# Technical Documentation

# Introduction

Queen Alia airport is the best local airport in Jordan and one of the best in the world. This airport introduces many services to the passengers in the airport. This airport serves many passengers around the world.

The airport system consists of information about passengers, flight, plane, boarding and the airlines in the airport. So, the system will offer an interface for each of the previous stakeholders to help them manage the information inside the airport professionally which will help the passenger to reach his destination easily and proficiently.

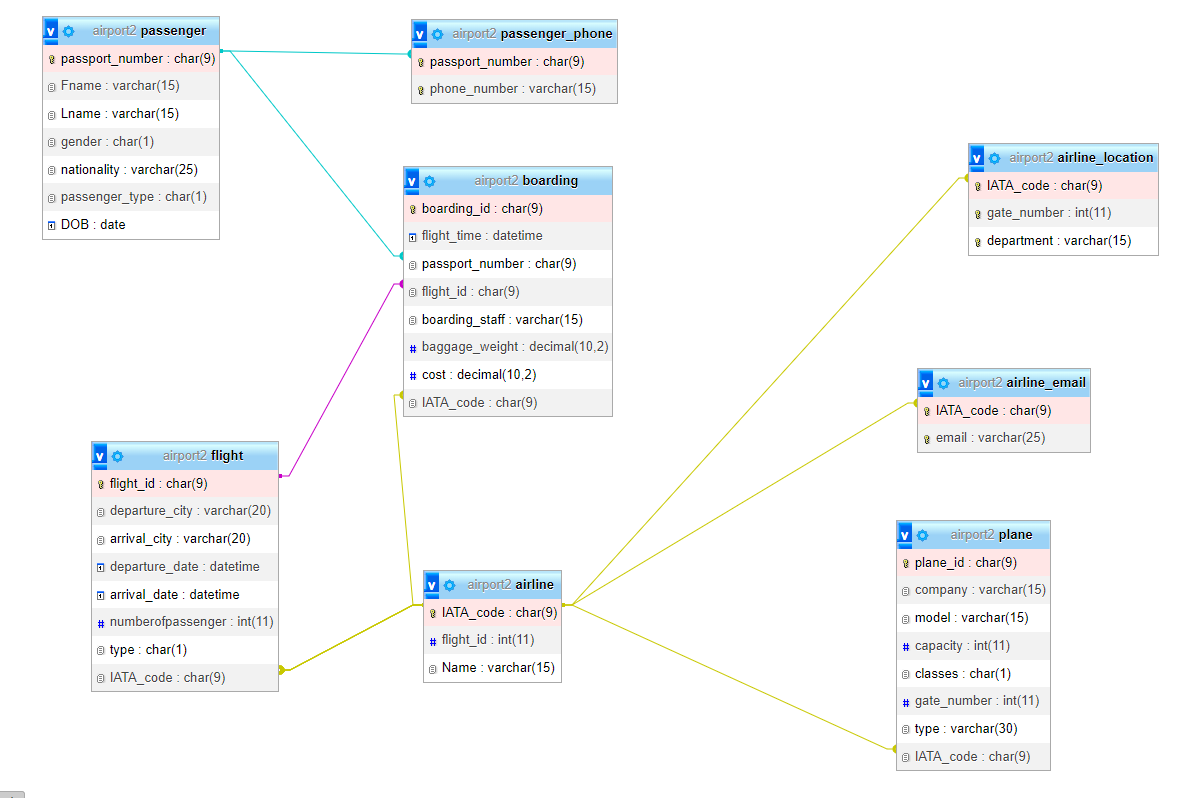
The scenario for the passenger is. There are many passengers will visit the airport, each passenger could make many boardings, and each boarding will be for one passenger. There is many boarding for many passengers in one flight, each flight is for one airline, but the airline contains multiple flights. Each airline has many planes, and each plane is for one airline. Each boarding is managed by one airline, but the airline could manage many boarding.

