

Airport Database Project - Group 12

Nicholas Wong

01 December 2024

# Table of Contents

<b>Table of Contents.....</b>	<b>2</b>
<b>Introduction.....</b>	<b>3</b>
<b>System Requirements.....</b>	<b>4</b>
<b>Conceptual Design.....</b>	<b>6</b>
<b>Logical Database Schema.....</b>	<b>11</b>
<b>Functional Dependencies and Normalization.....</b>	<b>14</b>
<b>Database System.....</b>	<b>19</b>
<b>Suggestions on Database Tuning.....</b>	<b>24</b>
<b>User Application Interface.....</b>	<b>25</b>
<b>Conclusions and Future Work.....</b>	<b>28</b>
<b>References.....</b>	<b>29</b>
<b>Appendix.....</b>	<b>30</b>

# **Introduction**

Airports located all over the world deal with immense amounts of data, especially those located in populated cities. A DBMS is necessary to handle all of the information that allows these airports to run smoothly. Our project follows the design of an airport, allowing for passengers to sign in, book/change flights, register luggage, etc. The database stores information regarding passengers, flights, luggage, boarding passes, and airlines. Our goal is to make storing, retrieving, and updating the required data simple while keeping the data consistent across all the tables. Then, passengers will be able to easily access information about their account and flights, as well as make any necessary changes which will be properly reflected in the database's tables.

This report follows the suggested section organization as provided in the Database Project Description, but where applicable, showcases comparisons and changes between our final product and initial concepts/schemas from earlier phases. This will be present in the listed sections below:

## **Conceptual Design**

## **Logical Database Schema**

## **Functional Dependencies and Normalization**

# System Requirements

System architecture diagram:



System requirements: The user must have the .html and .php files to open our website, as well as the SQL source code for our tables. Additionally, we use XAMPP to provide the interface and connection with the Apache server and MySQL database.

Database functional requirements:

- Store information on passengers, the flight(s) a passenger has booked, the flight details, luggage information, passenger boarding passes and luggage, and available airlines.

Initialized data should be consistent across tables.

- Retrieve data as requested by the client in the form of account details, flight information, and the boarding pass tied to their account.

- Update data in the form of a passenger changing their flight or registering luggage.

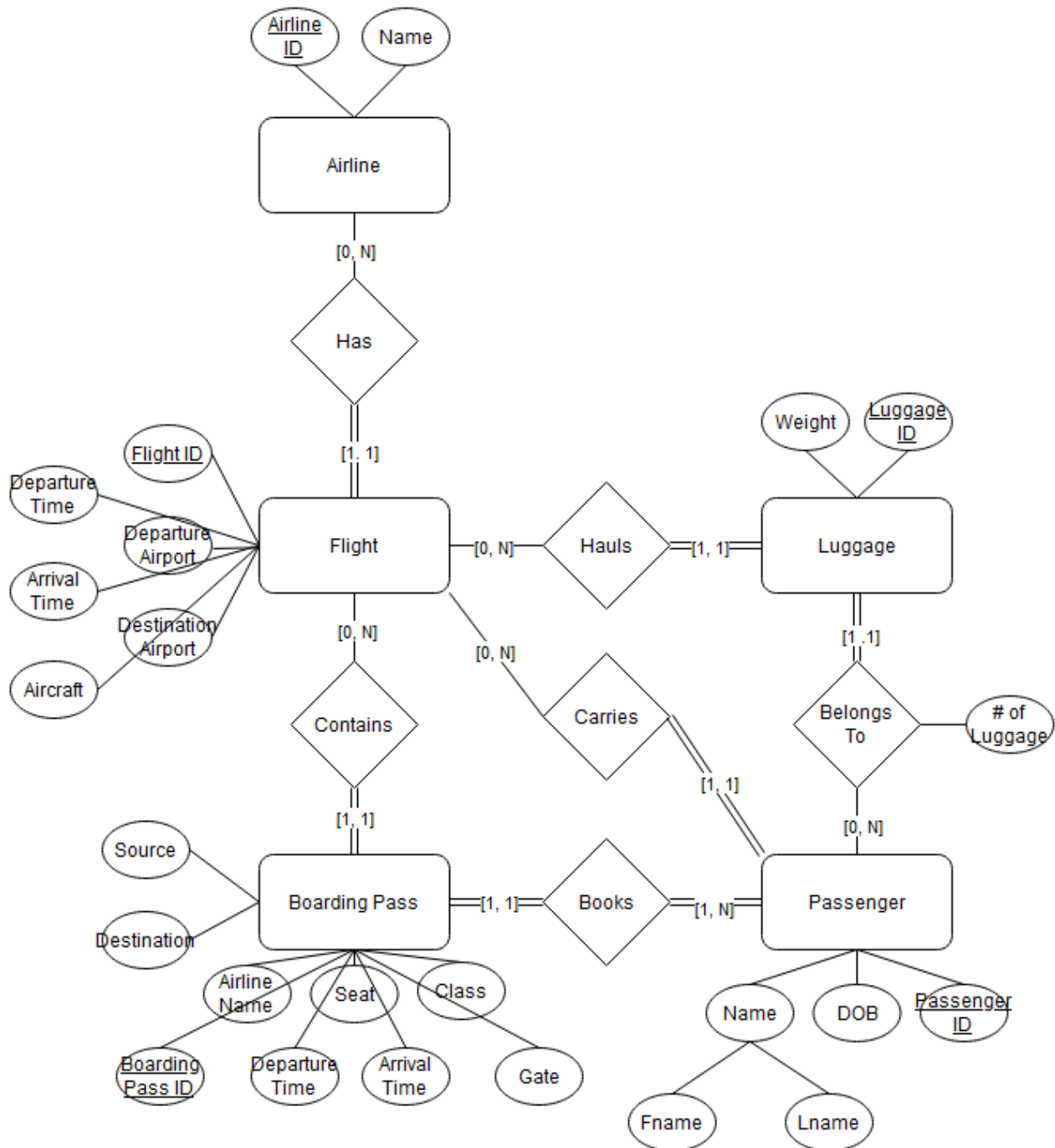
Updated data should be updated in every instance and be consistent across tables.

Database non-functional requirements:

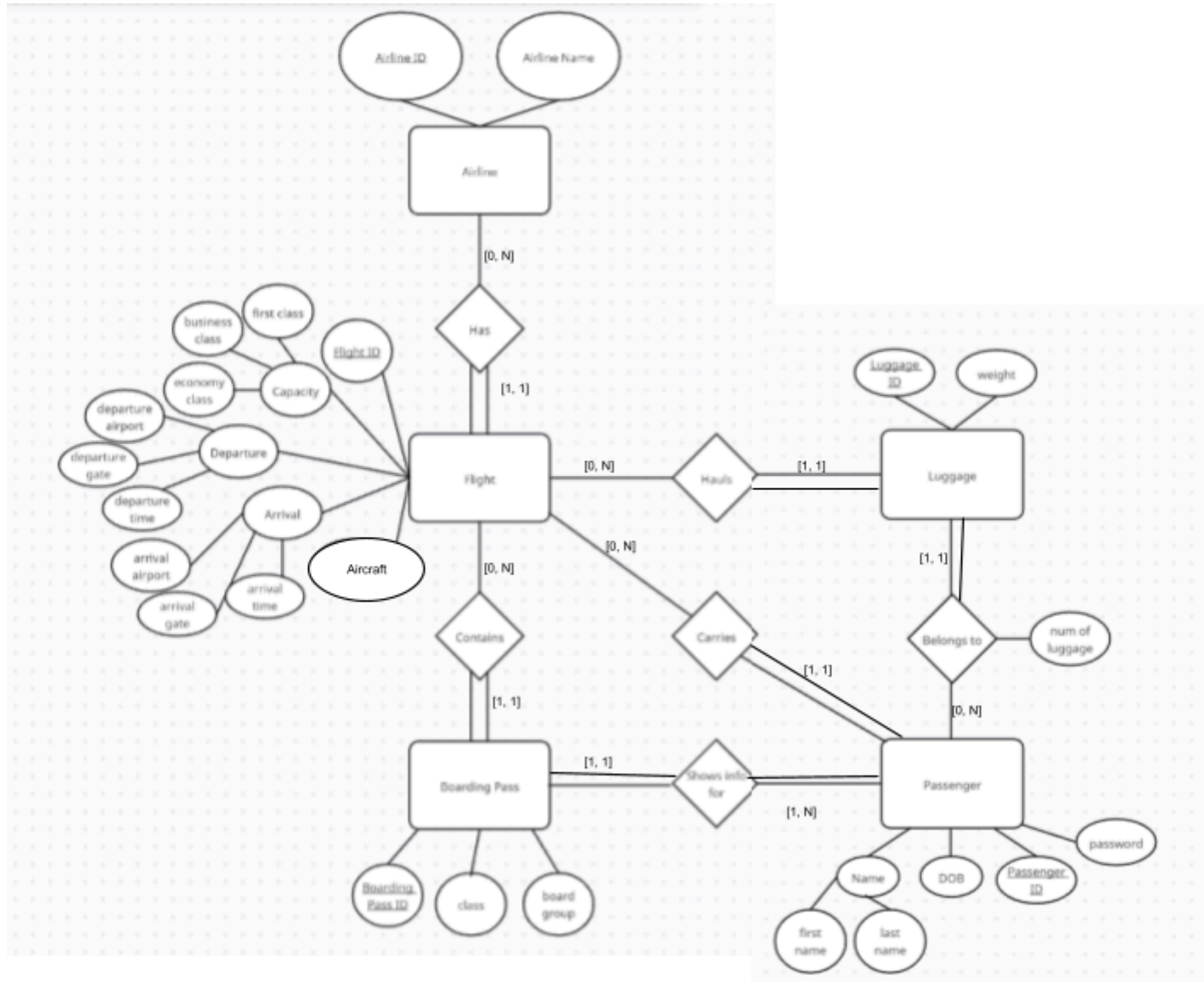
- Resistant toward SQL injection attacks
- Quick information retrieval from database
- Lack of data redundancy by utilizing the Third Normal Form for our database relations
- Simplicity of updating data
  - Users can update information via the “Change Flight” page
  - Admin can update information via SQL update statements

# Conceptual Design

Initial ER diagram from Phase 2:



Updated ER diagram to reflect final product (after normalization and practical changes):



**Data dictionary:**

## Passenger

Column	Data Type	Description	Default
last_name	VARCHAR	Last name of client	NULL
first_name	VARCHAR	First name of client	NULL
date_of_birth	DATE	Provides age of client	NULL
<u>passenger_id</u>	INTEGER	Randomly generated 8-digit unique ID given to a client	NOT NULL
password	VARCHAR	Client-chosen password linked to their account for login	NOT NULL

## Flight

Column	Data Type	Description	Default
flight_id	INTEGER	4-digit unique ID given to each distinct flight	NOT NULL
departure_airport	VARCHAR	Name of airport where flight is departing	NULL
departure_gate	VARCHAR	Gate of airport where flight is departing	NULL
departure_time	VARCHAR	Time at which flight is departing	NULL
arrival_airport	VARCHAR	Name of airport where flight is arriving	NULL



arrival_gate	VARCHAR	Gate of airport where flight is arriving	NULL
arrival_time	VARCHAR	Time at which flight is arriving	NULL
aircraft	VARCHAR	Name of aircraft used during the flight	NULL
first_capacity	INTEGER	Number of total first class seats available on the plane	10
business_capacity	INTEGER	Number of total business class seats available on the plane	20
economy_capacity	INTEGER	Number of total economy class seats available on the plane	120

### Boarding Pass

Column	Data Type	Description	Default
boarding_pass_id	INTEGER	7-digit unique ID for each boarding pass	NOT NULL
class	VARCHAR	Seat class that the passenger has booked	'Economy'
board_group	CHAR	Ordered group that determines when the passenger boards the plane; tied to seat class	'3'

## Luggage

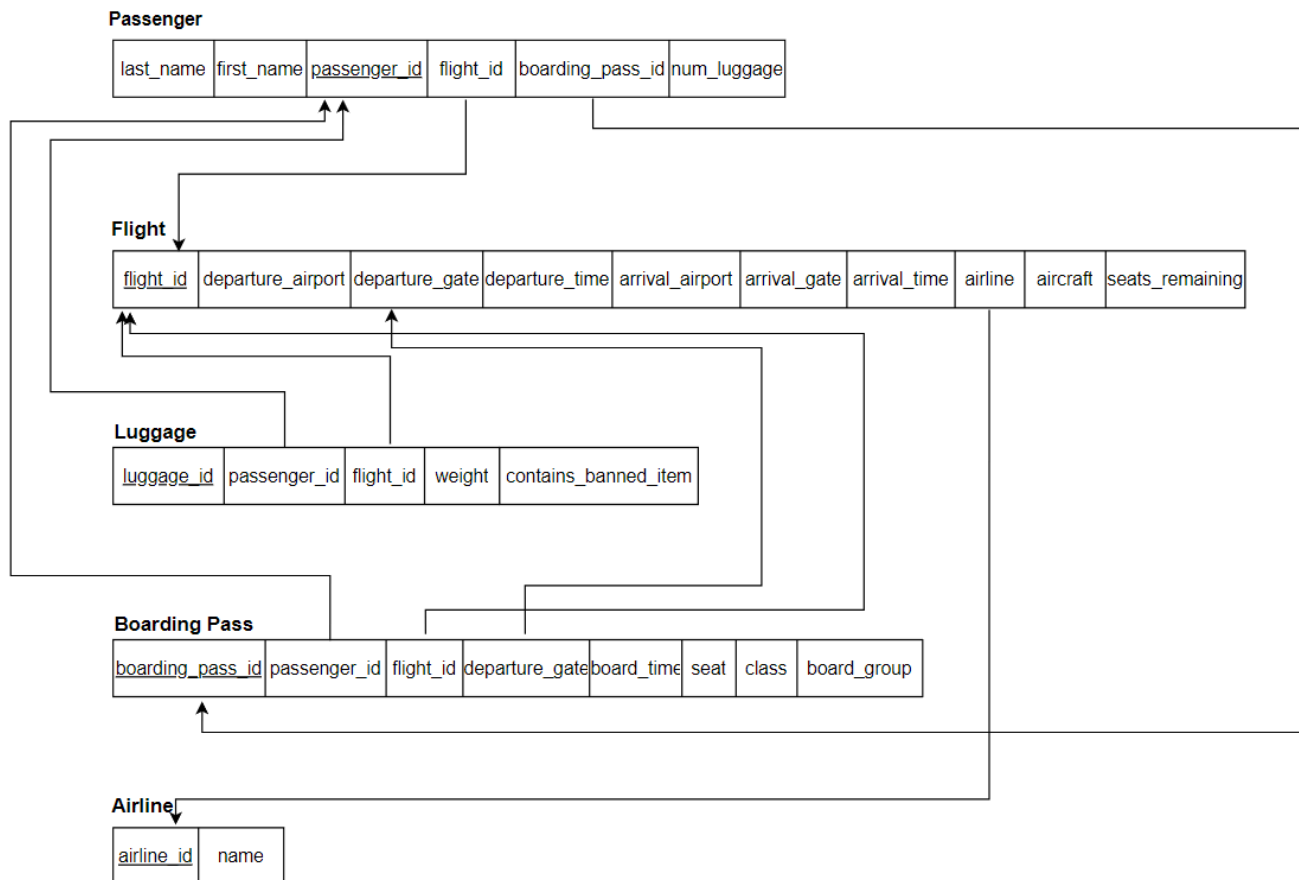
Column	Data Type	Description	Default
luggage_id	INTEGER	7-digit unique ID for each distinct unit of luggage	NOT NULL
weight	FLOAT	Weight of given luggage in lbs	50.0

## Airline

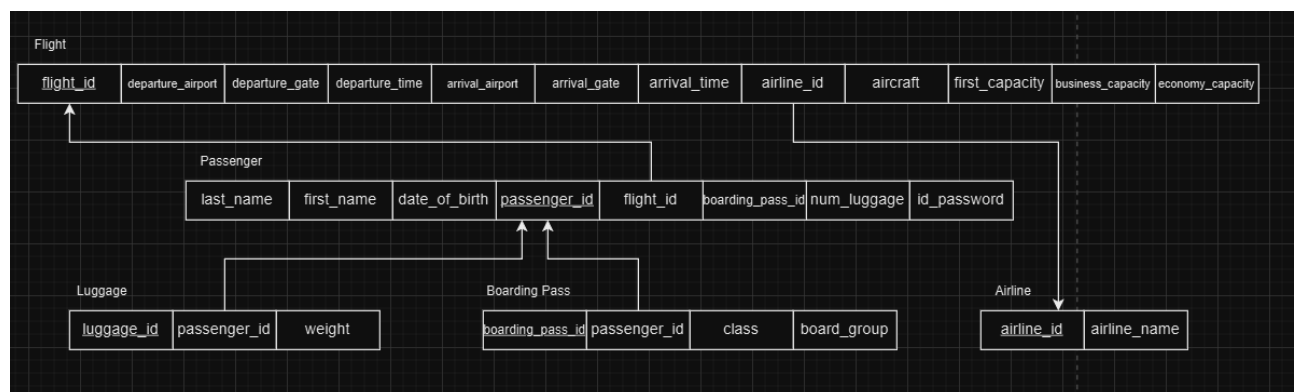
Column	Data Type	Description	Default
airline_id	VARCHAR	2-char unique ID indicating the airline providing the given flight	NOT NULL
airline_name	VARCHAR	Name of the airline providing the given flight	NULL

# Logical Database Schema

Schema based on original ER diagram from Phase 2:



Updated schema to reflect final product (after normalization and practical changes):



SQL code for construction (given order is necessary to avoid errors):

```
1 • ⊖ CREATE TABLE Airline (  
2     airline_id VARCHAR(2) NOT NULL,  
3     airline_name VARCHAR(255) DEFAULT NULL,  
4     PRIMARY KEY (airline_id)  
5 );  
6  
7 • ⊖ CREATE TABLE Flight (  
8     flight_id INTEGER NOT NULL,  
9     departure_airport VARCHAR(255) DEFAULT NULL,  
10    departure_gate VARCHAR(255) DEFAULT NULL,  
11    departure_time VARCHAR(255) DEFAULT NULL,  
12    arrival_airport VARCHAR(255) DEFAULT NULL,  
13    arrival_gate VARCHAR(255) DEFAULT NULL,  
14    arrival_time VARCHAR(255) DEFAULT NULL,  
15    airline_id VARCHAR(2) NOT NULL,  
16    aircraft VARCHAR(255) DEFAULT NULL,  
17    first_capacity INTEGER DEFAULT 10,  
18    business_capacity INTEGER DEFAULT 20,  
19    economy_capacity INTEGER DEFAULT 120,  
20    PRIMARY KEY (flight_id),  
21    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)  
22        ON DELETE CASCADE ON UPDATE CASCADE  
23 );
```

```
25 • CREATE TABLE Passenger (  
26     last_name VARCHAR(255) DEFAULT NULL,  
27     first_name VARCHAR(255) DEFAULT NULL,  
28     date_of_birth DATE DEFAULT NULL,  
29     passenger_id INTEGER NOT NULL,  
30     flight_id INTEGER NOT NULL,  
31     boarding_pass_id INTEGER NOT NULL,  
32     num_luggage INTEGER DEFAULT 0,  
33     id_password VARCHAR(255) NOT NULL,  
34     PRIMARY KEY (passenger_id),  
35     FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)  
36         ON DELETE CASCADE ON UPDATE CASCADE  
37 );  
  
39 • CREATE TABLE Boarding_Pass (  
40     boarding_pass_id INTEGER NOT NULL,  
41     passenger_id INTEGER NOT NULL,  
42     class VARCHAR(8) DEFAULT 'Economy',  
43     board_group CHAR DEFAULT '3',  
44     PRIMARY KEY (boarding_pass_id),  
45     FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)  
46         ON DELETE CASCADE ON UPDATE CASCADE  
47 );  
  
49 • CREATE TABLE Luggage (  
50     luggage_id INTEGER NOT NULL,  
51     passenger_id INTEGER NOT NULL,  
52     weight FLOAT DEFAULT 50.0,  
53     PRIMARY KEY (luggage_id),  
54     FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)  
55         ON DELETE CASCADE ON UPDATE CASCADE  
56 );
```

## Functional Dependencies and Normalization

The following processes shown will identify the normalization of our initial tables into 3NF if they needed to be normalized. If no functional dependencies were identified in the original table, it will be stated and the single table given represents the 3NF form. Following each entity is the SQL code required to create its normalized table.

### Passenger

last_name	first_name	date of birth	<u>passenger_id</u>	flight_id	boarding_pass_id	num_luggage	id_password
-----------	------------	---------------	---------------------	-----------	------------------	-------------	-------------

No Functional Dependencies (passenger\_id gives every attribute in this table aside from the foreign key flight\_id)

```

25 ● ○ CREATE TABLE Passenger (
26     last_name VARCHAR(255) DEFAULT NULL,
27     first_name VARCHAR(255) DEFAULT NULL,
28     date_of_birth DATE DEFAULT NULL,
29     passenger_id INTEGER NOT NULL,
30     flight_id INTEGER NOT NULL,
31     boarding_pass_id INTEGER NOT NULL,
32     num_luggage INTEGER DEFAULT 0,
33     id_password VARCHAR(255) NOT NULL,
34     PRIMARY KEY (passenger_id),
35     FOREIGN KEY (flight_id) REFERENCES Flight(flight_id)
36         ON DELETE CASCADE ON UPDATE CASCADE
37 );

```

**Flight**

<u>flight_id</u>	departure_airport	departure_gate	departure_time	arrival_airport	arrival_gate
------------------	-------------------	----------------	----------------	-----------------	--------------

**Flight Cont.**

arrival_time	airline_id	aircraft	first_capacity	business_capacity	economy_capacity
--------------	------------	----------	----------------	-------------------	------------------

No Functional Dependencies (flight\_id gives every attribute in this table)

```

7 • CREATE TABLE Flight (
8     flight_id INTEGER NOT NULL,
9     departure_airport VARCHAR(255) DEFAULT NULL,
10    departure_gate VARCHAR(255) DEFAULT NULL,
11    departure_time VARCHAR(255) DEFAULT NULL,
12    arrival_airport VARCHAR(255) DEFAULT NULL,
13    arrival_gate VARCHAR(255) DEFAULT NULL,
14    arrival_time VARCHAR(255) DEFAULT NULL,
15    airline_id VARCHAR(2) NOT NULL,
16    aircraft VARCHAR(255) DEFAULT NULL,
17    first_capacity INTEGER DEFAULT 10,
18    business_capacity INTEGER DEFAULT 20,
19    economy_capacity INTEGER DEFAULT 120,
20    PRIMARY KEY (flight_id),
21    FOREIGN KEY (airline_id) REFERENCES Airline(airline_id)
22        ON DELETE CASCADE ON UPDATE CASCADE
23 );

```

**Luggage (initial)**

<u>luggage_id</u>	passenger_id	flight_id	weight
-------------------	--------------	-----------	--------

Functional Dependencies:

passenger\_id -> flight\_id

This table is in 2NF because the only key has one attribute, so there are no partial dependencies.

This table is not in 3NF because there is a transitive dependency luggage\_id -> passenger\_id -> flight\_id

We can split this into: R1(luggage\_id, passenger\_id, weight) & R2(passenger\_id, flight\_id).

R2 is already covered by the **Passenger** table, so all we have is a new refactored Luggage table:

### **Luggage (final)**

<u>luggage_id</u>	passenger_id	weight
-------------------	--------------	--------

Now there are no functional dependencies (luggage\_id gives every attribute in this table)

```

49 • ○ CREATE TABLE Luggage (
50     luggage_id INTEGER NOT NULL,
51     passenger_id INTEGER NOT NULL,
52     weight FLOAT DEFAULT 50.0,
53     PRIMARY KEY (luggage_id),
54     FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
55     ON DELETE CASCADE ON UPDATE CASCADE
56 );

```

### **Boarding Pass (initial)**

<u>boarding_pass_id</u>	passenger_id	flight_id	departure_gate	departure_time	seat	class	board_group
-------------------------	--------------	-----------	----------------	----------------	------	-------	-------------

Functional Dependencies:

passenger\_id -> flight\_id

flight\_id -> departure\_gate, departure\_time

This table is in 2NF because the only key has one attribute, so there are no partial dependencies.

This table is not in 3NF because there are transitive dependencies:



TD1) boarding\_pass\_id -> passenger\_id -> flight\_id

TD2) boarding\_pass\_id -> flight\_id -> departure\_gate, departure\_time

Step by step we can factor the table:

TD1: R -> R1(boarding\_pass\_id, passenger\_id, seat, class, board\_group) & R2(passenger\_id, flight\_id, departure\_gate, departure\_time)

TD2: R2 -> R21(passenger\_id, flight\_id) & R22(flight\_id, departure\_gate, departure\_time)

R21 is already covered by the **Passenger** table, and R22 is already covered by the **Flight** table, so all we have is a new refactored **Boarding Pass** table:

#### **Boarding Pass (final)**

<u>boarding_pass_id</u>	passenger_id	class	board_group
-------------------------	--------------	-------	-------------

**Note:** The deletion of the 'seat' attribute was for practical purposes and not a result of the normalization to 3NF.

Now there are no functional dependencies (boarding\_pass\_id gives every attribute in this table aside from the foreign key passenger\_id)

```

39 ● ○ CREATE TABLE Boarding_Pass (
40     boarding_pass_id INTEGER NOT NULL,
41     passenger_id INTEGER NOT NULL,
42     class VARCHAR(8) DEFAULT 'Economy',
43     board_group CHAR DEFAULT '3',
44     PRIMARY KEY (boarding_pass_id),
45     FOREIGN KEY (passenger_id) REFERENCES Passenger(passenger_id)
46         ON DELETE CASCADE ON UPDATE CASCADE
47 );

```

**Airline**

<u>airline_id</u>	airline_name
-------------------	--------------

No Functional Dependencies (airline\_id gives every attribute in this table)

```
1 • ⊖ CREATE TABLE Airline (  
2     airline_id VARCHAR(2) NOT NULL,  
3     airline_name VARCHAR(255) DEFAULT NULL,  
4     PRIMARY KEY (airline_id)  
5     );  
6
```

# Database System

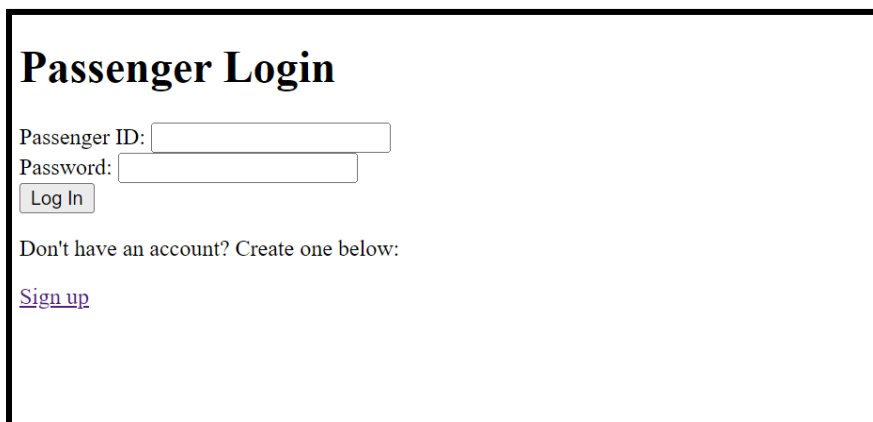
Our database system utilizes MySQL as the database and XAMPP to provide a graphical interface for our website through the Apache HTML server.

