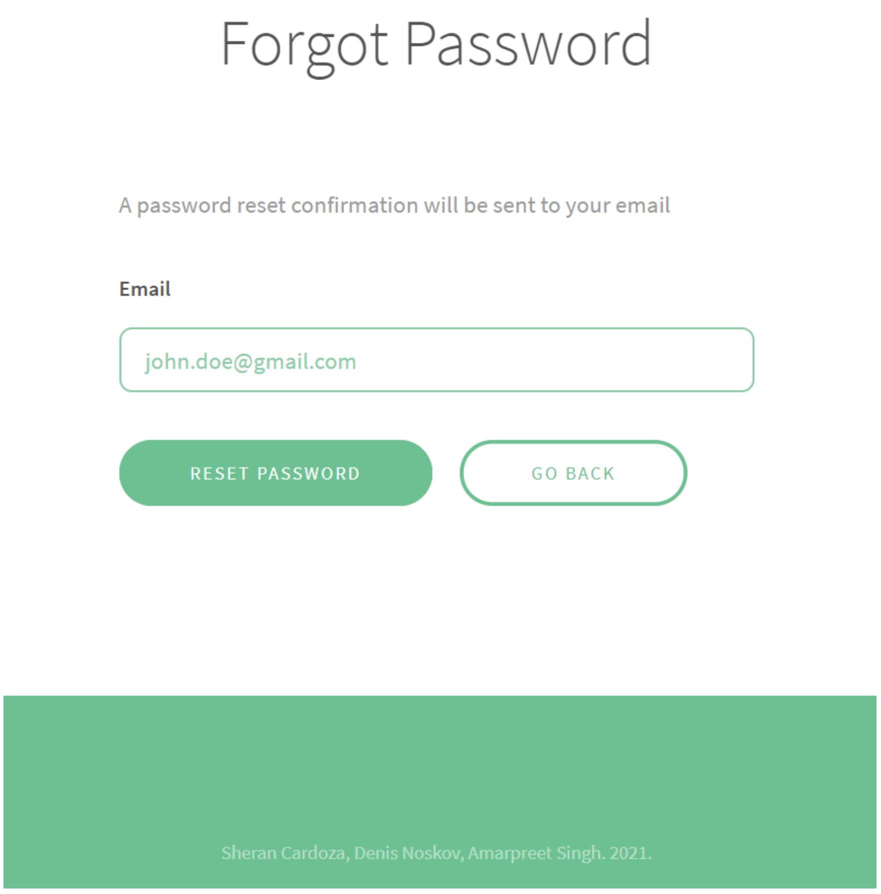

Mask Detector URL: ece1779-a2-alb-1282510941.us-east-1.elb.amazonaws.com:5000
Workers
Pool
Controls
Sign out
General Controls
CLEAR DATABASE
CLEAR S3
STOP WORKERS
STOP ALL
Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

### 4.2. User webapp

Login page:



Forgot password page:



