Airport Traffic Simulator

[User Requirements Specification]

Teacher:

Chung Kuah

ProCP

19/02/2018

Developers:Yoanna Borisova,Teodor Genov,Vladimir Katrandzhiev,Monika Kerulyte,Ignas Kybransas,Rostislav Tinchev

Contents

[Preface 3](#_Toc508310225)

[Introduction 3](#_Toc508310226)

[Processes 3](#_Toc508310227)

[Description of the processes 3](#_Toc508310228)

[Exceptional cases 3](#_Toc508310229)

[Non-Functional Requirements 4](#_Toc508310230)

[Requirements 5](#_Toc508310231)

[MoSCoW table 5](#_Toc508310232)

[Detailed requirements 6](#_Toc508310233)

[Must have 6](#_Toc508310234)

[Should have 6](#_Toc508310235)

[Could have 6](#_Toc508310236)

[Will not have 7](#_Toc508310237)

[User interface 7](#_Toc508310238)

[Use Cases 7](#_Toc508310239)

[Test Cases 7](#_Toc508310240)

# Preface

In this document, we will discuss the processes, the requirements, the design of the website and applications. It will also include decisions we made.

# Introduction

The expected result for “SIM Software Inc.” is a detailed simulated software of Airport Traffic, where users can interact with it by adding checkpoints through where the airplanes should fly before landing at an airport. It will also be suitable to benchmark different results on based input from the user and get more in-depth information on how many planes the simulation was able to land in specific time interval.

# Processes

## Description of the processes

Let’s consider Jonathan (User who just bought our Airport Traffic simulation software) is an ATF (Air Traffic Flow) manager responsible to find the best solution for Air Traffic in the newly renovated Eindhoven Airport.

**Running application, Choosing simulation outcome**. **(Set Up)** Once Jonathan executes the Airport Traffic simulation software he runs into the login page to check whether he is qualified to use the software or not. After he inputs his login and password Jonathan runs into user-friendly interface with a lot of options to choose from how he wants to interfere with the simulation that soon he is going to start. One of the options for the simulation is to choose from an airplane landing or taking off. After that Jonathan sees the weather control panel where he can adjust the temperature of the air, wind direction/strength and chance of precipitations for both snow and rain. He then only needs to add at least 1 checkpoint through where the airplane should go before landing/after taking off and Jonathan is going to be ready to start the simulation.

**Inside the simulation.** **(Execution)** When simulation has started, Jonathan still has some options to add, for example - unexpected events. For this reason, the simulation still contains the weather control panel - so the user could make emergencies on weather conditions and ability to change checkpoints - as an example to reroute airplanes from bad weather conditions.

**After simulation is complete. (Final)** After benchmarking is finished (it is either after Jonathan presses a finish button or after a specific time interval that simulation has to run), the user gets the results of how many planes landed/took off through the simulation hourly (simulation process is faster), how many flights had to be redirected because of weather conditions, software also creates an graph to show it visually.

## Exceptional cases

1. **A plane runs out of fuel.** If any of the planes by any chance runs out of fuel, they will land on the next checkpoint that is available after the last one they have passed.
   1. **The next checkpoint is unavailable.** If by any chance the next checkpoint is unavailable the plane will be redirected to any of the checkpoints that the nearest plane has passed.
   2. **If another plane’s checkpoint is unavailable.** The plane takes the shortest route considering the height and speed to the safest place to land.
2. **Two planes have the exact same checkpoint.** If two planes have a mutual checkpoint, one of the planes will increase its speed in order to avoid collision.
3. **Air Strip is taken for the landing.**  The plane must circle around the airport once to take a chance to land again.
4. **Air Strip is taken for the take off.** The plane must wait for the prior airplane to take off first and then he can ride into Air Strip for take off.
5. **Jonathan can’t add checkpoints.** The simulation is running. In order to be able to add/remove checkpoints the simulation must be stopped or paused (Whenever Jonathan presses an option to change checkpoints simulation must pause by itself immediately).