Each user of the previous scenario will use the system to manage and operate his own information to make It easier to the passenger to walk step by step in a well-designed structure from his arrival until he leaves the airport.

So, the passenger should use his passport id and his name to make a boarding. The boarding contains many information such as the unique id and many details about the flight and the passenger. The flights contain many boardings and each flights contains information such as the flight id, plane\_id and each flight are for one airline, but the airline contain multiple flight and the airline has a location, IATA\_code, email and each airline contain multiple planes and each plane which has a model, id, classes is for one airline. The system will be beneficial to store this information to make manageable.

In this part of project, I tried to make a database that links all the data together technically and that is by using the my sql to make the database. I also made the actual interface that the users will use in managing the data in the database and gave the permission depending on the user and system requirements that we talked about the previous part of the project. I made the entities and linked them together and applied the testing and the building of the actual project that could move the project from the plan and prototype to the actual usage of the data that is going to be used in the airport to manage all the operations technically and effectively.

# Physical Schema



# Database Development

## Database Overview

|  |  |  |
| --- | --- | --- |
| **Table** | **Name** | **Description** |
|  | passenger | Each one of the passengers have Fname and Lname and gender and nationality and passenger\_type and DOB and these attributes are very important to be existed in the passenger table. The passport\_number in this table is the primary key for any other attribute in this table |
|  | flight | Each flight has flight\_id and departure\_city and arrival\_city and departure\_date and plane\_id and numberofpassenger and type and IATA\_code and these attributes are very important to be existed in the flight table. This table have the flight\_id as a primary key for any other information in the table and because the relation between this table and the airline and tables is one to many so it has foreign key form that table which is and IATA\_code |
|  | boarding | Each boarding has boarding\_id, flight\_time, passport\_number, flight\_id, boarding\_staff, baggage\_weight, cost, and IATA\_code and these attributes are very important to be existed in the flight table. This table have the boarding\_id as a primary key and because the relation between passenger and flight and airline one to many so this table have foreign key form each one of them which are passport\_number, flight\_id, IATA\_code |
|  | Passenger\_phone | This table have the passport number and phone number because the phone number is multi valued and this table and this have candidate key |
|  | plane | This table have the plane\_id, company, model, capacity, classes, gate\_number, type, and IATA\_code. This table have the plane\_id as primary key and have the IATA\_code as foreign key |
|  | airline | This table have IATA\_code and flight\_id, Fname, Lname and the IATA\_code is the primary key |
|  | Airline\_location | This table have just IATA\_code and gate\_number and department because the location is composite value, and all the attributes are candidate keys |
|  | Airline\_email | This table have IATA code and email because the email is multivalued, and all the attributes are candidate keys |

|  |  |  |
| --- | --- | --- |
| **View** | **Name** | **Description** |
|  | PassengerInfo | This view combine data from two tables passenger and passenger\_phone which helps In providing information about passengers including their personal details and contact information phone numbers and it includes passport\_number, Fname, Lname, gender, nationality, passenger\_type, DOB and phone\_number and the passport\_number is the primary key to identify passengers.  I also made join and the view provide left join between the passenger and passenger phone tables and the connection between them based passport\_number. This view makes it easier to access the passenger information |
|  | FlightInfo | This view has combination between two tables which are flight and airline, and this view is designed to provide details about the flight and the name of the airline that organize the flight. This includes flight\_id, departure\_city, arrival\_city, departure\_date, arrival\_date, numberpfpassenger, type and the airline\_name. The flight\_id is the primary key to catch any flight. I also made a left join between flight and airline tables to connect them based on IATA\_code and this ensures that the flight information is included even if there are no matching airline record. This view is helpful in access any flight record easily because it simplifies the retrieval of flight data by combining it from multiple tables into a single view |
|  | BoardingInfo | The view combine data from three tables the boarding, passenger and flight and this view have boarding\_id, flight\_time, passport\_number, Fname, Lname, flight\_id, departure\_city and arrival\_city and the boarding\_id serves as the primary key to identify boarding events and the view include details about the boarding events such as the date and time boarding, passenger information. The join that is used is the inner join to connect both passenger and flight tables. The inner join include includes records for which there are matching entries in both the passenger and flight tables and records without matching passengers or flights will not appear in the view. And this view is used to simplify the retrieval of data related to passenger boarding making it easier to analyse passenger activity and flight information which is used in applications require view of boarding events and associated passenger and flight details. |
|  | LocationInfo | This view combine data from two tables the airline location and the airline tables and this view include the IATA\_code, gate\_number\_ department, airline\_name and the IATA\_code and gate\_name is composite primary key to identify the location and the inner join links them based on the matching IATA\_code and this ensures that only locations with matching airlines are included in the view. This view is useful in applications that consolidated information about airline location and the airline using those location |

