Software Requirements Specification

for

AirLegance

**Version 1.0 approved**

**Prepared by :  
Udrea Robert-Mihai  
Vasilescu Alexandru-Cristian  
Vaduva Andrei-Alexandru**

**Faculty of Automatics, Computers and Electronics**

**3.13.2023**

**Table of Contents**

[**1. Introduction 1**](#)

[1.1 Purpose 1](#_3znysh7)

[**1.2 Document Conventions 1**](#)

[**1.3 Intended Audience and Reading Suggestions 1**](#)

[**1.4 Product Scope 1**](#)

[**1.5 References 2**](#)

[**2. Overall Description 2**](#)

[2.1 Product Perspective 2](#_2s8eyo1)

[**2.2 Product Functions 2**](#)

[**2.3 User Classes and Characteristics 2**](#)

[**2.4 Operating Environment 3**](#)

[**2.5 Design and Implementation Constraints 3**](#)

[**2.6 User Documentation 3**](#)

[**2.7 Assumptions and Dependencies 3**](#)

[**3. External Interface Requirements 4**](#_2jxsxqh)

[**3.1 User Interfaces 4**](#)

[**3.2 Hardware Interfaces 6**](#)

[**3.3 Software Interfaces 6**](#)

[**3.4 Communications Interfaces 6**](#)

[**4. System Features 6**](#)

[**4.1 Flight Information 6**](#)

[4.2 Register Functionality 7](#_6fl9e2qam9x2)

[4.3 Login Functionality 7](#_t9x9fo6irhtz)

[4.4 User Profile Information 8](#_kxwkk4h2hy46)

[4.5 User Profile as Employee 9](#_gyey2mium1ri)

[4.6 Contact Us Functionality 9](#_jajjf2w4qflh)

[**5. Other Nonfunctional Requirements 10**](#_3whwml4)

[5.1 Performance Requirements 10](#_2bn6wsx)

[**5.2 Safety Requirements 10**](#)

[**5.3 Security Requirements 10**](#)

[**5.4 Software Quality Attributes 10**](#)

[**5.5 Business Rules 11**](#)

[**6. Other Requirements 11**](#)

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
| Udrea Robert, Vaduva Andrei,  Vasilescu Alexandru | 3/22/2023 | Created document | 1.0 |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document describes the specifications for the Automated Management of

flights and customers of an airline company.

## Document Conventions

* QA = Quality Assurance
* UI = User
* SQL = Standard query language
* T-SQL = Transact-SQL
* MSSMS = Microsoft SQL server management
* IDE = Integrated development environment
* ORM = Object-relational mapper
* API = Application programmable interface
* EF = Entity Framework

## Intended Audience and Reading Suggestions

This document will mainly be used by the development team which is composed of software developers and QA and by the person in charge of it ( project manager ).

## Product Scope

The objective of this project is to design and develop a software system that automates flight and customer management processes for an airline company. This software solution will enable the airline to collect data related to each performed flight, including take-off airports, landing, and intermediate points. The system will also manage passenger data, generating reports that can be easily accessed via a web portal.

Our software system will offer customers the ability to query the system and view important information related to their reservations, including their flight schedule and ticketing details. Additionally, our system will manage the document flow for all reservations, ensuring a seamless and efficient booking experience for customers and airline staff. The system's user-friendly interface will simplify the process of managing flight schedules, passenger data, and reservation documents, allowing staff to focus on delivering excellent customer service.

In summary, our software system will revolutionize the flight and customer management processes for an airline company, providing a reliable and efficient solution for data collection, passenger data management, report generation, web portal information sharing, and system querying.

## References

* [UI Style Guides](https://mui.com/material-ui/getting-started/overview/)
* [Javascript Code Conventions](https://google.github.io/styleguide/jsguide.html)

# Overall Description

## Product Perspective

From a product perspective, the project aims to provide a comprehensive software solution for automating an airline company's flight and customer management process. The software system will streamline various operations and processes, including flight schedules, ticket booking, check-in, baggage handling, and customer data management. The system will provide a user-friendly interface for both airline staff and customers, allowing them to interact with the software effortlessly and efficiently.