After installing MySQL, its applications, XAMPP, and the necessary SQL, HTML, and PHP code found in the .zip file in the 'Appendix' section of this report, use XAMPP to connect to MySQL and Apache.

## System walkthrough:

For this walkthrough, we will assume the identity of a new user that is not currently registered with the airport.

1. From the login page, access the signup page using the provided link.

A screenshot of a web form titled "Passenger Login". It contains two input fields: "Passenger ID:" and "Password:". Below the password field is a "Log In" button. At the bottom of the form, there is a text link that says "Don't have an account? Create one below:" followed by a blue underlined link labeled "Sign up".

**Passenger Login**

Passenger ID:




Password:

Don't have an account? Create one below:

[Sign up](#)




2. Register for an account and flight using the signup page.

## Passenger Sign Up

First Name:   
Last Name:   
Date of Birth:    
Select a Flight:    
Select Seat Class:    
Password:

[Return to Log in](#)

## Passenger Sign Up

First Name:   
Last Name:   
Date of Birth:    
Select a Flight:    
Select Seat Class:    
Password:

[Return to Log in](#)

Account created successfully! Your Passenger ID: 28882128, Boarding Pass ID: 2111138.

## Passenger Sign Up

First Name:

Last Name:

Date of Birth:

Select a Flight:

Select Seat Class:

Password:

[Return to Log in](#)

3. Return to the login page using the provided link and log in with the unique passenger ID given to you and the password you chose to view your dashboard.

## Passenger Login

Passenger ID:

Password:

Don't have an account? Create one below:

[Sign up](#)

## Welcome, Chris Water

### Your Flight Details

Flight Number: 2111

Departure: DFW at 1230

Departure Gate: C12

Arrival: LAX at 1545

Arrival Gate: D12

Airline: UA

[Boarding Pass](#)

[Change Flight](#)

[Log out](#)

4. (Optional) Access your boarding pass information using the “Boarding Pass” link to view it or to register your luggage. You may also change your flight using the “Change Flight” link.

## Hello Passenger ID: 28882128

### Your Boarding Pass Details

Boarding Pass ID: 2111138

Class: Economy

Boarding Group #: 3

Flight ID: 2111

Departure: Gate C12 at 1230

Arrival: Gate D12

Airline: UA

You currently have no luggage added.

### Add Luggage

Luggage Weight (lbs):

[Return To Dashboard](#)

## Available Flights

Flight ID	Origin	Destination	Departure Time
1111	DFW	JFK	0415
2111	DFW	LAX	1230
3111	DFW	LAX	1300

[Dashboard](#)

## Change Your Flight

Passenger ID:

Password:

New Flight ID:

5. After you perform all the actions desired, you may log out of your account via the dashboard.

## Suggestions on Database Tuning

Possible improvements to our database:

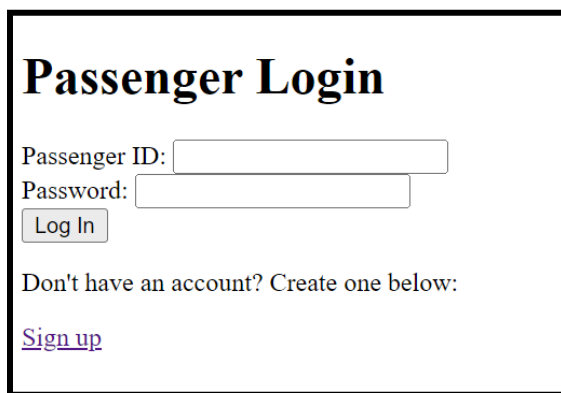
- Indexing: At the time of our live demonstration, there were no attempts to utilize indexing in our database to improve data retrieval. Given the structure of our database, we could use dense indexing on `flight_id`. This way, all passengers on the same flight would be sorted together, which is a far more intuitive organization for this database and would improve query retrieval times.
- User design and capabilities: The current design of our DBMS might be too client-focused. While our website and database design is useful for passengers and covers most scenarios of what a passenger might want to do within the context of an airport, we could provide means for employees or administrative figures of the airport to access and update information. For example, given a flight delay, the airport should publish the updated times for departure and arrival of any affected flights. Passengers obviously cannot make this change, but we do not have a dedicated employee login.
- Queries: When creating tables via SQL create statements, we relied heavily on default values to contain values within the bounds of what was practical. We could have supplemented this with CHECK statements to more clearly define bounds for values, such as the various capacity attributes in the Flight table, which we want to have a set value for.



