MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

PROJECT

for subject

Object Oriented Programming

Airport Management System

Andrei Pintilie

Academic Year: 2023 – 2024

Group: 30425

Semester Project
Computer Science Department
Subject: Object Oriented Programming
Coordinators: Prof. Dr. Eng. Marius Joldoș, Eng. Bogdan Bindea
Date: 13.01.2024

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

Contents

1. Introduction

a. Problem description

In this increasingly fast paced world, quick and reliable transport of both people and cargo finds itself in higher and higher demand. The fastest mode of transportation remains the airplane, which can connect virtually any two points of the globe in less than a day. Therefore the aviation industry has seen a record boom in spite of the COVID pandemic, with aircraft manufacturers receiving record orders from airlines (1) and numerous airports expanding their operations (2) to match the ever-increasing number of passengers (3). Naturally, airports around the world will need increasingly robust management systems to cope with the surging demand.

b. Objectives

This project aims to implement a small scale solution for airport management which allows day to day operations to be performed and certain supervising actions. The three main sections of this prototype revolve around administrative work (that of the admin users), check-in desk operations and ground handler operations. In its whole, the app keeps track of all passengers and their luggage, aircraft and flights which pass through the airport where it is deployed.

2. User requirements

a. Industry specific requirements

Since the aviation industry is a fast paced one, the application needs to be quick, responsive and to have a high degree of availability. Additionally, since the data collected by this app is highly sensitive and crucial for safe and legal flight operation, all data must be stored in multiple locations to prevent outages caused by hardware failures or other external factors. In order to achieve these two requirements, data which is required by the current window is stored in memory for quick access and any changes made by the users are instantly persisted to the database after validation.

b. User types

The application has 3 user types:

- Admin
- Check-in

- Ground Handler

Each user type has specific windows with functionality which allows them to complete the tasks required by their role.
The administrator has access to almost all data and can manage the users of the application and the schedule of arriving and departing flights, as well as the aircraft stored within the database.
The check-in desk user is tasked with adding the incoming passengers and their luggage data to the departing flights.
The ground handler user has access and control for the loading operations of aircraft.

3. Data modelling

a. Technologies

SQLite is a C-language based library which implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. It is the most common used database engine in the world. (4)
For the purposes of this prototype, SQLite was chosen as the preferred 'pseudo' RDBMS for several reasons.
Firstly the convenience of a "one-file" database which allows for better integration with version control software such as Git and allowed development to take place from multiple locations, networks and machines, whereas a "traditional" server based RDBMS such as PostGreSQL or MySQL would have required a more complicated setup to facilitate access to a host database from outside the machine and network on which it was based.
Secondly, as the name suggests, SQLite has very low hardware requirements which make it optimal for achieving high responsiveness of the application, and allows the application to be deployed on a variety of machines including those with lower hardware specifications.

b. Entities, attributes, relationships

To implement the required functionality the following entities were defined:

- User: keeps track of the application's users
- Aircraft Type: records technical specifications about a specific aircraft type (such as Airbus A320 or Boeing 737)
- Aircraft: records the specific details for each individual aircraft
- Flight: records details for all incoming and departing flights
- Passengers: records passenger details
- Baggage: records details of each piece of luggage

Each of the entities has the following attributes:

- User: **uid**, username, password, user type
- Aircraft Type: **id**, name, manufacturer, passenger capacity, hold capacity, fuel capacity
- Aircraft: **aid**, aircraft registration, notes, aircraft type id

- Flight: **fid**, flight number, type (arrival/departure), eta, etd, origin, destination, notes, aircraft registration, date
- Passenger: **pid**, first name, last name, phone number, flight id
- Baggage: **bid**, weight, category, passenger id, flight id

