**Airplane booking app**

Name: Stefan Tcaciuc

Contents

[Project description 2](#_Toc55692662)

[Technical issues 2](#_Toc55692663)

[Database design 2](#_Toc55692664)

[Database Prototype 2](#_Toc55692665)

[ER Diagram 4](#_Toc55692666)

[Aplication Design 5](#_Toc55692667)

[Plan/Log 6](#_Toc55692668)

[Conclusion/Review 8](#_Toc55692669)

[References 9](#_Toc55692670)

[Database Script 10](#_Toc55692671)

# Project description

In this project I will be building a java-based app for a flight booking system that uses DBMS to store data. This app will allow us to manipulate the database using SQL queries.

# Technical issues

During my work on this project I had to reinstall MySQL as it seemed that it was not working.

# Database design

While designing my database I chose to go with a top down approach. This means I made sure to follow the rules of normalization when creating the database so its normalized rather than create one from scratch and normalizing it afterwards. The rules that I followed for the normalization are 1stNF, 2ndNF and 3rdNF.

I started by making a database prototype following the rules.

### Database Prototype

Booking

**BookingID**

CostumerID

FlightID

Costumer

**CostumerID**

TelNr

FName

LName

Email

Password

Payment

**InvoiceID**

PaymentMethod

*BookingID*

Aircraft Detail

**AircraftID**

AircraftModel

Capacity

PropretyOf

Flight

**FightID**

Departure

Arrival

Date

Time

Terminal

Gate

AircraftID

Price

Once I had this done, I went over and made sure it satisfied the following conditions:

1stNF

1.It only contains atomic values

2.There are no repeating groups

3.No two columns contain the same name

2ndNF

1.It satisfies the conditions of 1st normal form

2.All non-key attributes are fully dependent on the primary key

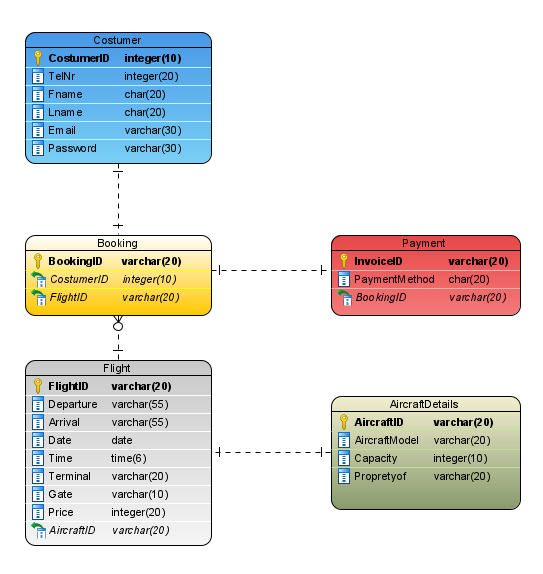
3rdNF

1.It satisfies the conditions of 2nd normal form

2.There are no transitive dependencies

Once I have validated my design to make sure that It meets all the conditions, I made the ER diagram and started writing the code script for the creating of the database and its tables.

### ER Diagram



[*Database Script at the bottom of the document*](#_Database_Script)

# Aplication Design

This app is a database with Java as its interface.

Designing this database requires us to first create an ER diagram normalize and validate it. Create the database and tables using MySQL.

The database contains the following tables that store information:

* Costumer - A costumer or user which has its details stored such as name and email .
* Booking - A costumer makes a flight booking and the detail of the booking are stored such as country of departure and destination.
* Payment - A costumer pays for the booking using their preferred method and payment details.
* Flight - Flight stores the information of the flight such as the terminal, time, and date.
* Aircraft Details - This holds the details of the plane such as the model, capacity and the airline who owns the plane.

This Java interface allows Java and MySQL to interact with each other with the use of Connector/J. Allowing us to modify the database using the Java interface.

# Plan/Log

November 29th: On the last day I have gone over my code and made sure all the functions work as intended and just cleaned up my code slightly by formatting my code. Once I was satisfied with my review on the java front end I when and updated the docx with all the new changes I made during the previous weeks such as updating the database create statement and redoing the ER diagram to reflect the updated database. I also finished writing up my review and conclusion and record my last entry in the log. Once this was done, I checked my docx to make sure there are no error or spelling mistake and checked to see if there was anything left to add. After finishing the document and I was happy with the result I uploaded my assignment.

November 27th: I have realized that my project must contain an insert statement for each table so I split the menu into 2 sections a user scenario and an admin scenario where you can add new aircrafts new flights and update aircraft details such as aircraft owner or capacity of an aircraft. During this day after some deliberating, I also made some changes to the database tables to better reflect what I intended while still being normalized.

November 21st: Today I worked on the code for the functions from the menu I made up last week. I was able to create a Sign-Up function for users that lets the users to create an account to be able to make bookings. After that I created a log in feature to allow users to see their flight tickets/bookings. And the last feature that I made for today was Flights which lists all the flights and asks you if u wish to make a booking and proceeds to ask you for details for the booking and before finalizing and moving to payments prompts you to log in or sign up and save the flight details you gave about your booking to your existing account or to the account you just created.

November 14th: I have started the front-end java and finished setting up a menu for all the available functions the program will facilitate. These functions are 1.SignUp 2.Log In 3.Flights

November 8th: I started by completing the application design, the conclusion and overview. At this point 90% of the work was done. I concluded this part of the project by going over my file document and formatting the text and checking that I had completed all the tasks I was supposed to do.

After Reviewing my document and was satisfied with the end result, I was 100% done with the part 1 of the project so I saved it and submitted it.

November 5th: Once MySQL was back and running I got to work on the database and table script code, after the code was done all that I had left was to finish some of the bullet points mentioned in the project brief.

October 31st: Got to work on ER diagram based on my prototype, once that was done I started working on MySQL script to create the database and the tables but I soon encountered an error with MySQL so I had to uninstall it and install it again. Due to this technical issue I was only able to finish my ER diagram.

October 24th: Started Project, Outlined the Contents of the project, started to prototype the tables and its application design, finished the planned work I had for today, roughly 20% of first part done,

Plan: For this project I decided to allocate 1 sessions of work per week until the deadline or until I finished the project and was satisfied with the result.

# Conclusion/Review

Looking back, I wish I had started 1 week ahead as my planned work schedule of 1 session a week of work fell apart the last week were, I worked 2 sessions. Where I left the last session the day of the submission in which I finished the remaining 10% of part1 of the project. If I started a week ahead, I would have been able to finish the document a few days ahead of the deadline. However since I was mostly finished by the last session I wasn’t in any rush, so I had ample type to go over my document and make sure everything was nicely presented and was able to make sure I covered all the points given in the project brief. I am also satisfied that until the last week before the deadline I was able to keep up with the schedule I made for myself to work on the project.

An important part of my process was reviewing my document which allowed me to catch mistakes that I did not notice beforehand when I made them.

It was also interesting looking into how Ryanair booking service on their website and coming up with my own prototype of it and crating its database filled with my rendition of what information the table might hold.

During my work on the java front-end I would come up with many different ideas on how to implement them and used a implement as I go mentality and because of that I think I wasted a lot of time on less important features when I could have spent more time implementing validation or to better structure my code. Due to the way I approached coding some parts are repeated and could have been turned into functions that I can call so that I could make the code more compact, readable, and faster. Another difficulty I encountered was naming the functions and variables, because I had no plan it was very easy to forget/mistake variables or even create duplicate that could conflict with each other and most variables naming scheme makes it hard to understand the purpose of the variable.

If I were to tackle of project like this again I would prototype how the front end would look like then I would devise a plan with proper steps and tasks laid out beforehand that I could tick off as I went along. I believe if I have done this from the very beginning not only would I have saved a lot of time that I could allocate to other tasks but I would have been able to write half as much code being more efficient and understandable.

# References

1. <https://www.ryanair.com/ie/en>
2. Database slides on Canvas
3. <http://www.oracle.com/technetwork/java/javase/jdbc/index.html>
4. <https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-overview.html>
5. <http://www.mysqltutorial.org/calling-mysql-stored-procedures-from-jdbc/>
6. <http://www.mysqltutorial.org/mysql-jdbc-tutorial/>

# Database Script

drop schema if exists R00192770RYANAIR;

create schema R00192770RYANAIR;

use R00192770RYANAIR;

CREATE TABLE AircraftDetails (

AircraftID VARCHAR(20) NOT NULL,

AircraftModel VARCHAR(20) NOT NULL,

Capacity INT(10) NOT NULL,

PropretyOf VARCHAR(20) NOT NULL,

PRIMARY KEY (AircraftID)

);

CREATE TABLE Booking (

BookingID VARCHAR(20) NOT NULL,

CostumerID int NOT NULL ,

FlightID VARCHAR(20) NOT NULL,

PRIMARY KEY (BookingID)

);

CREATE TABLE Costumer (

CostumerID int NOT NULL AUTO\_INCREMENT,

TelNr INT(20) NOT NULL,

FName CHAR(20) NOT NULL,

LName CHAR(20) NOT NULL,

Email VARCHAR(30) NOT NULL,

Password VARCHAR(30) NOT NULL,

PRIMARY KEY (CostumerID)

);

CREATE TABLE Flight (

FlightID VARCHAR(20) NOT NULL,

Departure varchar(55) NOT NULL,

Arrival VARCHAR(55) NOT NULL,

Date DATE NOT NULL,

Time TIME(6) NOT NULL,

Terminal VARCHAR(20) NOT NULL,

Gate VARCHAR(10) NOT NULL,

AircraftID VARCHAR(10) NOT NULL,

Price INT(20) NOT NUll,

PRIMARY KEY (FlightID)

);

CREATE TABLE Payment (

InvoiceID VARCHAR(20) NOT NULL,

PaymentMethod CHAR(20) NOT NULL,

BookingID VARCHAR(20) NOT NULL,

PRIMARY KEY (InvoiceID)

);

ALTER TABLE Flight ADD CONSTRAINT FKFlight571807 FOREIGN KEY (AircraftID) REFERENCES AircraftDetails (AircraftID);

ALTER TABLE Booking ADD CONSTRAINT FKBooking215882 FOREIGN KEY (FlightID) REFERENCES Flight (FlightID);

ALTER TABLE Payment ADD CONSTRAINT FKPayment52020 FOREIGN KEY (BookingID) REFERENCES Booking (BookingID);

ALTER TABLE Booking ADD CONSTRAINT FKBooking958784 FOREIGN KEY (CostumerID) REFERENCES Costumer (CostumerID);