## Product Functions

* Ability to explore and look up available flights
* Option to buy tickets for single or multiple flights
* Access to review booking history
* Capability to process payments for bookings
* Profile management, including contact details
* Ability to set up flight schedules and determine pricing
* View passenger and flight information
* Reservation management, including adjustments or cancellations
* Communication with passengers through email or phone for support or updates
* User account management and role assignment

## User Classes and Characteristics

Normal User:

* Can browse and search for available flights
* Can purchase tickets for one or more flights
* Can view their booking history
* Can make payments for reservations
* Can manage their profile information, such as contact details and payment preferences

Employee:

* Can configure flights schedules and pricing
* Has access to additional functionality compared to normal users
* Can view passenger and flight data
* Can manage reservations, including making changes or cancellations
* Can communicate with passengers via email or phone to provide support or information

Admin:

* Has access to all system functionality and data
* Can manage user accounts and can assign roles (employee)

Overall, our system caters to the needs of normal users, employees, and administrators, offering different levels of access and functionality to ensure a smooth and efficient experience for all.

## Operating Environment

* Windows 10,11
* Google Chrome (111.0.5563.115)
* Internet Access

## Design and Implementation Constraints

* IDEs:
  + Visual Studio 2022
  + Visual Studio Code 1.76
  + MSSMS 18
* Languages:
  + C# 11 using ASP .NET Core 7
  + React 18.2.0
  + T-SQL