Forgot Password

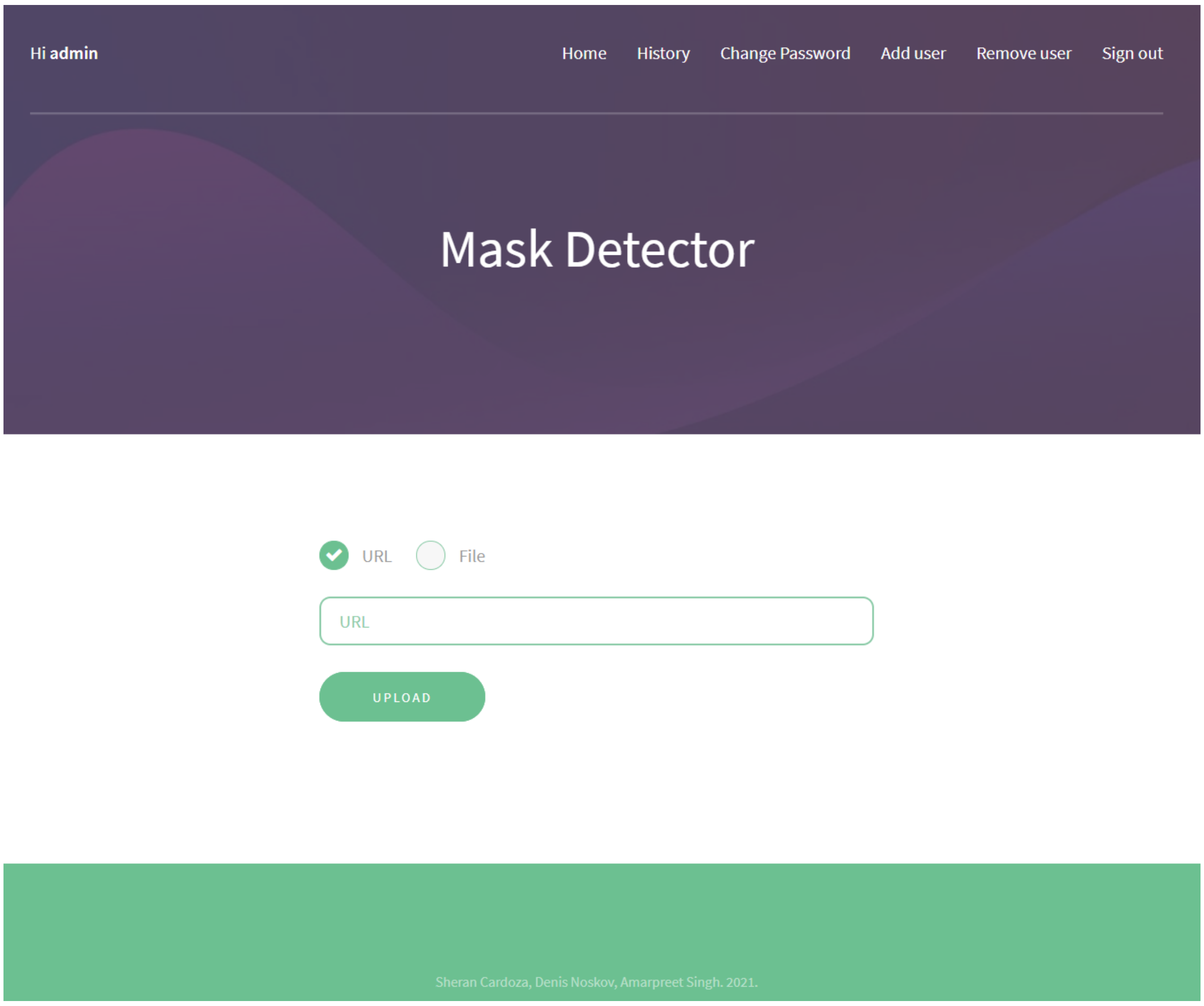
A password reset confirmation will be sent to your email