# Non-Functional Requirements

* Users must be only clients of the company, employees and higher ups.
* After launching the application there should be security check-up for employee to login.
* Application must contain user-friendly design with big, noticeable buttons, sliders.
* Airport Traffic Simulation software must maintain stability throughout the simulation.
* Before the release of the simulation, software must be tested and contain satisfactory results.
* The cost of the software development is going to be concealed.

# Requirements

# MoSCoW table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement | Must | Should | Could | Won’t |
| Simulated airport traffic | x |  |  |  |
| Benchmark-able outcome of simulation | x |  |  |  |
| Simulate arising of emergencies | x |  |  |  |
| Queue up traffic efficiently |  | x |  |  |
| Handling the closure of the airstrips |  |  | x |  |
| Managing the traffic related to the airstrips | x |  |  |  |
| Simulate severe weather conditions |  |  | x |  |
| Indicate safe speed in bad weather conditions |  |  | x |  |
| Indicate safe direction in bad weather conditions |  |  | x |  |
| Possibility to implement different airfields |  |  | x |  |
| Changing the number of airstrips |  |  | x |  |
| Changing the direction of airstrip |  | x |  |  |
| Reroute traffic to other airfields |  | x |  |  |
| Put planes on hold |  | x |  |  |
| The program will figure out a consequence in which the planes should take off and land on its own | x |  |  |  |
| The program will be able to guide the traffic safely | x |  |  |  |
| Previously handled flights should be able to be saved to a file |  |  | x |  |
| Previously handled flights should be able to be loaded from a file |  |  | x |  |
| Current position, routes, etc. of all relevant flights should be saved to a file |  |  | x |  |
| Current position, routes, etc. of all relevant flights should be loaded from a file |  |  | x |  |
| Emergencies will make it harder for planes to land/ take of |  | x |  |  |
| Average number of planes redirected in an hour |  | x |  |  |
| Average number of planes lifted/landed in an hour |  | x |  |  |
| Interaction with the air traffic by making checkpoints through where the planes should fly | x |  |  |  |
| Random planes entering/leaving air space are going to be generated, displayed and controlled |  | x |  |  |
| Application updates after release |  |  |  | x |

# Detailed requirements

## Must have

* Simulated airport traffic:
  + Random flights entering/leaving air space is going to be generated, displayed and controlled;
  + Interaction with the air traffic by making checkpoints through where the planes should fly;
* Benchmark-able outcome of simulation:
  + Average number of planes lifted/landed in an hour;
  + Average number of planes redirected in an hour;
* Simulate arising of emergencies:
  + Prompted by the user or randomly emergencies should arrive;
  + Emergencies will make it harder for planes to land/ take off;
  + It will result in slower landing/take off speeds or reroutings, which will make changes to the final benchmark outcome;
* Save and load simulation states:
  + Current position, routes, etc. of all relevant flights should be saved;
  + Previously handled flights should be able to be saved to a file;
* Queue up traffic efficiently:
  + The program will be able to guide the traffic safely;
  + The program will figure out a consequence in which the planes should take off and land on its own;

## Should have

* Handling the closure of an air strip and managing the traffic related to that strip adequately without breaking everything else:
* Reroute traffic to other airfields;
* Put planes on hold;
* In case of closed airstrip the program shouldn’t break the whole airspace, it should react accordingly in a way that the air traffic flow should be disturbed as little as possible;

## Could have

* Simulate severe weather conditions:
  + Inputted by user or randomly generated weather;
  + Ability to change chance of rain/snow, wind strength and direction, air temperature;
* Indicate safe speed/direction in bad weather conditions
  + Low visibility could lead to different speed/ altitude requirements for planes;
  + Strong winds could implement the change the direction of the planes in which they should take off or land;
* Possibility to implement different airfields:
  + Changing the number of strips;
  + Changing the direction of strip;

## Will not have

* Any kind of support or maintenance after the release of the product.

# User interface

This section will be updated in the future.

# Use Cases

Precondition  
The following preconditions are valid for every use-case mentioned under:

1. The application is running.

If there is a precondition unique for the simulation application, it will be mentioned in the specific use-case.

