

# BookMyFlight

## Table of Contents

### Appendix A

.1	Scenario - Book My Ticket Program .....	1
.2	Design Paradigm .....	2
.3	Expected Output .....	2

### Appendix B

.4	Program Features .....	5
.5	Challenges .....	10
.6	Learning Outcome.....	10

## Project description:

### Scenario: Book My Ticket

Passengers will be able to log in, create an account or use our program anonymously, where their information will be set by the program. Once they pass the identification page, they will be able to book a ticket. They will be given a set of hardcoded tickets. If the chosen ticket has a plane with full seats, then a message will appear stating that they cannot purchase that ticket due to the unavailability. Once selected, they will go through the payment process where they will enter the number of tickets wanted and their card information. If they have an account, a premium or a regular passenger, credits can be accumulated and can be transformed into discounts. Premium passengers will always have a 5% discount on their total. Their ticket will then be booked in their name. If they want to modify their ticket, they will have to request an employee to cancel the ticket. Once they are done with the process of cancelling their initial ticket, the passenger will choose the corrected ticket. To cancel a ticket, they will still need to request an employee, which will refund them. Their refund information will be available in the passenger's refund list. For the Employees, they can only log in or create an account. They will have the task of booking a ticket for a certain passenger. However, they handle the process of cancelling and refunding a passenger's ticket. The employee will send back a notice mentioning the amount returned, the ticket ID and the passenger's ID. Passengers who have used a discount on their total will also have their credits back, but not their 5% discount if they are a premium member. Each operation will be registered in case of any error. Once a ticket is booked, the ticket will register the seat attributed to the passenger, the passenger's temporary ID, if they are booking anonymously, and the passenger's name. Lastly, after their purchase, credits will be added to passengers with an account. They will then be presented to the "Option" page, where they have the choice to view their credits, view their booked and refunded tickets, and modify, book or cancel a ticket. Invalid input will not be accepted, and the user will have to re-enter their credentials until the system recognizes them as being valid.

## **Design Paradigm**

- The MVC pattern will be used to divide the project into three sections: model, view and controller. In the view, we will provide the interface using JavaFX.

### **View: Graphical User Inter**

- On the first page the user can decide if they would like to continue the program on light or dark mode, the default would be light mode
- The user will have to complete the requested information to go to the next page or they can cancel the process to leave the page.
- The passenger can view their refund list, accumulated credits and booked tickets.

### **Controller: Logical content and links the model with the view**

- Calculating the total price
- Validation for the user authentication and other user input
- Interaction between the program and the database to fetch and store information.
- Calculate the discount with a certain amount of credits.
- Fetching the entered input to create objects such as the ticket.
- Creating operations for each action being made and storing them in a database

### **Model: Representation of data**

- The fields that state all the data of the ticket
- User containing all the base information that both employee and passenger will inherit
- What data is going to be needed to create an object

## **Expected Output**

### **Creation of an account for a passenger**

- The passenger will have to enter their information: email, phone number, full name and password. A passenger user will be created and added to the passenger list. When click a button their account will be saved with the date and time created. All their lists, refund and booked lists, will be set to null and their credit will be set to 0.

### **Logging into an account for a passenger**

- The passenger will need to enter their email and the password that they've created

in the creation process. The credentials will be searched in the passenger's list and can only move on until a confirmation message appears on the page. A button will appear, and they can click on it to move to the next page.

- When the passenger enters information that is not found in the passengers list, they will not be able to move on to the next page and an error message will appear stating that they will need to re-enter the correct information.

### **Skipping the identification page for a passenger**

- The anonymous passenger will be able to skip the identification page and will continue to use the program with the given identification, meaning that their data will be set by the program to create an anonymous account, though they will still have a userID.

### **Passenger choosing their ticket**

- The passenger will be presented with a set of predefined tickets, they will pick a ticket using a radio button and then they will proceed. If a message pops up saying that the plane is fully booked, then the passenger will need to choose another ticket.

### **Passenger requesting a modification for their ticket.**

- In our program, to modify our ticket, the passenger will need to request to cancel their ticket using the ticket ID that will be cancelled. They will receive a refund on both their money and credits.

### **Passenger requesting to cancel a ticket.**

- The passenger cannot directly cancel their ticket. They will need to request the assistance of an employee. Their refund will be seen in their refunds list and their credit will be refunded.