|  |  |  |
| --- | --- | --- |
| **Procedure** | **Name** | **Description** |
|  | GetPassengerFlightInfo | In this procedure it is expected to get a single input which passportNum that represents the passenger’s passport number and then it retrieves flight information for that passenger. I used join operations to make a connection between passenger, boarding and flight tables based on matching passport numbers and flight Ids. This procedure returns information about passenger’s flights including their first name, last name, flight id, departure city, arrival city, departure date and arrival date. So, this is used to provide easy process of retrieving flight details for a particular passenger. |
|  | InsertBoarding | This procedure is designed to insert a new boarding record inro the boarding entity. It takes several parameters representing the attributes of a boarding record including boardingid, flighttime, passportNum, flightId, boardingStaff, baggageWeight, boardigCost and IATACode.  SO, I used these parameters to make a new record in the boarding table, so I used these values for each parameter to insert a new boarding record into the database. |
|  | GetPassengerInfoPassport | This takes the passport number as an input parameter and retrieve specific passenger information from the passenger table where the passport number matches the input. I used where clause to filter the results to include the rows where passport number matches the input parameters and to use this procedure, we can use this query 🡪 CALL GetPassengerInfoByPassport (‘The passport number’) |
|  | GetFlightsByAirline | This procedure retrieves flight information for all flights operated by a specific airline based on the airline IATA code. It takes airlineIATACode as an input parameter and returns details about the flight. I used join operation to connect the flight and airline tables based on matching IATA code and it retrieves (flight\_id, departure\_city, arrival\_city, departure\_date, arrival\_date) and I used also where clause to filter the result to include the flight operated by the airline that have the IATA code that matches the input parameter. I can use this procedure by using this query 🡪 CALL GetFlightsByAirline (‘Airline IATA code’) |

