**Project Report**

**Fly High – Airline Management System**

**Created by:**

|  |  |
| --- | --- |
| **Cristina Ailoaei (266543)** | **Dragoș Sîrbu (266500)** |
|  |  |
| **Michał Jurewicz (266892)** | **Michał Podgórni (267128)** |
|  |  |

**Supervisors:**

**Ib Havn**

**Jens Cramer Alkjærsig**

**Mona Wendel Andersen**

**Steffen Vissing Andersen**

**[Name and logo of educational institution]**

**[Logo of companies included]**

**[Number of characters]**

**Information and Communication Technology** **Engineering**

**2nd Semester**

**IT-SEP2Y-A17**

**Group I**

**07.07.2018**

****

**Table of content**

[List of figures and tables v](#_Toc515726015)

[Executive summary vi](#_Toc515726016)

[1 Introduction 1](#_Toc515726017)

[2 Theory/literature survey 2](#_Toc515726018)

[3 Methods 3](#_Toc515726019)

[3.1 Analysis 4](#_Toc515726020)

[3.1.1 Requirements 4](#_Toc515726021)

[3.1.2 Use case diagram 6](#_Toc515726022)

[3.1.3 Use case descriptions 7](#_Toc515726023)

[3.1.4 Activity diagrams 8](#_Toc515726024)

[3.1.5 Analysis class diagram 9](#_Toc515726025)

[3.2 Design 9](#_Toc515726026)

[3.2.1 Design class diagram 9](#_Toc515726027)

[3.2.2 Sequence diagrams 9](#_Toc515726028)

[3.2.3 GUI design 9](#_Toc515726029)

[3.3 Implementation 9](#_Toc515726030)

[3.4 Testing 9](#_Toc515726031)

[3.4.1 Test cases 9](#_Toc515726032)

[3.4.2 JUnit testing 11](#_Toc515726033)

[4 Results/findings and Discussion 11](#_Toc515726034)

[4.1 Results 11](#_Toc515726035)

[4.2 Discussion 11](#_Toc515726036)

[5 Conclusions 12](#_Toc515726037)

[6 Sources of information 13](#_Toc515726038)

Appendices

# List of figures and tables

Figure 1 – Use Case Diagram x

Figure 2 – Book a flight use case description x

Figure 3 – Book a flight activity diagram x

Figure 4 - x

Figure 5 - x

Figure 6 - x

Figure 7 - x

Figure 8 - x

Figure 9 - x

Figure 10 - x

Figure 11 - x

Figure 12 - x

Figure 13 - x

Figure 14 - x

Figure 15 - x

# Executive summary

# Introduction

# Theory/literature survey

# Methods

## Analysis

### Requirements

#### Functional requirements

1. An administrator should be able to add airports to the system. While adding a new airport, the administrator has to specify the code, name, city, postcode, country, number of gates.
2. An administrator should be able to add airplanes to the system. While adding a new plane, the administrator has to specify the number, model, number of seats.
3. An administrator should be able to add crew members to the system. While adding a new crew member, the administrator has to specify the name, position, address, birthdate, id, phone number, e-mail.
4. An administrator should be able to add flights to the system. While adding a new flight, the administrator has to specify the number, departure time, arrival time, departure place, arrival place, plane, crew, price.
5. A customer booking a flight should specify all of the following: name, birthdate, nationality, type of ID, ID number, expiration date.
6. An administrator should be able to delete data from the system.
7. A head administrator should be able to cancel flights.
8. An administrator should be able to change data for club members, crew, flights, airplanes and airports.
9. An administrator should be able to select date/time range for flights in order to get flights in a specified range.
10. An administrator should be able to select cities for flights in order to get flights from/to the specified cities.
11. A customer should be able to choose a seat number, luggage size, payment method in order to book a ticket.
12. A customer should be able to select departure and destination airport and the departure and return date (or departure only) for flights in order to get the available flights.
13. An administrator should be able to get a list of all flights and club members.
14. An administrator should be able to set the annual fee for club members.
15. A customer should receive the ticket via email.
16. A customer should be able to become a club member in order to get discounts.
17. A club member should be able to search only for cheap flights from his/her city.
18. A customer should be able to subscribe to the newsletter in order to receive new information regarding flights and offers via email.
19. An administrator should be able to log in the system in order to manage data.
20. A head administrator should be able to see the profiles of all administrators.
21. A head administrator should be able to create or delete an administrator account in order to ease the management of accounts.

#### Non-functional requirements

1. The system has to use the client-server architecture.
2. The system has to store persistent data using a database.
3. The system has to have a GUI.
4. The system has to provide a log in.
5. The system has to be implemented in Java.
6. The system and the system development process have to be documented.

### Use case diagram

Figure 1 - Use case diagram

The use case shown above (Figure 1) presents all functional feature that every user of  
the FlyHigh application can perform. The use cases are following:

* **Add an element** – The administrator or the head administrator can add an airplane, airport, crew member or flight into the system.
* **Find an element** – The administrator or the head administrator can search for an existing airplane, airport, crew member or flight in the system.
* **Edit an element** – The administrator or the head administrator can edit the data of an airplane, airport, crew member or flight.
* **Delete an element** – The head administrator can delete an airplane, airport or crew member from the system.
* **Cancel a flight** – The head administrator can cancel a flight.
* **Find a flight** – The customer can search for an existing flight in the system.
* **Book a flight** – The customer can book an existing flight.

### Use case descriptions

|  |  |
| --- | --- |
| UseCase | Book a flight |
| Summary | A customer books a flight |
| Actor | Customer |
| Precondition | None. |
| Postcondition | The flight becomes booked, the changes are stores in the database. |
| Base Sequence | 1. The person goes through find a flight use case.  2. The person enters all the required personal data: name, birthdate, nationality, type of ID, ID number, expiration date, seat number, size of luggage, method of payment.  3. The person confirms the decision to book the given flight.  4. If one or more of the entered data is not valid then go to step 2 else the decision is confirmed and the given flight becomes booked, changes are stored in the database, person is redirected to another site in order to make a payment and the use case ends. |
| Branch Sequence |  |
| Exception Sequence | The entered data could not be valid:  4 as base sequence  The system informs that the entered data is not valid |
| Sub UseCase | Find a flight |
| Note |  |

Figure 2 - Book a flight use case description

### Activity diagrams

Figure 3 - Booking a flight activity diagram

One of the crucial functionalities, booking a flight, works as follows:

First the customer initiates an option to search for a flight.

Then all the necessary data (departure place, arrival place, departure time, arrival time) is entered.

If the system cannot find a flight with given details, it displays an appropriate message and asks the customer to fill the new data.

If the system can find a flight with given details, it is displayed for the customer.

In the last step the customer is asked to fill all his personal details (name, birthdate, nationality, type of id, id number, id expiration date) and flight details (seat number, luggage size, payment method).

If any of the entered data is not correct, the system displays an appropriate message and asks to fill the new data.

If all the entered data is correct, the system makes a reservation which is stored in the system and the customer is informed about it. The flow ends.

### Analysis class diagram

## Design

### Design class diagram

### Sequence diagrams

### GUI design

## Implementation

## Testing

### Test cases

In order to get a complete list of all functionalities of the system together with all possible combinations of events that can go in a different way than they were meant, the project team developed a list of test cases that show it. They are based on the requirements and use cases of the system and their main purpose is to make sure that any possible scenario will not be forgotten. Owing to it the system should not have any situations it would not know how to handle.



Figure 4 - Book a flight test case

### JUnit testing

# Results/findings and Discussion

## Results

## Discussion

# Conclusions

# Sources of information

**Appendices**