## Use Cases

Name: Saving data to a file

Goal: User saves the current simulation state to a file

Actor: User

Pre: The user has logged in; a simulation is running.

MSS:

1. User selects to save the current simulation state to a file
2. A save file dialog is opened
3. User inputs a file name
4. User selects the desired location for the saving of the file
5. User selects to save the file
6. File is saved
7. The save file dialog is closed and the application returns to its previous state

Extensions:

3-5a: User selects to discard the changes

.1 The simulation returns to its previous state

6a: A problem with the saving occurs and the file is not saved

.1 An error message is displayed, informing the user of this

.2 The user dismisses the message

.3 Return to MSS 2

Name: Loading data

Goal: A previous simulation is reloaded from a file saved on the machine

Actor: User

Pre: Тhe user has logged in

MSS:

1. User selects to load a file
2. System displays a file selection screen
3. User finds the desired file and selects it
4. The file is successfully loaded up
5. The file selection screen is closed and the old simulation is reinitiated

Extensions:

3a: User selects to cancel the file loading

.1 The file selection screen is closed, the program returns to its previous state

4a: There is a problem with the file and it cannot be opened

.1 An error message is displayed, informing the user of this

.2 The user dismisses the message

.3 Return to MSS 2

Name: Weather conditions manipulation

Goal: User changes the weather conditions while the simulation is running

Actor: User

Pre: Тhe user has logged in, a simulation is running.

MSS:

1. User inputs the desired weather conditions with the provided interface
2. User selects to apply the changes
3. The system takes in the new weather and changes the routing of airplanes, if necessary

Extensions:

2a: User selects to discard the changes

.1 The simulation resumes without any

Name: Log in

Goal: User logs in the system and gains access to the simulator’s main page.

Actor: User

Pre: Тhe log in screen is being displayed.

MSS:

1. User inputs his credentials (username/email and password) in the appropriate fields
2. User selects option to log in
3. System acknowledges the user’s details are correct and logs him in
4. User is redirected to the simulation screen

Extensions:

3a: System checks the user’s details and they are incorrect

.1 An error message, informing the user of his incorrect details is displayed

.2 Return to step MSS 1

Name: Edit checkpoints

Goal: User adds new checkpoints to the map or removes already existing ones, the system accepts the changes and adapts the routing appropriately.

Actor: User

Pre: The user has logged in and a simulation is running.

MSS:

1. User selects the option to enter checkpoint edit mode
2. The simulation pauses and the checkpoint edit mode is entered
3. User selects where would they like new checkpoints by clicking on the spot on the map where the checkpoints should be
4. A new checkpoint is displayed
5. User selects to exit checkpoint edit mode
6. Application accepts the changes with the checkpoints, adapts the routing to the new environment and proceeds with the simulation

Extensions:

3a: User removes the checkpoints he wishes gone by clicking on them

.1 The checkpoints disappear from the map

.2 Proceed to MSS 5

3a.2a: Proceed to MSS 2

3a.2b: Less than the minimum number of checkpoints, needed for the algorithm to work are left on the map; the system displays a warning message, informing the user of that

.1 Proceed to MSS 2

Name: Add airplanes to the air space

Goal: After a prompt by the user, a new airplane is generated and enters the airspace.

Actor: User

Pre: The user has logged in and a simulation is running.

MSS:

1. User selects the option to add a new airplane to the air space
2. A set of controls regarding the airplane’s attributes such as position, heading, speed, altitude, etc. are displayed
3. With those controls the users sets the airplane’s attributes
4. User selects to “release” the aircraft
5. The airplane is generated and shown on the verge of the airspace with all the properties, defined by the user

Extensions:

4.a The user has inputted something incorrectly

.1 The system displays a warning message and asks the user to correct his input

.2 Return to MSS 3

4.b The user decides to discard the aircraft

.1 The set of controls for new airplane definition are dismissed

.2 Simulation proceeds without a change

# Test Cases

This section will be updated in the future.