## Security

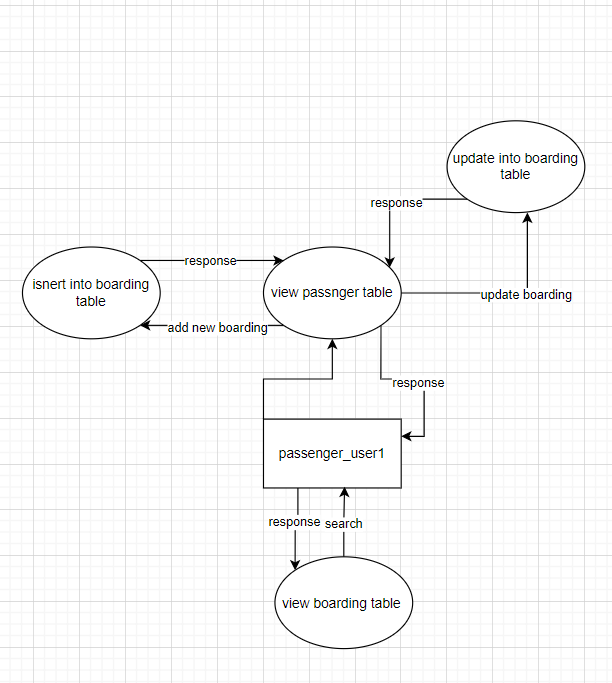
|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Privilege Command** | **Description** | **Screenshot** |
| Passenger  Flight manager | 1. GRANT SELECT ON airport2.passenger TO 'passenger\_user'@'localhost'; 2. GRANT INSERT ON airport2.boarding TO 'passenger\_user'@'localhost'; 3. GRANT SELECT ON airport2.boarding TO 'passenger\_user1'@'localhost'; 4. GRANT UPDATE ON airport2.boarding TO 'passenger\_user1'@'localhost'; | 1. This is used to allow passenger to access the system using their passport information. 2. So, the passenger is allowed to insert row into boarding table. 3. This allows passenger to view the boarding table. 4. This allows user to update table in boarding table |  |
| 1. GRANT UPDATE ON airport2.flight TO 'flight\_manager'@'localhost'; 2. GRANT INSERT ON airport2.boarding TO 'flight\_manager'@'localhost'; 3. GRANT SELECT ON airport2.boarding TO 'flight\_manager'@'localhost'; 4. GRANT UPDATE ON airport2.boarding TO 'flight\_manager'@'localhost'; | 1. This allows user to update in the flight table. 2. This allows the flight manager to insert data into boarding table. 3. This allows the flight manager to read data in boarding. 4. This allows flight manager to update table row in boarding table |  |
| Airport staff | GRANT SELECT,INSERT,UPDATE,DELETE ON airport2.flight TO 'airport\_manager'@'localhost';  GRANT SELECT,UPDATE ON airport2.boarding TO 'airport\_manager'@'localhost';  GRANT SELECT ON airport2.BoardingInfo TO 'airport\_manager'@'localhost';  GRANT SELECT ON airport2.LocationInfo TO 'airport\_manager'@'localhost';  GRANT SELECT ON airport2.FlightInfo TO 'airport\_manager'@'localhost'; | The airport\_manager could view and insert and update and delete on the flight table and also he can view and update the boarding table and he could also view some views that I made before which are the boardingInfo, LocationInfo, FlightInfo from his interface. |  |
| 1. GRANT SELECT ON airport2.boarding TO 'staff\_airport'@'localhost'; 2. GRANT SELECT ON airport2.flight TO 'staff\_airport'@'localhost'; 3. GRANT SELECT ON airport2.passenger TO 'staff\_airport'@'localhost'; 4. GRANT UPDATE ON airport2.boarding TO 'staff\_airport'@'localhost'; 5. GRANT INSERT ON airport2.boarding TO 'staff\_airport'@'localhost’; | 1. This allows the staff to read data of the boarding table. 2. This allows the staff to read the flight table. 3. This allows staff to read the passenger table. 4. This allows staff to update table row. 5. This allows staff to insert row into boarding table. |  |

## User Interface

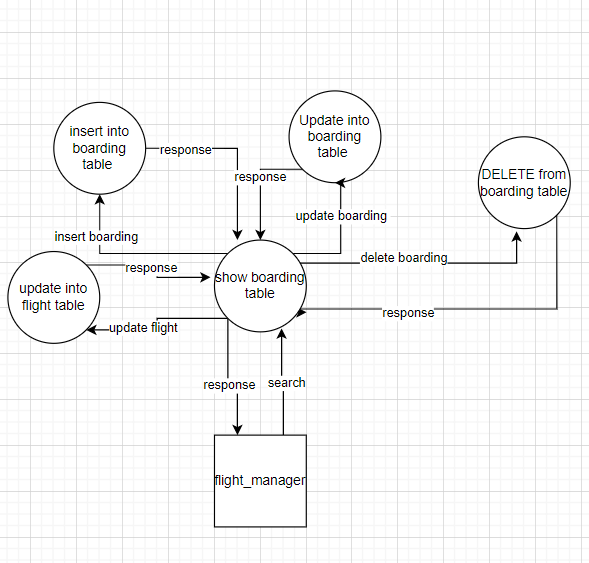
### Flowchart and Data Movement Diagrams