### **Employee booking a ticket for a passenger**

- Just like the passenger, the employee will be able to book a ticket for a passenger using their provided credentials. The employee will enter the passenger's payment information and they will book a ticket in their name.

### **Employee cancelling a ticket for a passenger**

- The employee will receive a request to cancel a certain ticket from a passenger. The passenger will provide them with the ticketID that they want to cancel, to help the employee cancel the ticket successfully. The employee will refund the money and the credits of the passenger

### **Employee “modifying” a ticket for a passenger**

- The employee will help the passenger to modify their ticket. The passenger will need to request an employee to cancel the ticket. The employee will do the same steps as the cancellation process.

### **Employee logging in the system**

- The employee will need to enter their email and their password to be able to log in. in the system. If the entered information does not match a user then they won't be able to log in to their account.

### **Employee creating an account**

- The employee has the benefit of creating their account which will then be stored among the other employees' accounts.

## **Program Features:**

The GUI is well-deigned for input and output (database and a receipt csv file). The code implements 2 patterns which are singleton and MVC. We used data structures such as Priority Queue, Linked List and Array List. The application gives the choice to the user to choose a language between French and English. The application implements CRUD operations and manages and manipulates data entered by the user, keeping track of changes. In the code we used lambda expressions. We followed a TDD approach and we did a solid testing. We have a git repository and we did some planning there, divided tasks and used branching to not pollute the main branch. We also used UML diagrams, to have a clear view of our project.

In our program, each time a passenger is created he is a regular passenger. It takes a lot of time to become a premium member. This is why we hardcoded a regular passenger.

Here is a brief explanation of the user experience with the GUI:

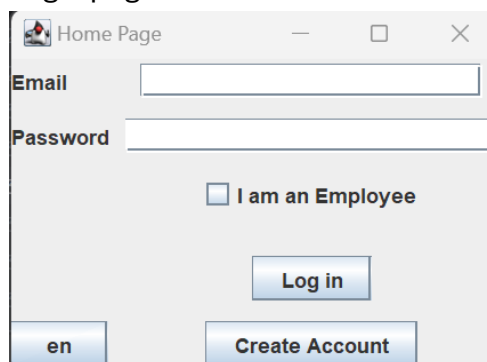
- In the GUI, the first page the user(Passenger or Employee) sees is the Login page. If he wants to create an account he goes to the CreateAccount page, then after he created his account he goes back to the Login page.
- If he is a Passenger he goes to Payment Page, where he chooses or simply takes a look at the ticket he wants. If he is just looking he then goes to the Exit page. If he chooses a ticket, then he goes to BuyTicket page where he inputs his payment information. When the payment is successful, then he goes to Exit page where he

can see the tickets he bought and can select which ones he wants to request a refund if he wants to or he can simply logout and then he will go back to the login page.

- If he is a Employee then he goes to EmployeePayment page, where he chooses or simply takes a look at the ticket that a certain client wants. If he is just looking or he wants to just refund a client, then he goes to the RefundTickets page. If he chooses a ticket for a client, then he goes to BuyTicket page where he inputs the client payment information. When the payment is successful, then he goes to RefundTickets page. In this page, he can refund tickets that are in his queue or he can simply logout and then he will go back to the login page.

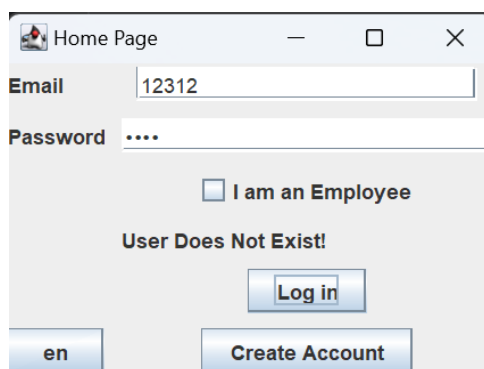
Here is a quick view of how the GUI looks like:

Login page:



The screenshot shows a window titled "Home Page" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a login form. It features two input fields: "Email" and "Password". Below the "Password" field is a checkbox labeled "I am an Employee". There are three buttons: a "Log in" button centered below the checkbox, a "Create Account" button at the bottom right, and a small button labeled "en" at the bottom left.

**\*\* This picture shows what happens when the unexistant user tries to log in\*\***



This screenshot shows the same "Home Page" window, but with an error message displayed. The "Email" field now contains the text "12312" and the "Password" field is filled with dots. The error message "User Does Not Exist!" is centered on the screen above the "Log in" button. The "I am an Employee" checkbox remains unchecked. The "en" button and "Create Account" button are still visible at the bottom.