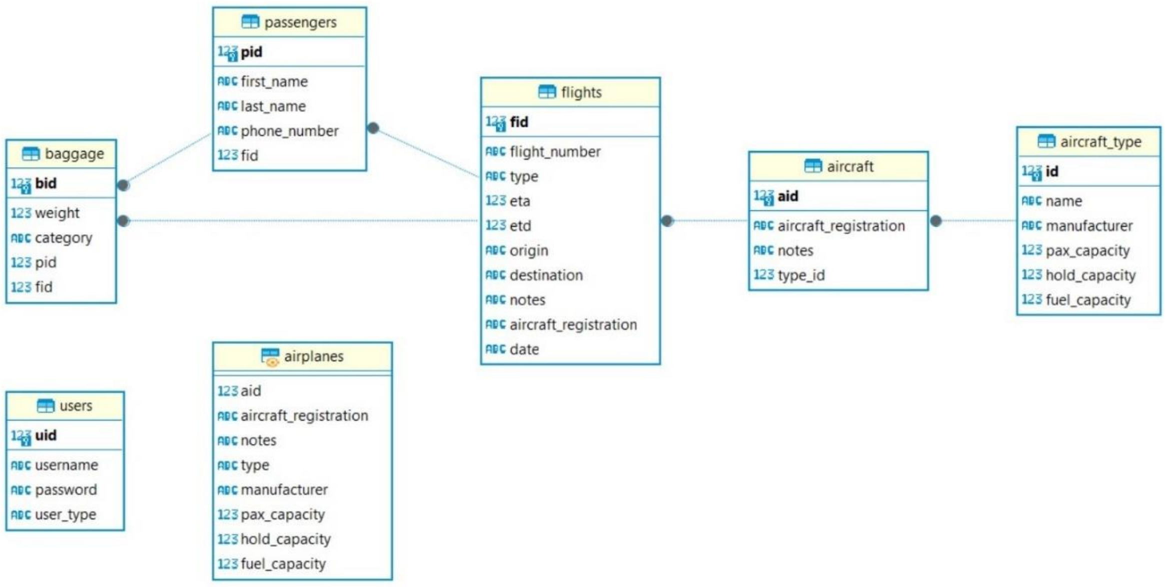
For the purposes of this prototype the following relationships were considered:

- Each **aircraft** must have one **type**
- Each **flight** must have an **aircraft** assigned. One **aircraft** can be assigned to multiple **flights**
- Each **passenger** must be assigned to a **flight**. One **flight** can have multiple **passengers** assigned to it
- Each **baggage** must be assigned to a **flight**. One **flight** can have multiple **baggages** assigned to it.
- Each **baggage** must be assigned to a **passenger**. One **passenger** can have multiple **baggages** assigned to it.

A simple view which compiles all data in tables Aircraft and AircraftType was added for ease of access to all information pertaining to an entry in the aircraft table.

c. ERD for complete data model

The following ERD depicts the entire data model:



4. Implementation details

a. Technologies

i) Java
Java was chosen as the primary programming language for this project for multiple reasons including: ease of use, high number of libraries available and multi-platform capabilities.

ii) Java Swing Framework
Since the purpose of this project is to present a working prototype, Java Swing was selected as the framework in which the GUI was built. Swing, whilst not as visually attractive or easily customisable as other, more modern frameworks, is an incredibly robust, provenly reliable library which allows developers to quickly produce functioning applications.

iii) IntelliJ Swing GUI Designer
In order to further speed-up the development process, the IntelliJ built-in GUI designer for Swing was used to allow for quicker and more effective building of the relative component alignments and general layout of the application windows and dialogs.

b. Module description

The implementation of this project consists of 25 classes, 2 enumerations and 1 interface. The codebase is split into several main parts:

- Entities: the representations of the entities in the data base and the corresponding enums and interfaces to properly implement the required structure and functionalities
- Windows: the classes controlling all the windows in the application, the logic, data processing and the corresponding layouts
- Dialogs: the classes controlling all the dialogs in the application, the logic, data processing and the corresponding layouts
- Renderers: the custom CellRenderers for proper display of entities in the JLists present throughout the application

c. GUI overview

In this section will be presented the 3 expected workflows corresponding to each of the 3 types of users.

Upon application startup all users will be presented with the login screen:

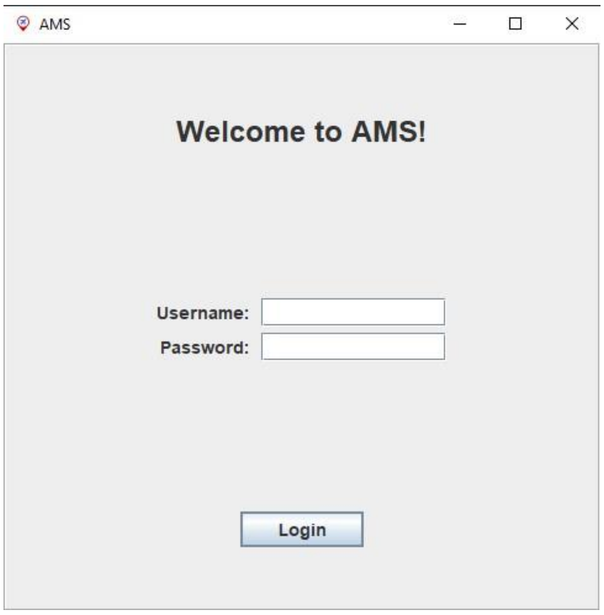
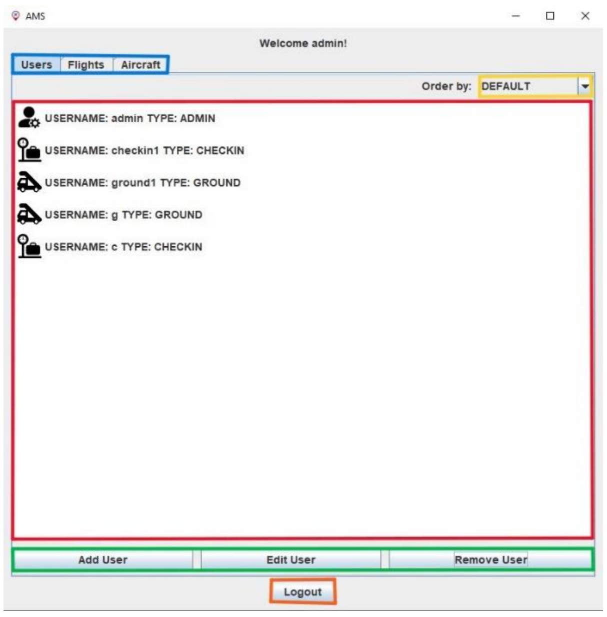


Figure 1 Login Screen

This is where the application flow diverges.

i) Admin user

After logging in, the admin user is greeted with the following home screen:



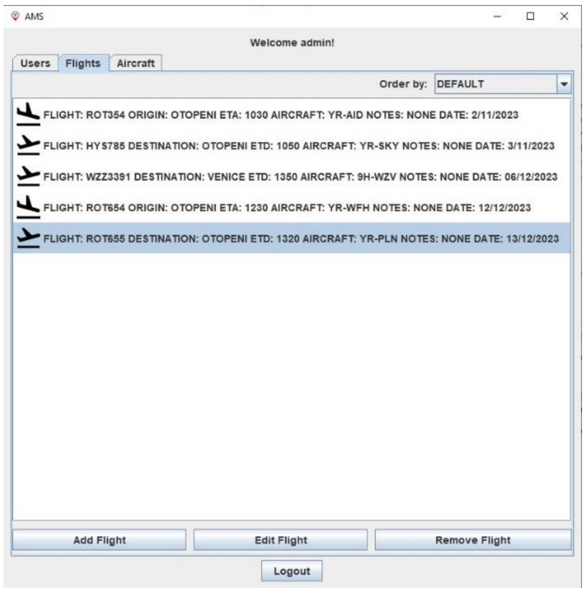
```
USERNAME: admin TYPE: ADMIN
USERNAME: checkin1 TYPE: CHECKIN
USERNAME: ground1 TYPE: GROUND
USERNAME: g TYPE: GROUND
USERNAME: c TYPE: CHECKIN
```

Figure 2 Admin home screen

The main parts of this window are:

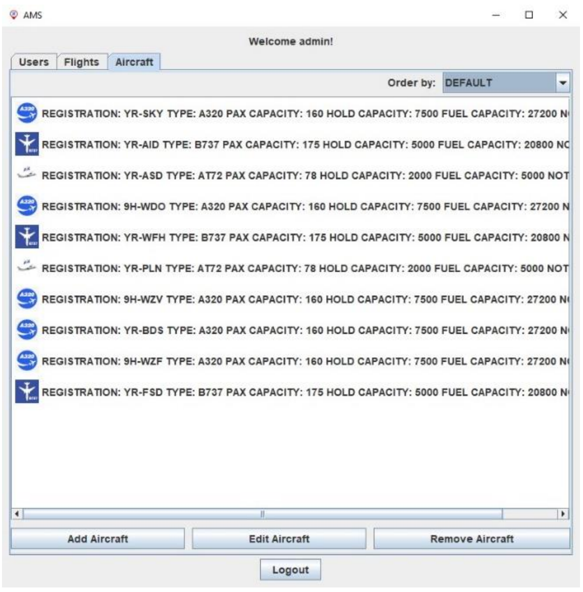
- The 3 tabs it the upper part of the window corresponding to the users, flights and aircraft (highlighted in blue)
- The list in the center which displays all rows of the corresponding type (highlighted in red)
- The selection box in the upper left part which allows the reordering of the elements in the list (highlighted in yellow)
- The 3 buttons in the lower part which allow basic Create, Update and Delete operations on the selected table (highlighted in green)
- The logout button (highlighted in orange)

The other 2 tabs provide similar layouts but with all elements adapted to suit the data type presented in the central list.