Data movement Diagram

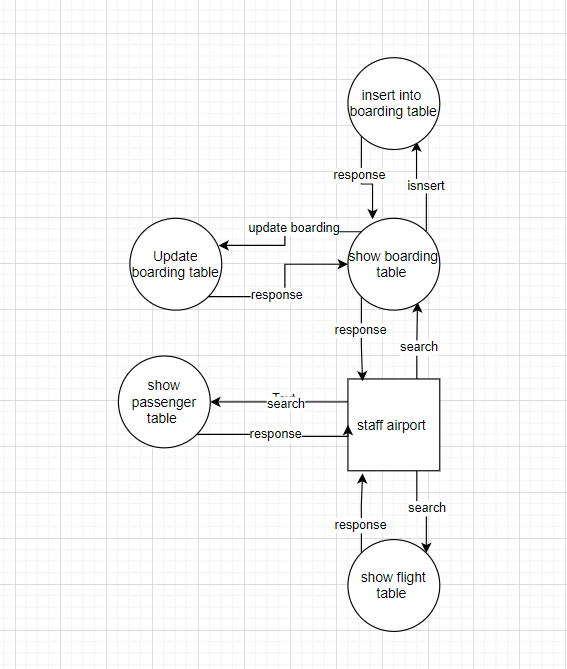
1. The passenger\_user1 diagram



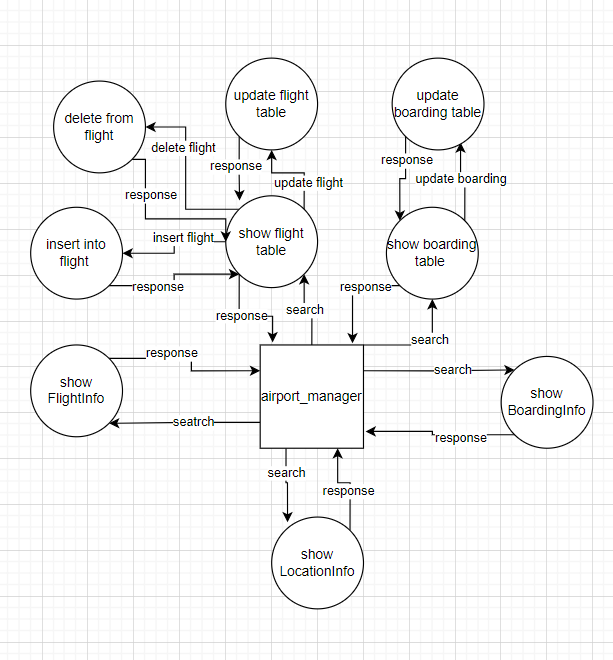
1. The flight manager diagram



1. The staff\_airport diagram

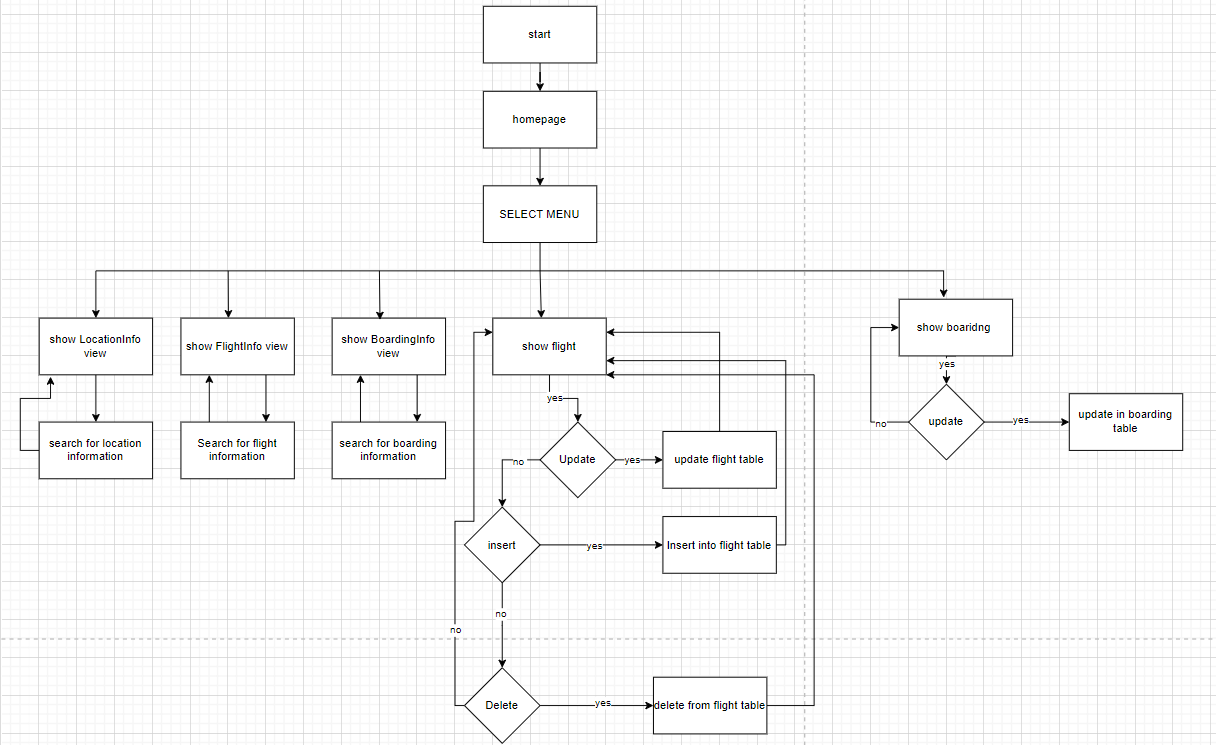


1. Airport\_manager diagram



Note: I discovered that we put “search” in the diagram just when we deal with view query.

The flow chart for the system



This flowchart explains the logic off all the system and the interface of the airport\_manager. It shows the structure of the system and the functionality of the system and al the steps that the manager could follow using his interface.

### Interfaces Development

|  |  |  |  |
| --- | --- | --- | --- |
| **Page ID** | **Title** | **Description** | **Screenshot** |
|  | boarding | This page show me the boarding entity that ii made in my database so we can control all the data by using this page and this the interface that the airport manager is going to view while he interact with the data in the database |  |
|  | flight | This page show me the flight entity that ii made in my database so we can control all the data by using this page and this the interface that the airport manager is going to view while he interact with the data in the database |  |
|  | boardinInfo | This is the interface of view query that is used to reach data from more than one table in one screen and the airport manager could use the view because he have the permission to select and view these data together |  |
|  | FlightInfo | This is the interface of view query that is used to reach data from more than one table in one screen and the airport manager could use the view because he have the permission to select and view these data together. It including many data |  |
|  | locationinfo | This is the interface of view query that is used to reach data from more than one table in one screen and the airport manager could use the view because he have the permission to select and view these data together. It including many data |  |

# Maintenance

## Database recovery & backups

I used the notepad in the first place to recovery by putting all the query that I used during building the database in the notepad and order them with step so I can reach the data easily. I also tried to import the file from time to time so that I don’t lose my database that I build. Also, I used to save the project in the filles, and I documented each step in the word file and the notepad so I can reach any information that I need easily to recover the data and back it up. I also saved the database file in more than one device so I could reach it easily if any error happened to my laptop that I’m building the project on. Also, I took picture for a lot of important information so if anything, happen I could recover the data manually.

## Database maintenance in general

The database maintenance is essential step while working on my project. I faced a lot of bugs and errors during building my database. I tried to solve it using several ways such as. After I finished creating the tables I worked on fixing the Buggs that I faced such as wrong data type and so on. Also, after choosing the foreign keys I take a lot of time in maintenance the database to make it without any mistakes and that moved me to the next step which is inserting the data inside the table. I also faced some issues during the process of making the documents so sometimes I discovered that I’ve entered wrong data, so I make the project aging form the zero. I depended on the slied to fix the bugs and I did some research about fixing some bugs when there is no information’s in the slide. The query that ql provide is so useful in maintenance so if you have any error, most of the times there is a query that makes everything right.

# Testing

## Data Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Type** | **Description** | **screenshot** |
|  | All cases of PK | The primary key in passenger table is passport\_number so I tried to insert two rows with the same primary key and because the primary should be unique it gave an error which is expected |  |
| The primary key can not be null so it is impossible to give null value, so the SQL gave an error which is expected |  |
|  | All cases of FK | Insert invalid foreign key because the foreign key is a primary key in other table so we cannot but a new value in the table if the primary key which is the boarding |  |
| updating the foreign key value with an invalid foreign key not found in the primary key table itself |  |
| I will test on update cascade, I will change the value of the foreign key in the primary key, and check it in a table that has it as a foreign key, I will the flight id and check it in the boardingtable |  |
| I will test the delete on cascade, by deleting a primary key and checking if it’s deleted in the foreign key table, I also will do the flight and boarding table |  |
|  | Default | I inserted into the flight table, without putting the type, it's default is ‘D’ |  |
|  | Not null | I will be inserting a NULL value into a not null entity, which is gate number in the airline location table |  |

## Output Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Query Description** | **Screenshot (query + result)** | **Result validation** |
|  | I will insert into the passenger table a new record |  | After insert: |
|  | Deleting from the passenger table a record |  | After delete: |
|  | I will update the passenger passport number |  | After update: |
|  | I will only print Male passengers |  |  |

## Security Validation

**Note**: you need to test the given and not given privileges.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Username** | **Description of privilege/no privilege** | **Screenshot (query + result)** |
|  | passenger | We tried to delete from the boarding entity, but the passenger doesn’t have the permission to delete |  |
|  | Airport\_manager | We tried to insert data into boarding entity, but the airport manager user doesn’t have the permission to insert data in that table |  |
|  | Flight\_manager | We tried to insert data into plane entity, but the flight manager user doesn’t have the permission to insert data in that table |  |
|  | Staff | The staff doesn’t have the permission to delete from the plane so the sql gave an error which was expected |  |

### 

## GUI Validation

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **screenshot** |
|  | I tried to delete a row of table from the boarding table where the airport manager doesn’t have the permission to delete from this table so the sql maestro show an error. |  |
|  | I tried to insert a row inside the boarding table but because this user who the airport\_manager doesn’t have the permission to insert so the sql maestro show this error |  |
|  | Here I tried to delete a row from the flight table using the manager of the airport interface and the row has been deleted from the sql maestro and the sql | AFTER delete |
|  | In this table I tried to update a value in the flight table and the admin has the permission so I epdated the flight\_id to F100002 TO F100012 AND THE UPDATE TAKE PLACE IN THE SQL AND SQL maestro | After update |

## Assess whether meaningful data has been extracted.

**Creating the tables**

1)

CREATE TABLE passenger (

passport\_number char(9) PRIMARY KEY,

Fname VARCHAR(15) NOT NULL,

Lname VARCHAR(15) NOT NULL ,

gender CHAR(1) DEFAULT 'm' CHECK (gender IN ('f', 'm')),

nationality VARCHAR(25) DEFAULT 'jordan',

passenger\_type CHAR(1) DEFAULT 'D' CHECK (passenger\_type IN ('D', 'A' )),

DOB DATE

) ;

2) CREATE TABLE flight (

flight\_id CHAR(9) PRIMARY KEY,

departure\_city VARCHAR(20) NOT NULL DEFAULT 'Jordan',

arrival\_city VARCHAR(20) NOT NULL,

departure\_date DATETIME NOT NULL,

arrival\_date DATETIME NOT NULL,

numberofpassenger INT,

type CHAR(1) DEFAULT 'D' CHECK (type IN ('D', 'A')),

IATA\_code CHAR(9) NOT NULL

);

