**User Requirements Specifications**

**“Only Flights”**



Project Members & Student Numbers:

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

Elaboration Date: 14/09/2022

**Table of content**

[1. Introduction 2](#_Toc116312606)

[2. Objective 2](#_Toc116312607)

[3. Requirements 2](#_Toc116312608)

[3.1. Functional Requirements 2](#_Toc116312609)

[3.2. Non-Functional Requirements 3](#_Toc116312610)

[4. Supporting documents 5](#_Toc116312611)

[4.1. Website Wireframe 5](#_Toc116312612)

[4.2. User Personas 7](#_Toc116312613)

# Introduction

“OnlyFlights” is a case-study project that has as a purpose to simplify migrating to a cloud infrastructure for airlines. This user requirements specification (URS) document is created with the purpose of outlining the user requirements for the project “OnlyFlights”. Following this introduction the document will include in detail the scope of the project, all the requirements both functional and non-functional, a wireframe of the UI and a MoSCoW table.

# Objective

Our objective is to create a simplified and automated process which creates a cloud based infrastructure which a client can imediatelly migrate to with little to no downtime and very few expenses. The project should allow users to create infrastructures in AWS tailored to their needs and later manage it as they wish. Other websites and applications provide similar services but lack in some areas. Often times you have very limited control over your created infrastructure or cannot design it to your needs.. Our project aims to streamline this process and remove unnecessary elements along the way while adding much needed features.

# Requirements

## Functional Requirements

1. The project MUST be delivered with documentation including: Idea Pitch, Project Plan, Design Document, URS (This document), Backup Policy, Security Analysis and GDPR Analysis. Documents that will describe the idea of the project, it’s structure in detail and how to manage it.
2. A web-based UI for the end user MUST also be delivered operating on a dedicated webserver.
3. The end-product MUST have a cloud-based infrastructure which will reduce costs as opposed to a onsite one and will also provide more worldwide coverage.
4. The project MUST have a database server to store both critical for the application data, user and employee data and logs.
5. The project MUST have a secure infrastructure which ensures that sensitive data of the stakeholders is safely stored.
6. The provided infrastructure MUST be able to hold an initial 100 users per day with an option to extend the current infrastructure in response to a scaling user flow.
7. The infrastructure MUST provide automation for repetitive tasks.
8. The end-product SHOULD provide monitoring which will ensures availability and integrity.
9. The end-product SHOULD have a robust infrastructure therefore ensuring availability and integrity.

## Non-Functional Requirements

1. The infrastructure MUST include a secure VPN connection to the cloud environment.
2. An end-user MUST be able to book flights and then manage them.
3. An end-user MUST be able to manage his personal information in his account.
4. An admin user MUST be able to manage and monitor the state of the system on AWS.
5. Ansible MUST be used for the infrastructure creation.
6. AWS Lambda MUST be used.
7. AWS Step function MUST be used.
8. Infrastructure MUST be delivered with IAM users, group policies and key-pair authentication.
9. Files stored in S3 MUST have a server-side or customer key encryption.
10. Secrets management MUST be used for database instances.
11. The front-end instances MUST be in a separate subnet than the back-end instances to ensure security.
12. The webpage MUST have a certificate (HTTPS).
13. The components of the infrastructure MUST be automatically registered by the monitoring service.
14. The components MUST by deployed in multiple AZs.
15. File versioning MUST be provided.
16. Timely backups MUST be configured.
17. The costs SHOULD be thoroughly analyzed and justified.
18. An admin user SHOULD be able to access and monitor the system remotely.
19. A Docker container SHOULD be a part of the infrastructure.
20. AWS Gateway SHOULD be used.
21. NAT instance design SHOULD be used for back-end instances.
22. If more than one Docker containers are used they SHOULD communicate with each other.
23. The project SHOULD provide failover and a second database and webserver thus ensuring availability.
24. The front-end instances SHOULD be in a separate VPC than the back-end instances.
25. A Kubernetes environment COULD be used.
26. Ansible and Terraform COULD be used for the orchestration of repetitive and/or complex tasks.
27. The infrastructure COULD use 2 clouds, further ensuring availability and integrity.
28. Transit gateway COULD be used with BFP routing protocol to access back-end instances in different subnets.
29. The product COULD have regularly shipped to a monitoring service logs for observation and analysis.

# Supporting documents

## Diagram Description automatically generated with medium confidenceWebsite Wireframe

Graphical user interface

Description automatically generated

Table

Description automatically generated

## User Personas

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated