**Case Study – Test Report**

**“Only Flights”**

A picture containing text, clipart

Description automatically generated

Project Members & Student Numbers:

* Kaloyan Andreev (4408020)
* Francisco Marcó (4467752)
* Sava Vasilev (4663438)

Elaboration Date: 10/01/2023

Case Study - Group 10

**Table of content**

[1. Preface 2](#_Toc124356171)

[a. Goal of the Test Report 2](#_Toc124356172)

[b. Way of working 2](#_Toc124356173)

[c. Scope of the testing 2](#_Toc124356174)

[d. Objective List 2](#_Toc124356175)

[2. Applications testing 4](#_Toc124356176)

[a. Testing Results 4](#_Toc124356177)

[b. Stress test result 4](#_Toc124356178)

[c. Testing reflection 7](#_Toc124356179)

[d. References 7](#_Toc124356180)

[e. Observations 7](#_Toc124356181)

[3. Infrastructure testing 8](#_Toc124356182)

[a. Testing Results 8](#_Toc124356183)

[b. Testing Reflection 9](#_Toc124356184)

[c. References 9](#_Toc124356185)

[d. Observations 9](#_Toc124356186)

[4. Security testing 10](#_Toc124356187)

[a. Testing Results 10](#_Toc124356188)

[b. Testing Reflections 11](#_Toc124356189)

[c. References 11](#_Toc124356190)

[d. Observations 11](#_Toc124356191)

# Preface

This document will detail the results of the Testing process that will undergo the Case Study Project. Alongside this document, we will follow the steps detailed in the Test Manual document so please refer it to it if its necessary to gain a deeper understanding of some of the goals. In the following chapters, we will illustrate which testing goals were achieved and a reflection, for each of the categories of the test manual.

## Goal of the Test Report

The main objective of this document is to determine how functional is our Project, by applying a series of tests. These tests were selected with the intention of conforming to our functional and non-functional requirements (outlined in the URS document of this Project) and to ensure that major issues are identified and fixed before delivering the project.

## Way of working

This document will outline which of the goals stated in the Test Manual were achieved. Each chapter presents a table with the status (achieved or unachieved) of each goal as of the writing of this document, and a brief explanation on how was the test performed. Furthermore, a written conclusion is included to reflect on the respective aspect of our Project.

## Scope of the testing

These test results come from three core aspects of the project: Application, Infrastructure and Security. We believe these to be the foundations of the project and as such we must ensure that their functioning is up to expectations and conforms with our requirements. In each chapter, we will go in-depth about our goals with the test and the components that must be tested.

## Objective List

Below is a list of the rough objectives we hope to fulfill by performing the Tests.

* **Infrastructure is cloud-based and will reduce costs as opposed to an onsite one and will also provide more worldwide coverage**
* **Web-application is delivered on a dedicated webserver**
* **Database server securely stores application data, user data, and logs**
* **Application delivers cloud infrastructure automation**
* **Ensure all core-concepts from the Course are implemented**
* **The provided infrastructure can hold an initial 100 users per day with an option to extend the current infrastructure in response to a scaling user flow.**

# Applications testing

## Testing Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Status | Test Operator | Test Description |
| Website is accessible through the internet | + | Developer team | Finding and opening the website through specified domain name |
| Log-in page is mandatory for users to access | + | Developer team | If the user does not have account, he or she cannot reach any webpage, except login/signup page. |
| Sign-up page is accessible for new users | + | Developer team | The new users are able to reach and create account through the signup page |
| Log-in authenticates users by storing their credentials in the Database | + | Developer team | Users’ credentials are stored safely in MySQL database |
| Passwords are hashed for security | + | Developer team | Passwords are saved as a string of random characters, the real password is not shown |
| User can book flights in the homepage | + | Developer team | The users are able to use the booking feature |
| Flight list shows real-time information by pulling data from an API | + | Developer team | After fulfilling the database table, it was checked manually by the user through skimming the data |
| Application stress test | + | Developer team | The webapp response normally as autoscale the Fargate tasks if needed |
| TrExecutor successfully runs code (connecting to AWS account + executing scripts) | + | Developer team | The automation app can build, update and destroy services successfully |

## Stress test result

To test how the webapp deals with the high number of requests and whether the autoscaling works, a simple stress test was performed. For this purpose, several tools were used:

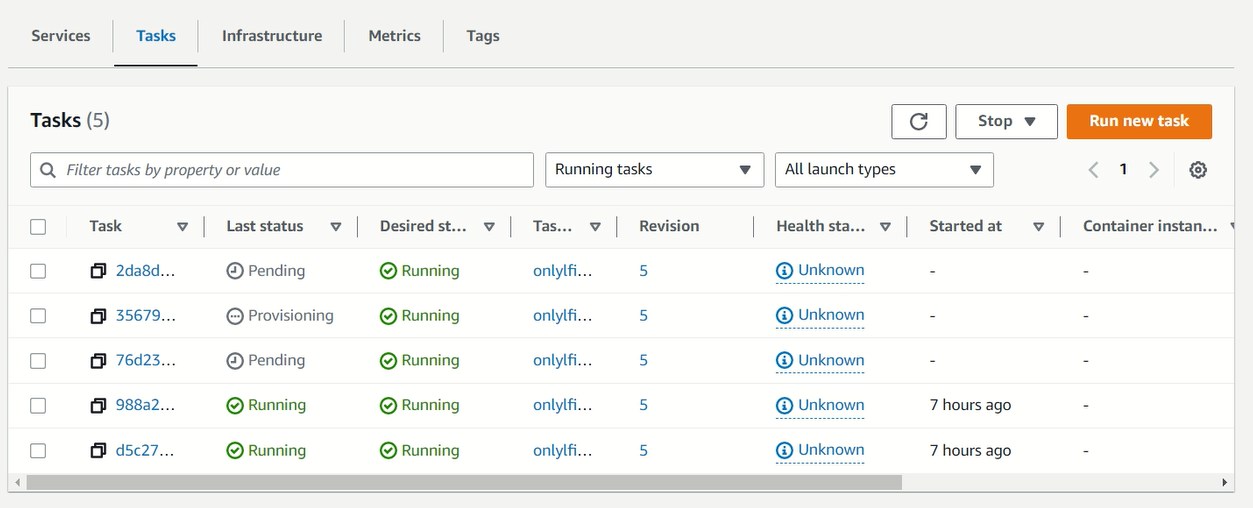
* Latest version of Python must be installed
* Locust library must be available in Python. If not download it using ‘pip’ command

After making sure the previous requirements are available, a pre-set python script was executed. Through this script, the target pages were configured as well as the type of requests that would be sent.

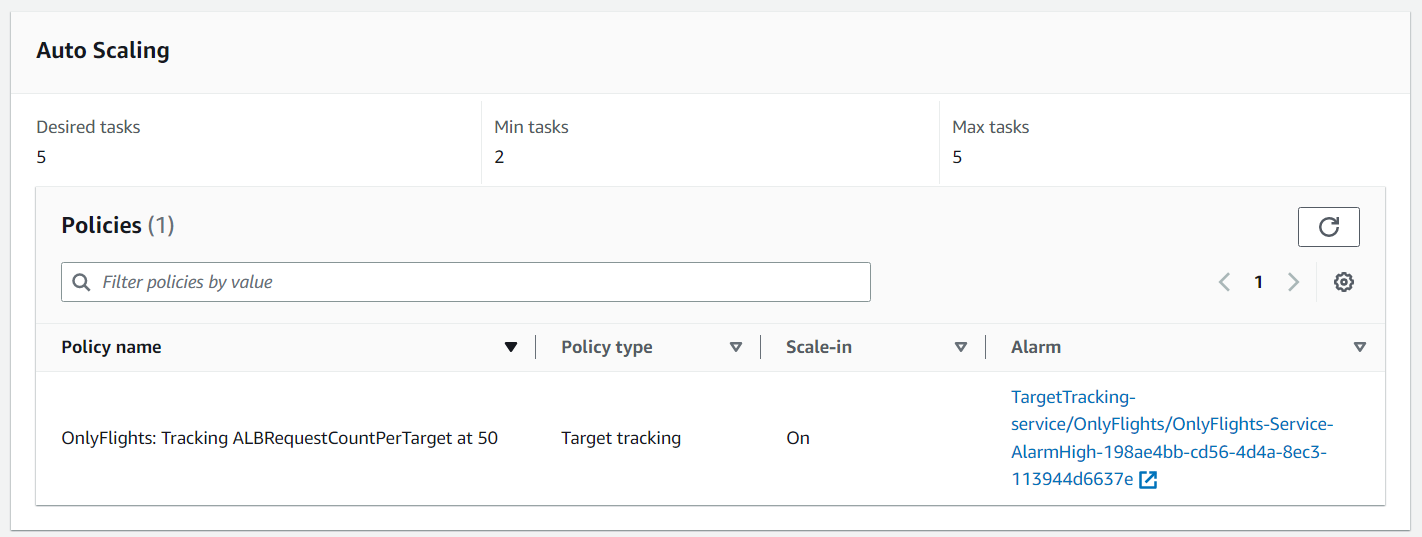
***Note***: You can take a look of the code by clicking the link below:

* [OnlyFlights - Testing/locust.py · main · Andreev,Kaloyan K.K. / Semester 3 - Case Study project · GitLab (fhict.nl)](https://git.fhict.nl/I476236/semester-3-case-study-project/-/blob/main/OnlyFlights%20-%20Testing/locust.py)

The next step was to open the Locust webpage which will be available on the localhost IP address and to fill the number of users that would be generated, the spawn rate and the URL of the target website. Then the requests would be sent gradually. After a few minutes, the new Fargate tasks would be started successfully.