Email

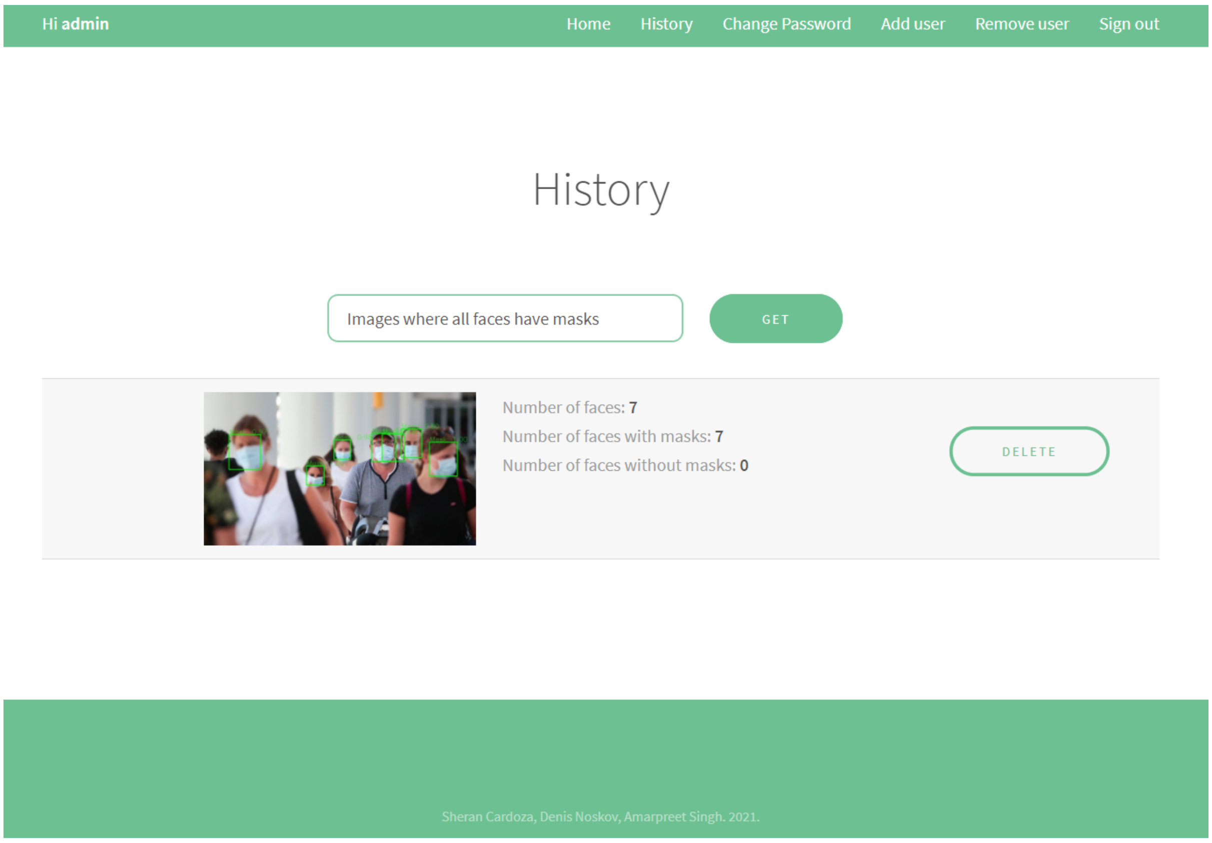
john.doe@gmail.com

Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

Home page:



History page:

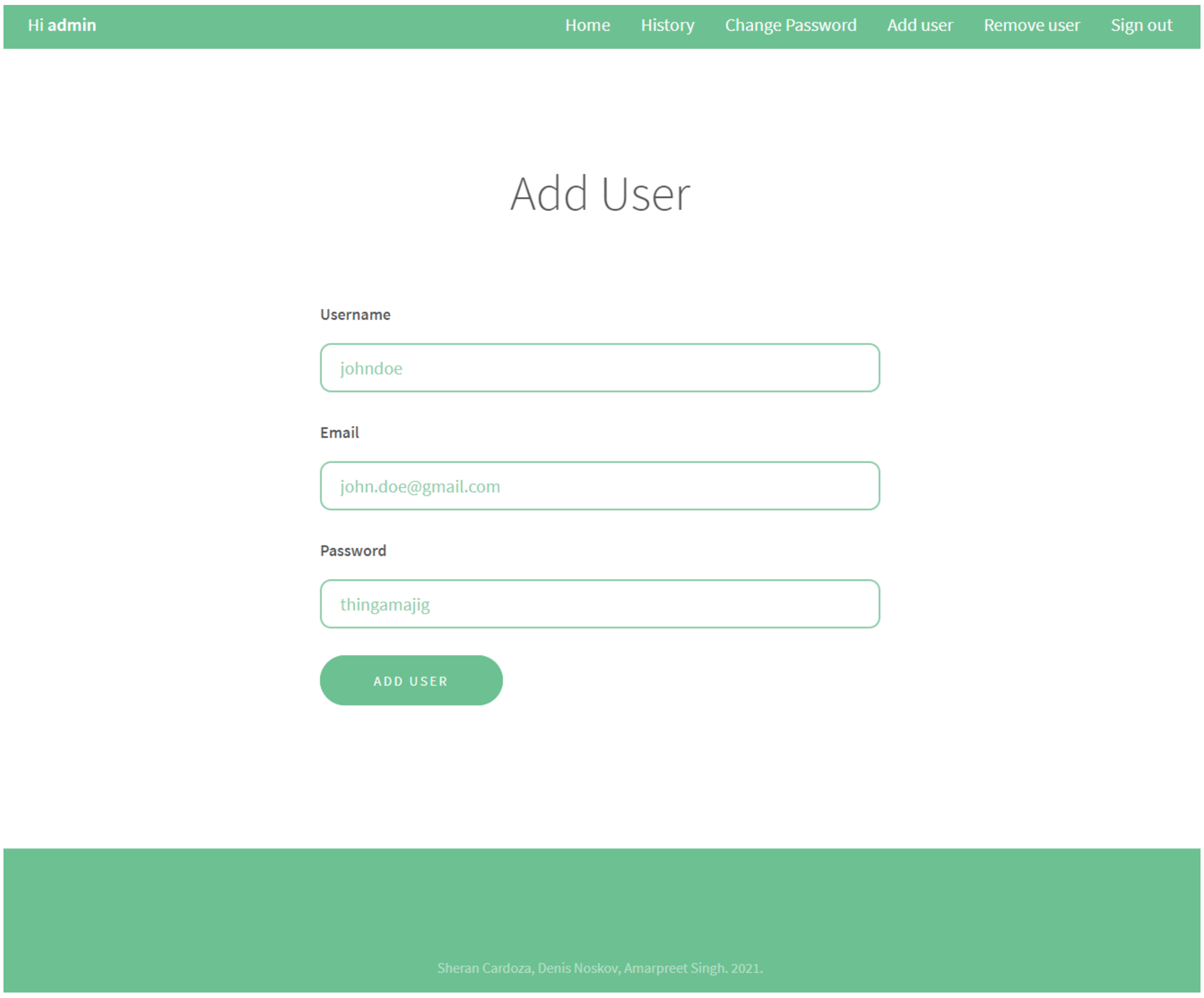


History

Number of faces: 7
Number of faces with masks: 7
Number of faces without masks: 0

Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

Add users page:



Add User

Username

johndoe

Email

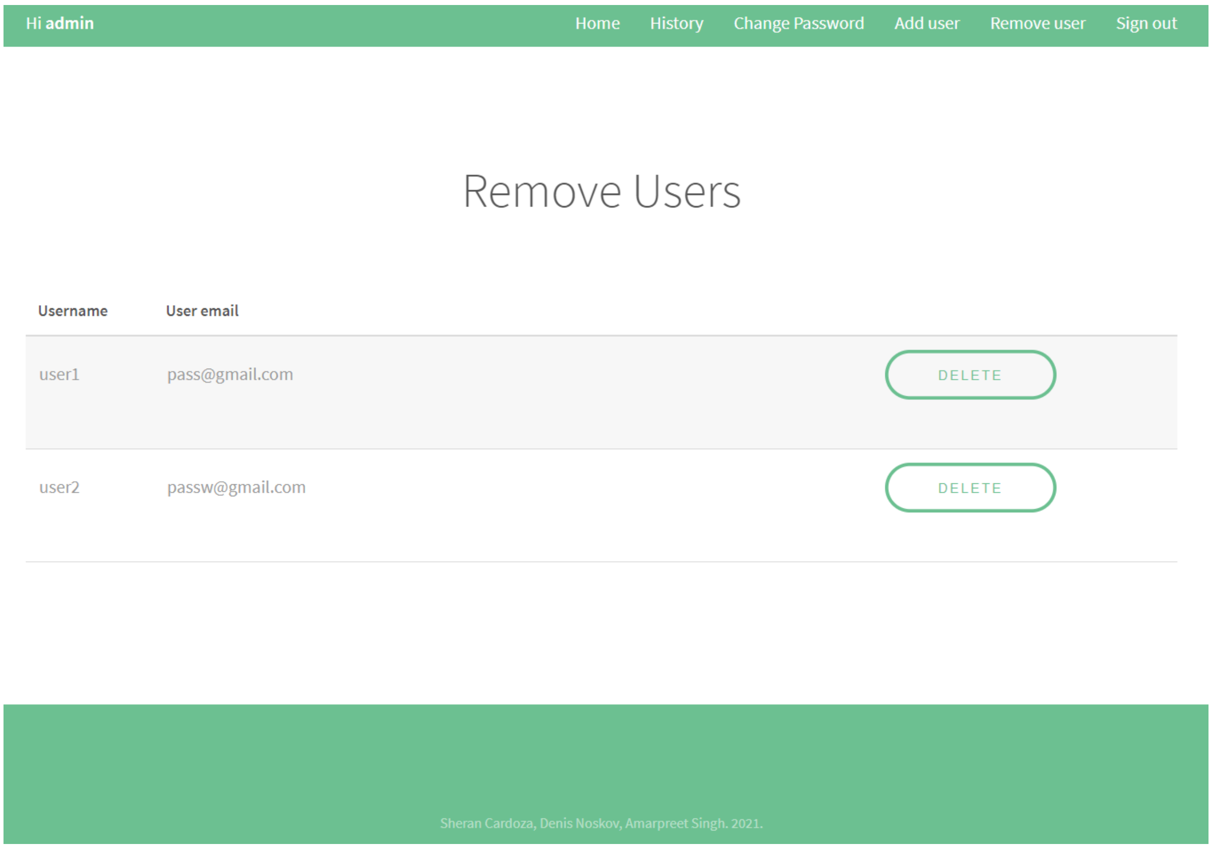
john.doe@gmail.com

Password

thingamajig

Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.

Remove users page:



Remove Users

| Username | User email | |
|---|---|---|
| user1 | pass@gmail.com | DELETE |
| user2 | passw@gmail.com | DELETE |

Sheran Cardoza, Denis Noskov, Amarpreet Singh. 2021.