## User Documentation

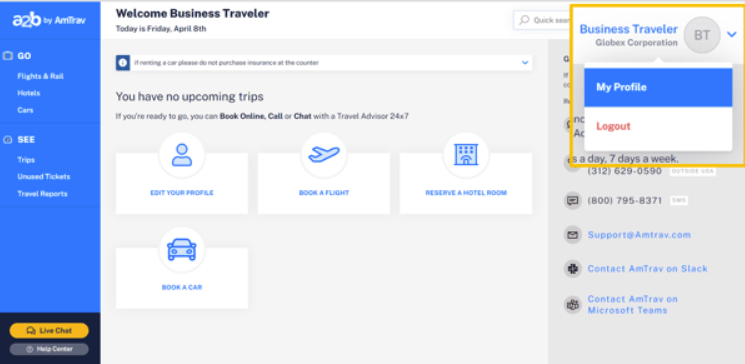
AirLegance user manual.pdf

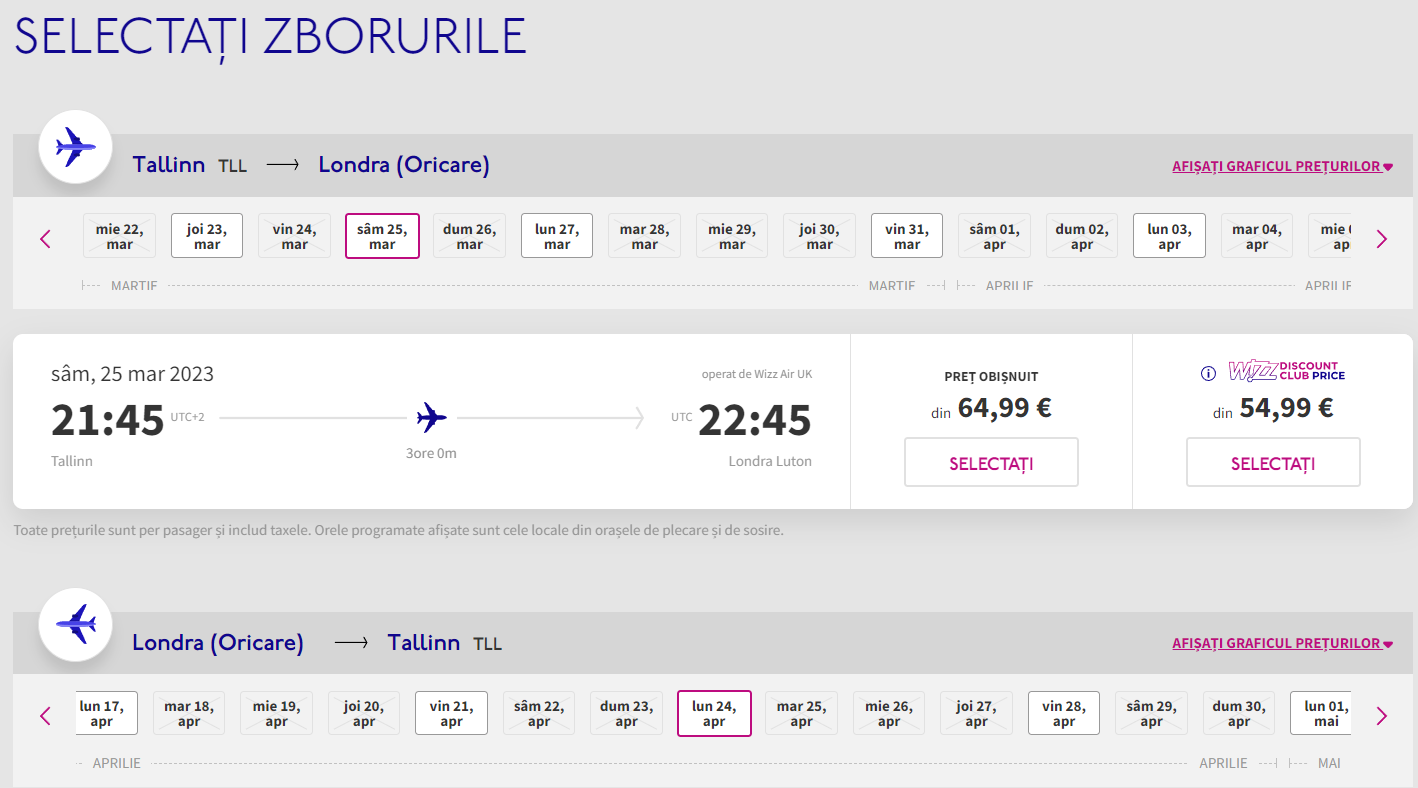
## Assumptions and Dependencies

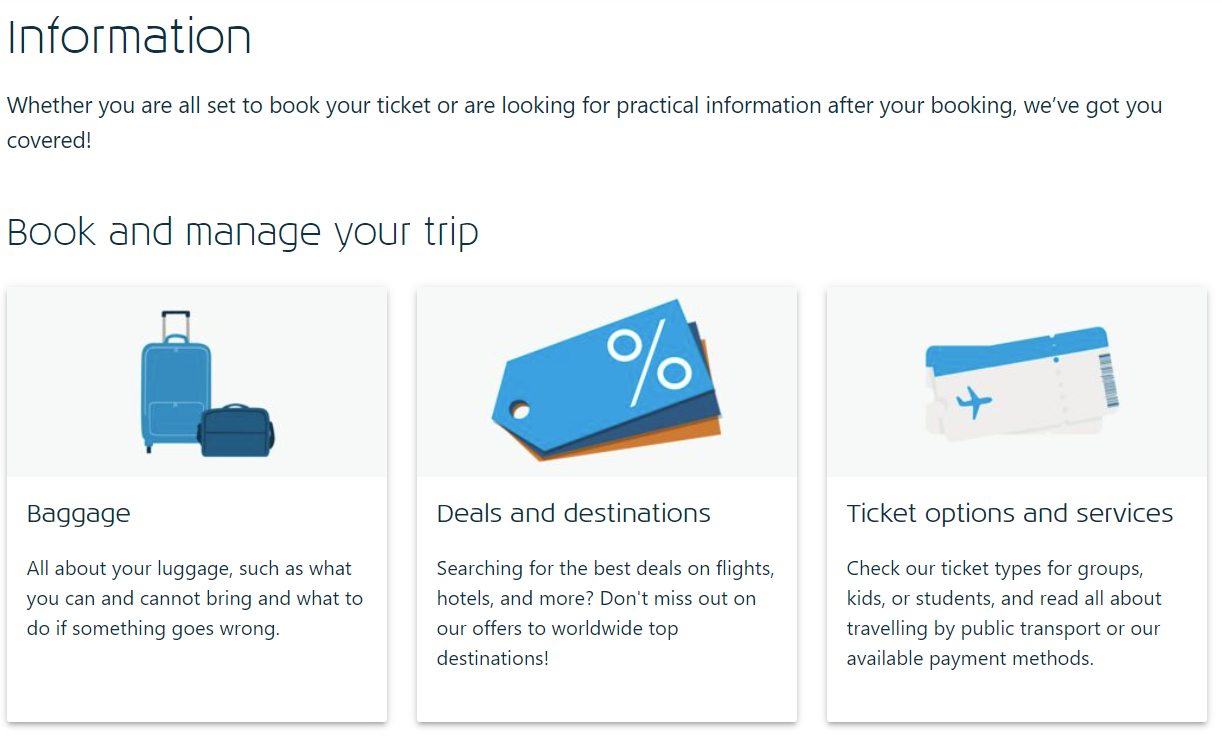
N/A

# External Interface Requirements

## User Interfaces







## Hardware Interfaces

N/A

## Software Interfaces

Entity Framework Core 7: an ORM that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.