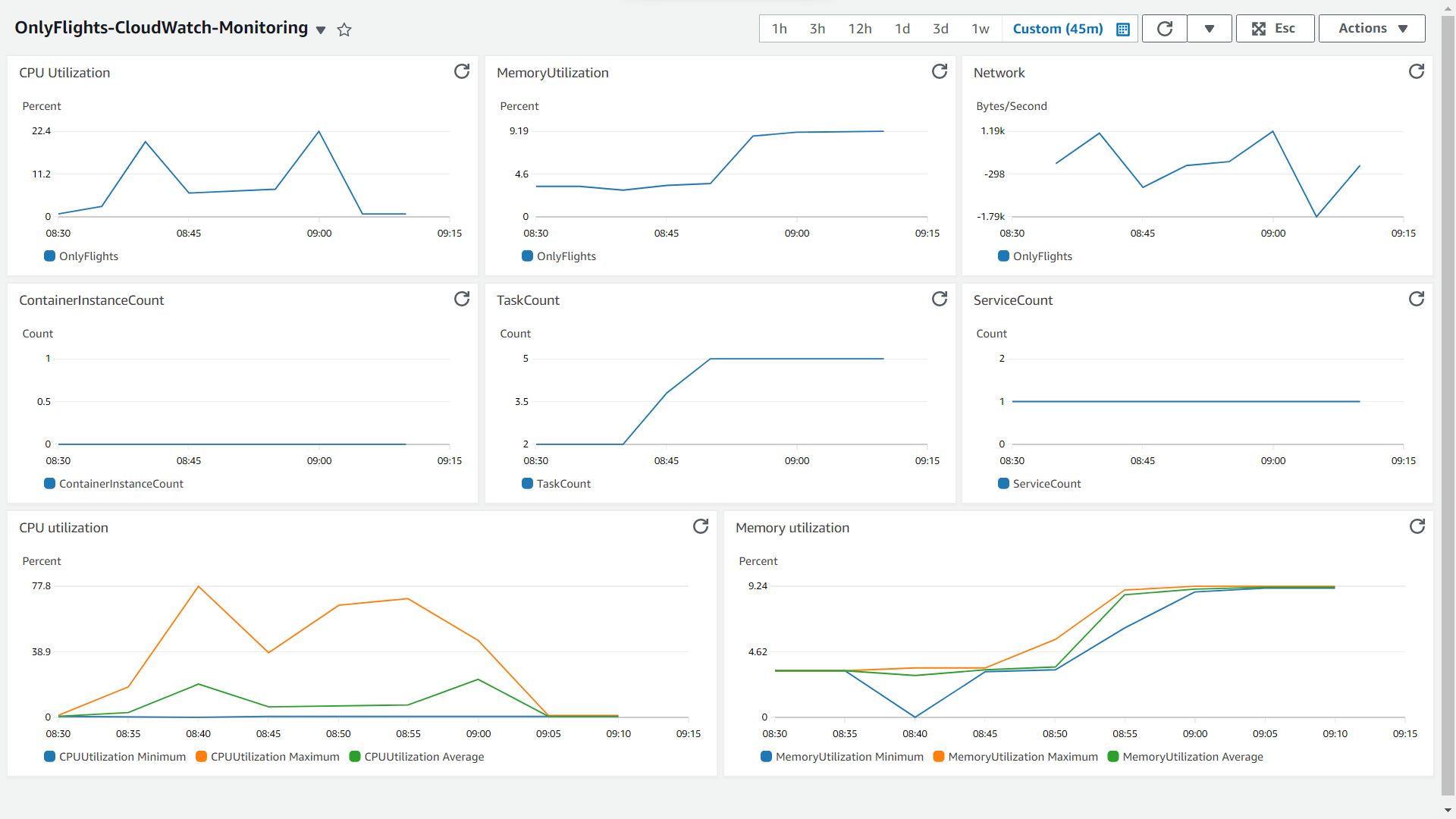


***Image:*** Newly started Fargate tasks



***Image:*** Autoscaling configuration

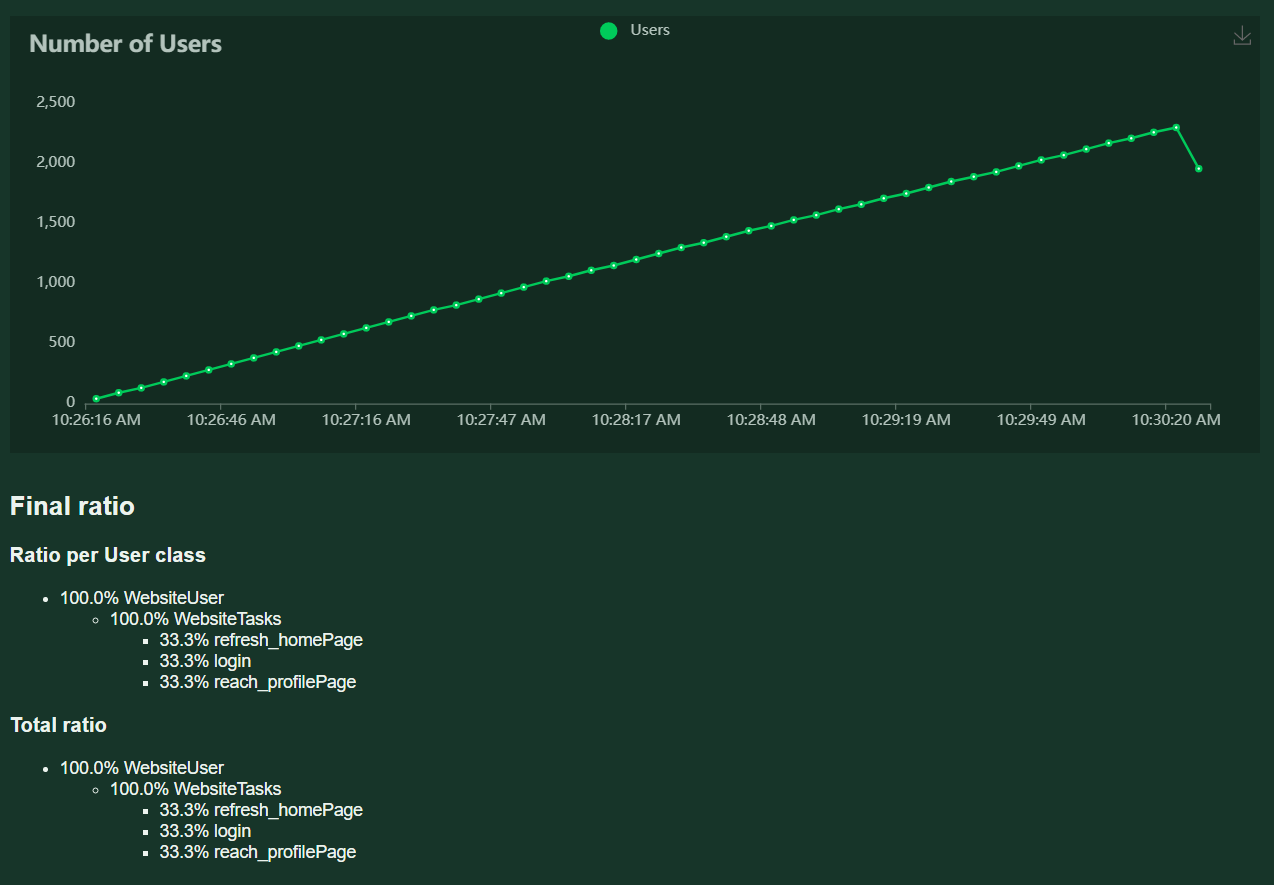
To track the changes, there were a CloudWatch dashboard that measures the utilization of the CPU, memory and the disk of the ECS cluster.



***Image:*** ECS cluster CloudWatch dashboard

Graphical user interface

Description automatically generatedFor more details, you can take a look of images of the generated Locust report:



## Testing reflection

All the performed tests were successful on our Applications. Additionally, metrics were collected for the resources hosted in AWS and the ECS Web Application. The metrics collected were deemed normal for the specified resources and their role in our Project, which included the following:

* **Memory utilization**
* **CPU usage**
* **Uptime**
* **Network traffic**
* **Network sockstat**
* **ECS Metrics:**
  + **NetworkRXBytes**
  + **Tasks**
    - **Running**
    - **Desired**
  + **CPU utilization**
  + **Memory utilization**