**\*\* This picture shows what happens when we try to log in as an employee without checking the checkBox\*\***

Home Page

Email

Password

☐ I am an Employee

**This is an Employee Account!**

Payment page:

Payment Page

00002

Ticket id	00002	Price	2500.0	<input type="checkbox"/> 00001
Trip type	Two-way	Seat class	BUSINESS	<input type="checkbox"/> 00002
Outbound date	2024-12-20	Departure	Dubai	<input type="checkbox"/> 00003
Return date	2024-12-30	Destination	New York	<input type="checkbox"/> 00004
Airplane	Emirates	Status	UNBOOKED	<input type="checkbox"/> 00005

Choose a Ticket First!

Credits 0

EmployeePayment page:

Select a ticket/Selectionnez vos Billets

Ticket id	...	Price	...	<b>Select Your Tickets</b> <input type="checkbox"/> 00001 <input type="checkbox"/> 00002 <input type="checkbox"/> 00003 <input type="checkbox"/> 00004 <input type="checkbox"/> 00005
Trip type	...	Seat class	...	
Outbound date	...	Departure	...	
Return date	...	Destination	...	
Airplane	...	Status	...	

BuyTicket page:

Card number (13 to 19 digits)

Card Holder name

Expiration date(MM/YY)

Cvc (3 to 4 digits)

☐ I am an employee

en Pay Ticket(s)

\*\*The next photo shows that the ticket has been associated to a client \*\*

```
[Ticket{ticketID='00001', passengerID='null', airplane=Airplane(airplaneID=00001, assignedAirline=Qatar Airways, availableSeats=150, type=Boeing 707), outboundDate='2024-12-20', returnDate='2024-12-30', price=500.0, tripType='Two-way', status=UNBOOKED, seatType=ECONOMY, departure='Lebanon', destination='Montreal'}]
```

\*\*The next photos shows that the database is populated\*\*

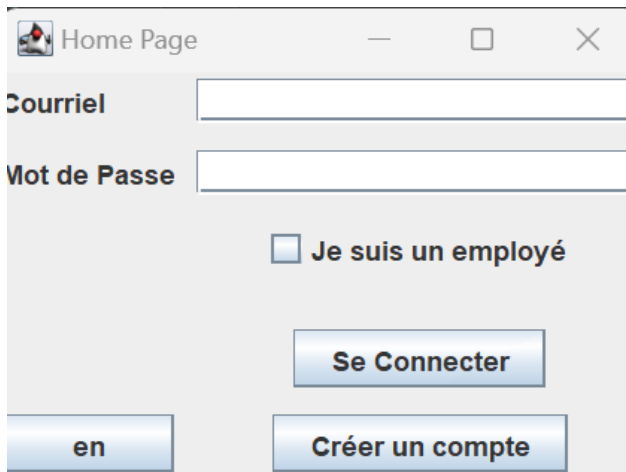
airplane...	assigne...	availabl...	type
00001	Qatar Air...	150	Boeing 7...
00002	Emirates	200	Boeing 7...
00003	Canada A...	150	Boeing 7...
00004	Japan Air...	100	Boeing 7...
00005	Turkish A...	140	Boeing 7...

Table: airplane Page: 0

airplane	assigne...	availabl...	type
00001			Boeing 7...
00002			Boeing 7...
00003	Canada A...	150	Boeing 7...
00004	Japan Air...	100	Boeing 7...
00005	Turkish A...	140	Boeing 7...

\*\*The next photo shows that i18 is present in our app





Home Page

Courriel

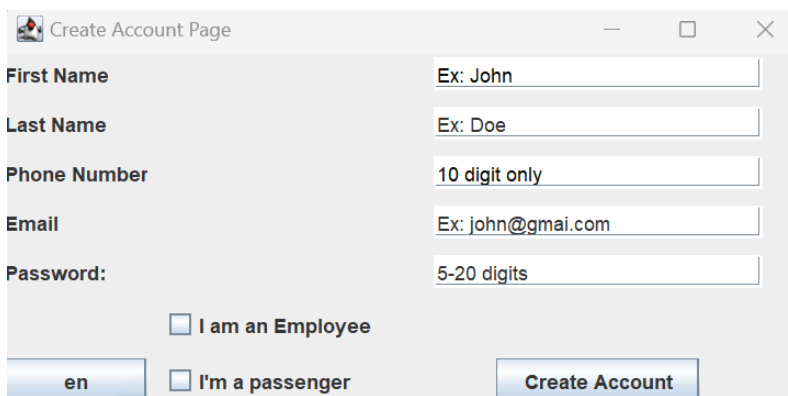
Mot de Passe

☐ Je suis un employé

Se Connecter

en Créer un compte

CreateAccount page:



Create Account Page

First Name Ex: John

Last Name Ex: Doe

Phone Number 10 digit only

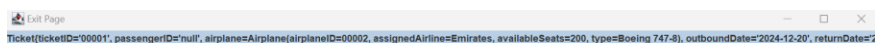
Email Ex: john@gmail.com

Password: 5-20 digits

☐ I am an Employee

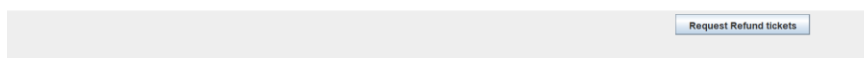
en ☐ I'm a passenger Create Account

Exit page:

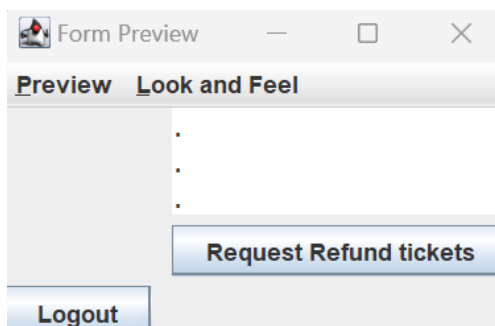


Exit Page

Ticket(ticketID='00001', passengerID=null, airplane=Airplane(airplaneID=00002, assignedAirline=Emirates, availableSeats=200, type=Boeing 747-8), outboundDate='2024-12-20', returnDate='2024-12-20')



Request Refund tickets



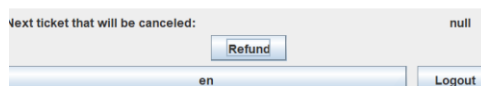
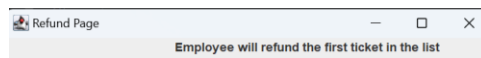
Form Preview

Preview Look and Feel

Request Refund tickets

Logout

### RefundsTicket page:



## Challenges:

It was hard to visualize the project in advance and to imagine and predict how the GUI looks like and how it will be the logic supporting it. It was hard to solve many logical errors, and sometimes the IDE didn't work well and wasn't saving automatically and we had to open the files in File Explorer and copy the code manually into it. It was also challenging to find the right moment to work together because we are both busy during the semester especially in the end because we each have our exams and projects to deliver. So to find the right moment in our schedules to work together in the same time was challenging.

We decided to remove the writing to file feature (the receipt for the passenger) because we didn't have time.

Since we started on a complicated way, our project took way longer than expected since we would mess things up from the beginning, which would later affect our future enhancements. This became a domino effect and created a huge problem at the end. One of our biggest challenges, but also a great learning experience was our time management. Having many courses, many exams, many assignments and many exams during the same week was truly a humbling experience. Though I know that life won't get better, this project has thought us so much about organization, management and having discipline, that it will give us a huge push for our future projects and any kind of obstacles.

## Learning Outcomes:

We have understood better the concepts seen in class because we have practiced them more and in a different way. We understand better how a booking system works and how complex it can be. This hands-on experience has been invaluable and fun. We have learned

the way databases work in java, using sqlite, and storing information about the system in the database. For example, the lists in the singleton class are initialized with the database's queryAll method (for our case). We are updating the system along with the data from the database. This synchronization can be really confusing at first, but when everything is organized, everything runs smoothly. For the mvc pattern, this was our biggest challenge since organizing classes and dividing them into different groups is hard to do it at first. We get confused on what method goes where and if we need to make them static and all these little things that become much bigger when we're getting near the deadline. Talking about deadlines, time management and our communication was truly horrendous. We gained some experience and knowledge about each other's weaknesses and strengths and we learned how to work with one another. Though I may say that it is still not perfect, we made a great huge improvements and we are greatly honored about our growing skills. Synchronizing and juggling all these new concepts that we've learned throughout the year was though, but we have gained so much experience from it and it will certainly help our future selves.