## Communications Interfaces

HTTP to communicate to the Restful API

# System Features

## Flight Information

**4.1.1 Description and Priority**

The user wants to view available flights with specific criteria.

High Priority.

**4.1.2 Stimulus/Response Sequences**

**IN:** A search window with information will be completed.

**OUT:** A list of available flights respecting the criteria.

**4.1.3 Functional Requirements**

**REQ-1:** Display available flights - The system should present a list of available flights based on the user's search criteria (e.g., origin, destination, date, etc.).

**REQ-2:** Not all the search criteria must be completed when looking for a flight (such as returning date or destination city).

**REQ-3:** Show departure and arrival times - The system should clearly display the scheduled departure and arrival times for each flight in the search results.

**REQ-4:** Provide ticket pricing information - The system should display the ticket prices for each flight, including different fare classes (e.g., economy, business, first-class) if applicable.

**REQ-5:** Filter and sort options - The system should allow users to filter and sort the search results based on various factors such as departure time, arrival time, ticket price, duration.

**REQ-6:** Flight details page - The system should provide a dedicated page for each flight, displaying more in-depth information about the flight, such as the airline, flight number, aircraft type, seat availability, and any additional services or fees.

**REQ-7:** Real-time updates - The system should provide real-time updates on flight availability, prices, and changes in departure or arrival times, ensuring that users have access to the most accurate information.

**REQ-8:** Responsiveness and accessibility - The system should be user-friendly and accessible on various devices, such as desktop computers, tablets, and smartphones, ensuring a smooth experience for all users.

## Register Functionality

**4.2.1 Description and Priority**

The user will complete the registration fields with personal information and it will submit the form to create an account.

High priority.

**4.2.2 Stimulus/Response Sequences**

**IN:** Registration fields will be completed with personal information.

**OUT:** Account created.

User already existing.

**4.2.3 Functional Requirements**

**REQ-9:** The inputs will be read from the interface.

**REQ-10:** The inputs that are read will be stored in the database.

**REQ-11:** Check if all the required fields are completed.

**REQ-12:** If not all fields are completed display a message.

**REQ-13:** If the user’s information is already present in the database display a message.

**REQ-14:** Register button will have a hover animation.

**REQ-15:** Register button will have a click animation.

**REQ-16:** All the fields to be completed will have a hover and focus animation.

**REQ-17:** Username field must be unique.

## Login Functionality

**4.3.1 Description and Priority**

The user will complete the login fields with username and password and will gain access to the website if the form is submitted successfully.

High priority.

**4.3.2 Stimulus/Response Sequences**

**IN:** Login fields will be completed with username and password.

**OUT:** Wrong username/password.

Access granted and redirect to index page.

**4.3.3 Functional Requirements**

**REQ-18:** Username and password will be read in the interface.

**REQ-19:** Check if all the required fields are completed.

**REQ-20:** If the information submitted corresponds with database information the user will be granted access.

**REQ-21:** If the information submitted does not correspond with database information the user will not be granted access.

**REQ-22:** Login button will have a hover animation.

**REQ-23:** Login button will have a click animation.

**REQ-24:** Username field will have a hover animation.

**REQ-25:** Username field will have a focus animation.

**REQ-26:** Password field will have a hover animation.

**REQ-27:** Password field will have a focus animation.

## User Profile Information

**4.4.1 Description and Priority**

The user wants to view and modify information displayed on his account.

Medium Priority.

**4.4.2 Stimulus/Response Sequences**

**IN:** Information changed successfully.

**OUT:** The user can modify personal information.

**4.4.3 Functional Requirements**

**REQ-28:** When “My Profile” is pressed, the user’s profile along with his personal information will be displayed in a new page.