# Infrastructure testing

## Testing Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Status | Test Operator | Test Description |
| EC2 instances are connected to the internet and assigned to their proper respective VPCs and Subnets | + | Developer team | The instances must be able to ping ‘Google’ |
| ECR and ECS interaction is working as intended, and it is possible to push Docker images in the ECR to deploy new versions of the application | + | Developer team | After starting the ECS service, the Fargate tasks has to be tagged as healthy targets and they must be running successfully |
| Each EC2 instance fulfills its role and doesn’t provide any difficulties to the rest of the project. Every resource is relevant and there should not be any idle or unused instance. | + | Developer team | There are no extra instances in the cloud. There is role for each machine |
| Load Balancer is functioning and manages traffic according to expectations. | + | Developer team | The ALB distributes the traffic equally to the Fargate tasks. The listeners forward the traffic to the correct port |
| S3 storage interacts successfully with target resources and data is stored in a safely manner. | + | Developer team | After the test operation, the data is checked manually by the user |
| RDS instances interact successfully with target resources and data is stored in a safely manner. Costs are taken into account after the incident. | + | Developer team | After connecting to the RDS, the admin must be able to execute various queries.  In the webapp, the user must be able to search for different flights. |
| VPCs were created in an efficient manner, dividing and isolating resources by their role. Subnets are also implemented to allow HA and enforce security. | + | Developer team | After setting them up by Terraform, their configuration could be checked in AWS console |
| Terraform code successfully creates resources in AWS and configures them properly | + | Developer team | If there are no error messages after applying the changes in the terminal, everything is done properly |
| Ansible playbooks successfully creates resources in AWS and configures them properly | + | Developer team | After executing the playbooks, the user could check the status of the certain service by a command. More details can be found in the Technical Manual |
| Shell scripts successfully creates resources in AWS and configures them properly | + | Developer team | After executing the shell scripts, the user could check the status of the certain service by a command. More details can be found in the Technical Manual |
| Lambda functions are integrated and fulfill their intended purpose without affecting the infrastructure | + | Developer team | Simple tests are performed from the AWS console, to check if the functions work properly |
| Route53 correctly uses and manages the indicated domain | + | Developer team | If the website is reachable from the domain, the records are configured correctly |
| Monitoring is carried on relevant resources | + | Developer team | There is monitoring on every instance in the cloud. Monitoring tools as Prometheus, CloudWatch and Zabbix are used |

## Testing Reflection

The performed tests on our Infrastructure were carried out and provided us with good results. All items were covered by following the items showed in the table above. We managed to monitor AWS resources (as shown in the previous chapter of this document) and InfraLab hosted resources as well by making use of Zabbix. Additionally, on this environment, we performed Workload tests to ensure that relevant metrics were collected.

# Security testing

## Testing Results

|  |  |  |  |
| --- | --- | --- | --- |
| Test Name | Status | Test Operator | Test Description |
| Keys are stored safely and are accessible, each key is ideally not reused and has the proper permissions assigned to ensure SSH authentication. | - | Developer team | The private key pair was intended to be stored as a secret in the Secret Manager automatically through Terraform. Unfortunately, the prepared scripts did not work. The data from the key pair fail could not be retrieve and save automatically. |
| Relevant Secrets are confidentially stored and encrypted in some form | + | Developer team | The database credentials are retrieved and save as a secret in Secret Manager via Terraform. The password is generated randomly via Terraform resource. |
| VPC configuration ensures that the resources are grouped and managed in a way the ensures no risk for the project (i.e: Route Tables or VPN ensure a secure connection) | + | Developer team | The routing tables are configured through Terraform at the begging of the creation of the whole environment. The VPN tunnels are setup manually. |
| Security groups properly manage each instance according to their needs only (i.e: not giving unnecessary permissions that could prove a security risk) | + | Developer team | Only the ports that are needed are opened. |
| VPN ensures a safe, private tunnel to access the infrastructure. | - | Developer team | - |
| Off-site backups are provided for relevant data | - | Developer team | - |
| IAM roles, users and policies don’t imply a security risk (i.e: unnecessary permissions, try to follow CIA triangle) | + | Developer team | Only the users that are part of the group ‘Developers’ have full permission granted |
| HTTPS certificates work as intended | + | Developer team | After requesting SSL certificate and adding a new listener on port 443 in the ALB, each client is redirected to the HTTPS connection |

## Testing Reflections

This topic of our testing was the one we felt we didn’t do as well as expected. While the majority of tests were completed, we failed to deliver on some of the aspects of our Security as we were handling too many topics at the same time. These being Pair Key management, VPN connection, and Backups.

In future work, we will ensure a better planning at the beginning of the project to ensure that such relevant items are covered by the end of development.