FLIGHT: ROT354 ORIGIN: OTOPENI ETA: 1030 AIRCRAFT: YR-AID NOTES: NONE DATE: 2/11/2023
FLIGHT: HYS785 DESTINATION: OTOPENI ETD: 1050 AIRCRAFT: YR-SKY NOTES: NONE DATE: 3/11/2023
FLIGHT: WZZ3391 DESTINATION: VENICE ETD: 1350 AIRCRAFT: 9H-WZV NOTES: NONE DATE: 06/12/2023
FLIGHT: ROT654 ORIGIN: OTOPENI ETA: 1230 AIRCRAFT: YR-WFH NOTES: NONE DATE: 12/12/2023
FLIGHT: ROT655 DESTINATION: OTOPENI ETD: 1320 AIRCRAFT: YR-PLN NOTES: NONE DATE: 13/12/2023

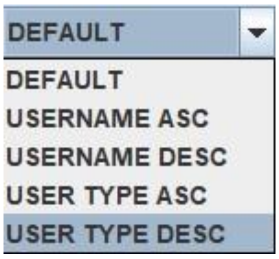
Figure 3.a Flights Tab of Admin home screen

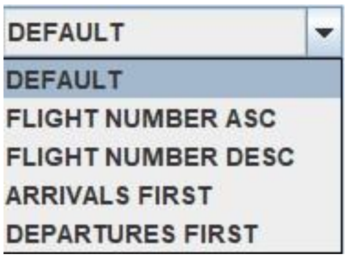


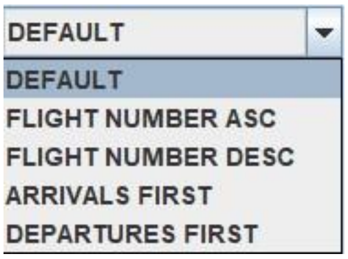
REGISTRATION: YR-SKY TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 FUEL CAPACITY: 27200 N
REGISTRATION: YR-AID TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 FUEL CAPACITY: 20800 NO
REGISTRATION: YR-ASD TYPE: AT72 PAX CAPACITY: 78 HOLD CAPACITY: 2000 FUEL CAPACITY: 5000 NOT
REGISTRATION: 9H-WDO TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 FUEL CAPACITY: 27200 N
REGISTRATION: YR-WFH TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 FUEL CAPACITY: 20800 N
REGISTRATION: YR-PLN TYPE: AT72 PAX CAPACITY: 78 HOLD CAPACITY: 2000 FUEL CAPACITY: 5000 NOT
REGISTRATION: 9H-WZV TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 FUEL CAPACITY: 27200 N
REGISTRATION: YR-BDS TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 FUEL CAPACITY: 27200 N
REGISTRATION: 9H-WZF TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 FUEL CAPACITY: 27200 N
REGISTRATION: YR-FSD TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 FUEL CAPACITY: 20800 N

Figure 3.b Aircraft Tab of Admin Home screen

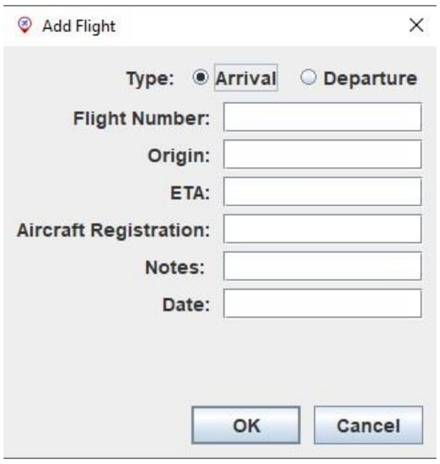
The selection box provides sorting options specific to the data in the central list as exemplified below:





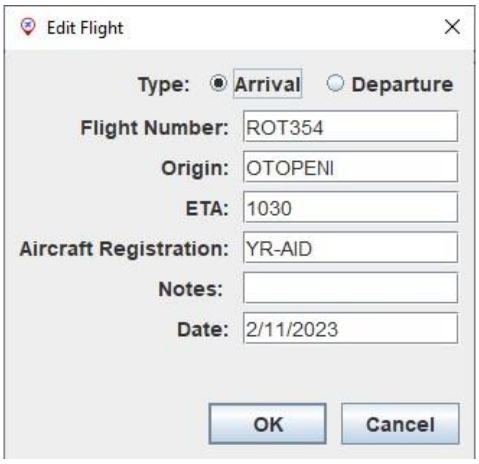


The Create and Update buttons launch corresponding dialogs for each entity type. They are exemplified only for the Flight entity:


Add Flight

Type:	[✓ Arrival] Departure
Flight Number:	
Origin:	
ETA:	
Aircraft Registration:	
Notes:	
Date:	

Figure 4.a Add flight dialog

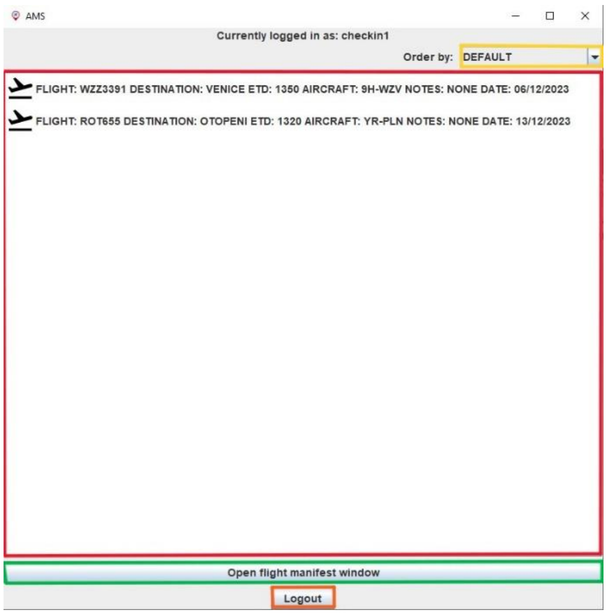

Edit Flight

Type:	[✓ Arrival] Departure
Flight Number:	ROT354
Origin:	OTOPENI
ETA:	1030
Aircraft Registration:	YR-AID
Notes:	
Date:	2/11/2023

Figure 4.b Edit flight dialog

ii) Check-in user

After logging in, the check-in user is greeted by the following home screen:


AMS

Currently logged in as: checkin1

Order by: DEFAULT

FLIGHT: WZZ3391 DESTINATION: VENICE ETD: 1350 AIRCRAFT: 9H-WZV NOTES: NONE DATE: 06/12/2023
FLIGHT: ROT655 DESTINATION: OTOPENI ETD: 1320 AIRCRAFT: YR-PLN NOTES: NONE DATE: 13/12/2023

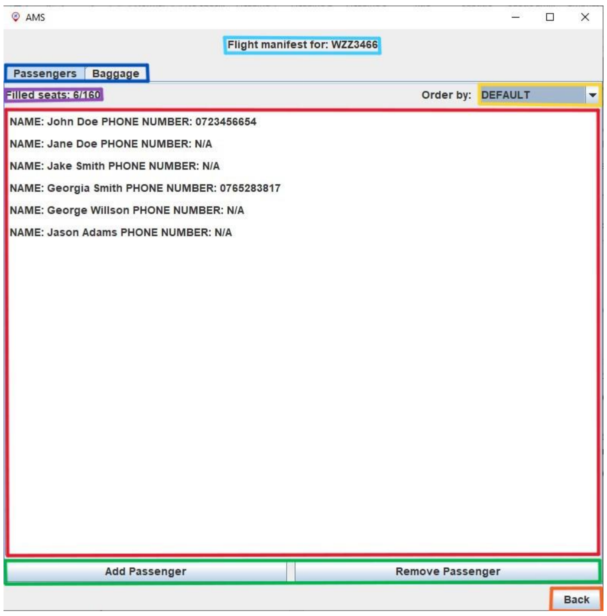
Figure 5 Check-in home screen

The main elements of this screen are:

- The list in the center which displays all departing flights (highlighted in red)
- The selection box in the upper left part which allows the reordering of the elements in the list (highlighted in yellow)

- The button in the lower center part which opens the Flight Manifest for the selected flight (highlighted in green)
- The logout button (highlighted in orange)

After selecting a flight, the user can either click the button or double click the left mouse button to open the Flight Manifest which leads to the following screen:



Flight manifest for: WZZ3466

Filled seats: 6/160

Order by: DEFAULT

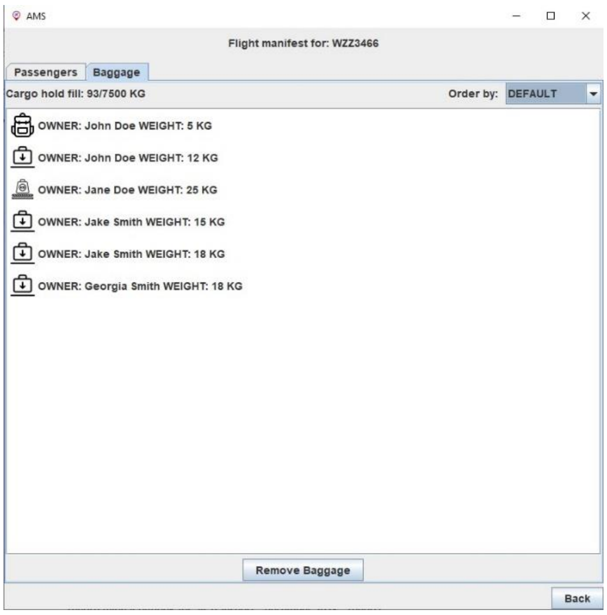
NAME: John Doe PHONE NUMBER: 0723456654
NAME: Jane Doe PHONE NUMBER: N/A
NAME: Jake Smith PHONE NUMBER: N/A
NAME: Georgia Smith PHONE NUMBER: 0765283817
NAME: George Willson PHONE NUMBER: N/A
NAME: Jason Adams PHONE NUMBER: N/A

Figure 6 Flight Manifest

The main components of this screen are:

- The 2 tabs it the upper part of the window corresponding to the passengers and baggage (highlighted in blue)
- The list in the center which displays all entities of the corresponding type (highlighted in red)
- The selection box in the upper left part which allows the reordering of the elements in the list (highlighted in yellow)
- The information label in the center upper part which displays the flight the user has previously selected (highlighted in light blue)
- The flight load label which displays the number of checked-in passengers/bags and the maximum capacity of the flight based on the aircraft which will carry it out (highlighted in purple)
- The 2 buttons for adding/removing passengers (highlighted in green)
- The back button (highlighted in orange)

The baggage tab of the flight manifest has a similar layout:



Cargo hold fill: 93/7500 KG

OWNER: John Doe WEIGHT: 5 KG
OWNER: John Doe WEIGHT: 12 KG
OWNER: Jane Doe WEIGHT: 25 KG
OWNER: Jake Smith WEIGHT: 15 KG
OWNER: Jake Smith WEIGHT: 18 KG
OWNER: Georgia Smith WEIGHT: 18 KG

Figure 7 Baggage tab of Flight Manifest

The only notable difference is that in the case of luggage there is no option to add a bag separately since all bags must have a passenger as owner.

The sorting options for passengers and bags are:

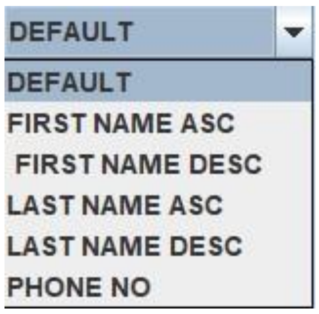


Figure 8.a Passenger sorting options

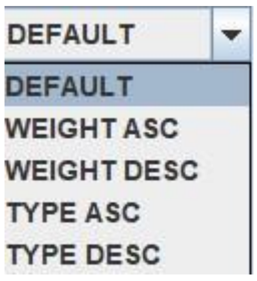


Figure 8.b Baggage sorting options

The add passenger button launches the following dialog:

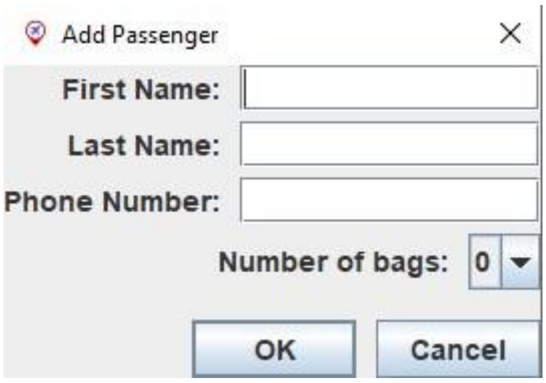


Figure 9 Add passenger dialog

The "Number of bags" selection box allows adding up to 5 bags for the passenger being checked-in and dynamically adjusts the dialog:

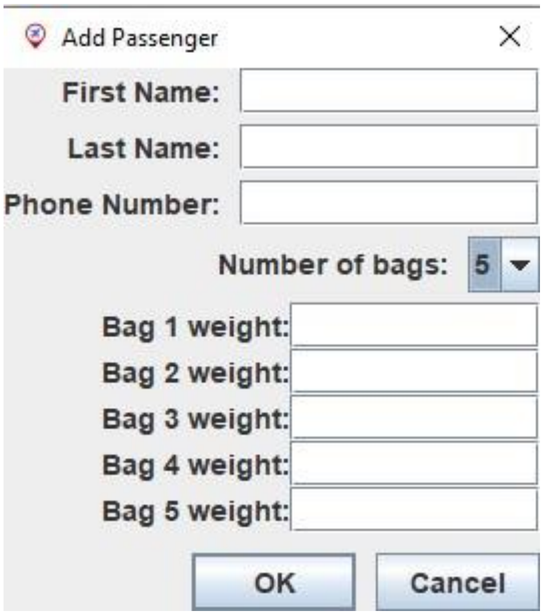
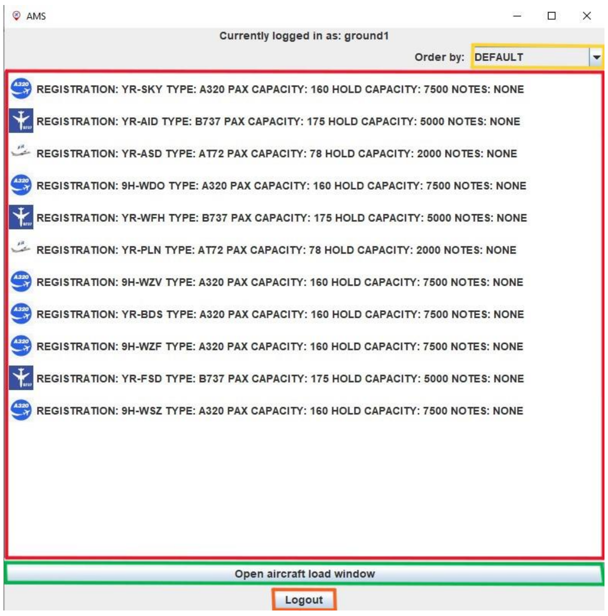


Figure 10 Add passenger dialog expanded

iii) Ground handler

After logging in, the ground handler is greeted by the following screen:



Currently logged in as: ground1

Order by: DEFAULT

```
REGISTRATION: YR-SKY TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
REGISTRATION: YR-AID TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 NOTES: NONE
REGISTRATION: YR-ASD TYPE: AT72 PAX CAPACITY: 78 HOLD CAPACITY: 2000 NOTES: NONE
REGISTRATION: 9H-WDO TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
REGISTRATION: YR-WFH TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 NOTES: NONE
REGISTRATION: YR-PLN TYPE: AT72 PAX CAPACITY: 78 HOLD CAPACITY: 2000 NOTES: NONE
REGISTRATION: 9H-WZV TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
REGISTRATION: YR-BDS TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
REGISTRATION: 9H-WZF TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
REGISTRATION: YR-FSD TYPE: B737 PAX CAPACITY: 175 HOLD CAPACITY: 5000 NOTES: NONE
REGISTRATION: 9H-WSZ TYPE: A320 PAX CAPACITY: 160 HOLD CAPACITY: 7500 NOTES: NONE
```

Figure 11 Ground handler home screen

The main component of this screen are:

- The list in the center which displays all departing flights (highlighted in red)
- The selection box in the upper left part which allows the reordering of the elements in the list (highlighted in yellow)
- The button in the lower center part which opens the Aircraft Load for the selected aircraft (highlighted in green)
- The logout button (highlighted in orange)

One can easily spot the similarities between this screen and the check-in user home screen. That is because they are in fact the same class, which adjusts the layout and elements at runtime to suit the functionality required by the user which logged in.