**REQ-29:** When “Change account information” is pressed, a new page will be displayed with all the old information in the specific fields.

**REQ-30:** When “Change password” is pressed, a new page will be displayed.

**REQ-31:** The new page from “Change password” must have 2 fields: “New password” and “Confirm new password”.

**REQ-32:** The information written in these two fields must be the same in order for the password to be changed.

**REQ-33:** A message with “Password changed successfully” will be displayed.

**REQ-34:** If the new password corresponds with the old one, a message “The new password cannot match the current one” will appear.

**REQ-35:** Any field completed in the registration phase, after pressing “Change account information” can be modified.

**REQ-36:** All the fields after pressing “Change account information”, will have a hover animation.

**REQ-37:** All the fields after pressing “Change account information”, will have a focus animation.

**REQ-38:** After changing whatever information a button “Save changes” will be pressed in order to save the changed information.

**REQ-39:** After pressing “Save changes” a message with “Information changed successfully” will appear.

## User Profile as Employee

**4.5.1 Description and Priority**

The employee wants to see a user’s profile.

Low Priority.

**4.5.2 Stimulus/Response Sequences**

**IN:** The employee chooses to view a user’s profile.

**OUT:** The selected user’s profile will be displayed to the employee

**4.5.3 Functional Requirements**

**REQ-40:** The full name must be displayed to the employee.

**REQ-41:** The upcoming flights of the user must be displayed to the employee.

**REQ-42:** The email address must be displayed to the employee.

**REQ-43:** The phone number must be displayed to the employee.

**REQ-44:**The employee must have the ability to cancel or book specific flights for specific users.

## Contact Us Functionality

**4.6.1 Description and Priority**

The user must be able to send a complaint to the air company.

Medium Priority.

**4.6.2 Stimulus/Response Sequences**

**IN:** The user will complete specific fields, and add the complaint.

**OUT:** The message will be stored somewhere and read by the responsible staff.

A message displaying “Your complaint has been submitted. You will be contacted by our team as soon as possible.” will appear.

**4.6.3 Functional Requirements**

**REQ-45:**Fullname field will have a hover/focus animation.

**REQ-46:**Email field will have a hover/focus animation.

**REQ-47:**Phone number field will have a hover/focus animation.

**REQ-48:**Your complaint field will have a hover/focus animation.

**REQ-49:**When the button “Submit” will be pressed, a message displaying “Your complaint has been submitted. You will be contacted by our team as soon as possible.” will appear.

**REQ-50:**Email field must respect a specific format (@gmail, @yahoo etc. - must be a valid email).

**REQ-51:**If email doesn’t respect the format, a message showing that you need to respect a specific format will be displayed.

# Other Nonfunctional Requirements

## Performance Requirements

* Internet band of 5MB/s
* 4GB of RAM

## Safety Requirements

N/A

## Security Requirements

* Access is done by username and password.
* Passwords will be stored in the database and encrypted.

## Software Quality Attributes

* Use pascal casing ("PascalCasing") when naming a class, record, or struct
* When naming an interface, use pascal casing in addition to prefixing the name with an I. This clearly indicates to consumers that it's an interface.
* When naming public members of types, such as fields, properties, events, methods, and local functions, use pascal casing.
* Use camel casing ("camelCasing") when naming private or internal fields, and prefix them with \_.
* When writing method parameters, use camel casing.
* Use implicit typing to determine the type of the loop variable in [for](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/statements/iteration-statements#the-for-statement) loops.
* When writing positional records, use pascal casing for parameters as they're the public properties of the record.
* Use parentheses to make clauses in an expression apparent.
* Place the comment on a separate line, not at the end of a line of code.
* Insert one space between the comment delimiter (//) and the comment text.

## Business Rules

* Only authorized personnel should have access to the software system and its data.
* Flight schedules must be regularly updated to reflect any changes or cancellations.
* All customer information must be kept confidential and protected in accordance with data protection regulations.
* Customers must be able to view their reservations and ticketing details online via the web portal.  
  The system must have a user-friendly interface that simplifies the process of managing flight schedules, passenger data, and reservation documents.

# Other Requirements

*N/A*

**Appendix A: Glossary**

*N/A*

**Appendix B: Analysis Models**

*N/A*

**Appendix C: To Be Determined List**

*N/A*