## User Application Interface

The system user interface we developed was created using HTML and PHP. Our website is comprised of 5 different pages. 4 of these pages utilize forms to facilitate interaction between the user and our database. All pages perform some degree of data retrieval and 3 include some way to manipulate data.

**Login:** For users that already have an account stored in our database, they can use their unique passenger ID and password to login and see their dashboard. For users who have yet to register with us, a link to the sign-up form page is provided.

A screenshot of a web form titled "Passenger Login". The form is enclosed in a black rectangular border. It contains the following elements: the title "Passenger Login" in a large, bold, black serif font; a label "Passenger ID:" followed by a white rectangular input field; a label "Password:" followed by a white rectangular input field; a "Log In" button with a grey gradient and black text; and a text link "Don't have an account? Create one below:" followed by a blue, underlined "Sign up" link.

**Passenger Login**

Passenger ID:

Password:

Don't have an account? Create one below:


[Sign up](#)


**Signup:** New users can fill in the required information and book a flight and their account and flight information will be stored in our database. A message after signing up will confirm a successful registration and the user can then return to the login page.


## Passenger Sign Up

First Name:

Last Name:

Date of Birth:  

Select a Flight:  

Select Seat Class:  

Password:

[Return to Log in](#)

**Dashboard:** The dashboard page serves as the primary data retrieval page. Using the login information provided by the user through the login page, all relevant account and flight information to the user is retrieved and displayed.

## Welcome, Peter Pan

### Your Flight Details

Flight Number: 1111

Departure: DFW at 0415

Departure Gate: A5

Arrival: JFK at 0715

Arrival Gate: B5

Airline: DL

[Boarding Pass](#)

[Change Flight](#)

[Log out](#)

**Boarding Pass:** The boarding pass page gives more detail regarding the passenger's booked flight. The passenger may register their luggage through this page.

## Hello Passenger ID: 25656510

### Your Boarding Pass Details

Boarding Pass ID: 1111127

Class: First

Boarding Group #: 1

Flight ID: 1111

Departure: Gate A5 at 0415

Arrival: Gate B5

Airline: DL

You currently have no luggage added.

### Add Luggage

Luggage Weight (lbs):

[Return To Dashboard](#)

**Change Flight:** This page allows passengers to switch the flight they are on. The page displays the available flights to choose from and the forms that will update the database with the new information.

### Available Flights

Flight ID	Origin	Destination	Departure Time
1111	DFW	JFK	0415
2111	DFW	LAX	1230
3111	DFW	LAX	1300

[Dashboard](#)

### Change Your Flight

Passenger ID:

Password:

New Flight ID:

## Conclusions and Future Work

As a team, we feel that we were able to deliver a satisfactory database system and demonstrate its capabilities proficiently. Admittedly, however, our original vision when we first conceptualized this airport database system included more features, such as the attribute ‘seat’ in the Boarding Pass class, and a way to choose a specific seat on the plane during booking. However, with limited time, we felt that it was not necessary given that not all airlines allow for seat selection, though this may be an update to make in the future.

In regards to feedback about this project, we as a team believe that most of this project is relatively straightforward and follows the content discussed in lectures well on the database side. However, in regards to HTML and PHP, the lecture notes did not feel sufficient in providing us the knowledge required to create the website, and much of this was self-taught by our members. Additionally, we believe that it would be helpful to go over or suggest possible applications for connecting the website and database. Our group simply found XAMPP by searching, while others might use Next.js, but giving a brief introduction on any valid option would save students the time of struggling with an unknown application.

## References

Our database project did not require the use of published references or books. Instead of searching for real airport data, we decided to generate our own sample data to work with. This allowed us to create data that fit our database's tables without the need to perform data cleaning and trim excess data columns outside of our project's reach that published data might have.

## Appendix

The .zip file containing the requested documents and source code for our database project is available via the Google Drive link below:

Please note that there is no appendix in the final report in the .zip file due to this document and the Google Drive link containing each other, disallowing for both to have the correct version of each other.

[https://drive.google.com/file/d/1UIm1dBroulOan-lEeoiz\\_QEtY5hV6G/view?usp=drive\\_link](https://drive.google.com/file/d/1UIm1dBroulOan-lEeoiz_QEtY5hV6G/view?usp=drive_link)