After selecting a flight, the user can either click the button or double click the left mouse button to open the Aircraft Load which leads to the following screen:

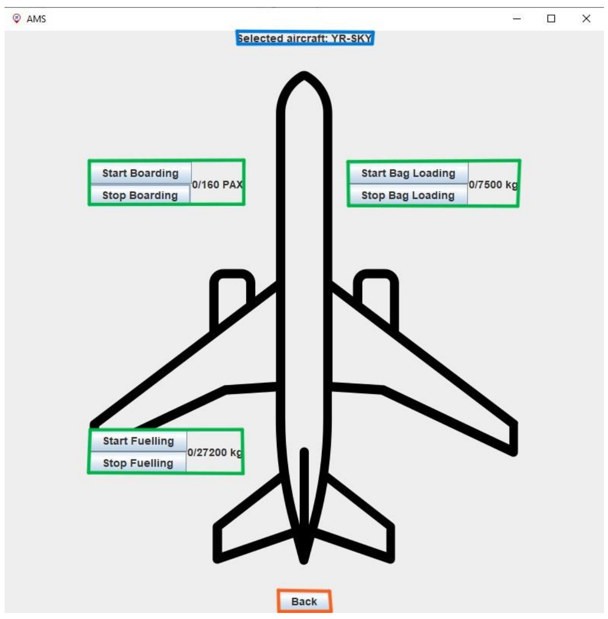


Figure 12 Aircraft Load screen

The main components of this screen are:

- The information label in the upper center which displays the selected aircraft (highlighted in blue)
- The 3 loading panels corresponding to passenger boarding, bag loading and fuel loading, each having 2 start/stop buttons and a label displaying the progress (highlighted in green)
- The back button (highlighted in orange)

In order to make the 3 loading processes run in parallel, the application makes use of multithreading. Each start button launches a new thread which increases the loaded quantity with a fixed amount per second until either the maximum capacity is reached, or the user presses the stop button which interrupts that thread.

5. Known issues and future improvements

a. Known issues

i) Single user connection database

Since the RDBMS in usage is SQLite, the application supports just one concurrent connection to the database, which obviously makes it impossible to deploy the solution in an actual airport where there may be tens or hundreds of users which require simultaneous access to the database.

The solution to this is quite trivial thanks to the modularity of the application. It consists of simply replicating the current database's structure using a RDBMS which allows multiple connections, such as PostGreSQL or MySQL, and modifying the method which establishes the connection to the database, which is then used througout the application.

ii) Hard-coded constants

There are multiple constants used throughout the app such as the weights delimiting baggage categories, the maximum number of bags per passenger, the rate of boarding/bag loading/fuel loading and others which were hard-coded to speed up the development process, since the purpose of this project is primarily demonstrative.

iii) Aircraft Load functionality

All the features presented in the Aircraft Load screen are solely for demonstration purposes. They are stored only in memory and are not persisted into the database.

iv) Relaxed security measures

All data stored in the database including user passwords are stored in clear-text and the database in not protected with any form of authentication. This is a conscious decision in order to aid the developmpent and debugging process. However it is obvious that in a real world scenario this is an enormous security risk, and almost certainly illegal since it does not comply with GDPR regulations regarding sensitive personal data such as names and phone numbers.

The solution to this issue is once again relatively trivial. All data must pass through an encryption algorithm before being stored in the database, and decrypted after it has been extracted. Alternatively to reduce workload on reading and writing in the database, it is possible to encrypt only sensitive fields such as passwords or names, and to restrict access to the database either via password or public key authentication.

b. Future improvements

i) Real-time tracking

In order to expand the application functionality, real-time tracking of the current date and hour could be added along with additional sorting options to keep track of flights taking place in specific time intervals. This additional information would allow clarifying which flight is currently being carried out by each aircraft, since one aircraft can theoretically carry out multiple flights at different points in time. Additionally this would allow the application to keep track of which aircraft are currently on the ground and it would give ground handler users more options to report delays in operation which could then be relayed to the passengers.

ii) Global settings and configuration

To address the previously mentioned issue of hard-coded constants, configuration file functionality could be added along with a corresponding tab in the Admin user home screen which would allow the administrators to dynamically adjust the functioning of the application. This concept could be further refined by defining local configurations, which would affect only one particular application instance, and global configurations, which would alter the functioning of all application instances.

6. References

1. **Reuters.** *Reuters.* [Interactiv] https://www.reuters.com/business/aerospace-defense/airbus-posts-record-orders-keeps-delivery-crown-2023-2024-01-11/.

2. **[Interactiv] https://airportindustry-news.com/terminal-construction-and-renovation-projects-in-2023/.**

3. IATA. [Interactiv] https://www.iata.org/en/iata-repository/publications/economic-reports/global-outlook-for-air-transport---december-2023---report/.

4. *sqlite.org.* [Interactiv] https://www.sqlite.org/index.html.