3) CREATE TABLE boarding (

boarding\_id CHAR(9) PRIMARY KEY,

flight\_time DATETIME NOT NULL,

passport\_number CHAR(9) NOT NULL,

flight\_id char(9) NOT NULL,

boarding\_staff VARCHAR(15),

baggage\_weight DECIMAL(10, 2),

cost DECIMAL(10, 2),

IATA\_code CHAR(9) NOT NULL

) ;

4) CREATE TABLE passenger\_phone (

passport\_number CHAR(9),

phone\_number VARCHAR(15) NOT NULL,

PRIMARY KEY (passport\_number, phone\_number)

);

5) CREATE TABLE plane (

plane\_id CHAR(9) PRIMARY KEY,

company VARCHAR(15),

model VARCHAR(15) NOT NULL,

capacity INT,

classes CHAR(1) DEFAULT 'A' CHECK (classes IN ('A', 'B')),

gate\_number INT,

type VARCHAR(30),

IATA\_code CHAR(9) NOT NULL

);

6) CREATE TABLE airline (

IATA\_code CHAR(9) PRIMARY KEY,

flight\_id INT NOT NULL,

Name VARCHAR(15) NOT NULL

);

7) CREATE TABLE airline\_location (

IATA\_code CHAR(9),

gate\_number INT NOT NULL,

department VARCHAR(15),

PRIMARY KEY (IATA\_code, gate\_number, department)

);

8) CREATE TABLE airline\_email(

IATA\_code CHAR(9),

email VARCHAR(25) UNIQUE,

PRIMARY KEY (IATA\_code, email)

);

This process is considered the basic of the database which include the creating of the database and the data inside the database. This is important because it include the database and the tables and the datatype of each attribute. So, this process forms the foundation for efficient data organization, retrieval. So, we ensure data integrity, reduce redundancy, and improve query performance by creating the tables and the datatype.

**Adding foreign keys**

1)

ALTER TABLE boarding

ADD CONSTRAINT fk\_boarding\_IATA\_code

FOREIGN KEY (IATA\_code)

REFERENCES airline(IATA\_code)

ON UPDATE CASCADE

ON DELETE CASCADE;

2)

ALTER TABLE boarding

ADD CONSTRAINT fk\_boarding\_passport\_number

FOREIGN KEY (passport\_number)

REFERENCES passenger(passport\_number)

ON UPDATE CASCADE

ON DELETE CASCADE;

3)

ALTER TABLE passenger\_phone

ADD CONSTRAINT fk\_phone\_passport\_number

FOREIGN KEY (passport\_number)

REFERENCES passenger(passport\_number)

ON UPDATE CASCADE

ON DELETE CASCADE;

4)

ALTER TABLE plane

ADD CONSTRAINT fk\_plane\_IATA\_code

FOREIGN KEY (IATA\_code)

REFERENCES airline(IATA\_code)

ON UPDATE CASCADE

ON DELETE CASCADE;

5)

ALTER TABLE airline\_location

ADD CONSTRAINT fk\_location\_IATA\_code

FOREIGN KEY (IATA\_code)

REFERENCES airline(IATA\_code)

ON UPDATE CASCADE

ON DELETE CASCADE;

6)

ALTER TABLE airline\_email

ADD CONSTRAINT fk\_email\_IATA\_code

FOREIGN KEY (IATA\_code)

REFERENCES airline(IATA\_code)

ON UPDATE CASCADE

ON DELETE CASCADE;

7)

ALTER TABLE boarding

ADD CONSTRAINT fk\_boarding\_flight

FOREIGN KEY (flight\_id)

REFERENCES flight (flight\_id)

ON UPDATE CASCADE

ON DELETE CASCADE;

8)

ALTER TABLE flight

DROP FOREIGN KEY fk\_flight\_IATA\_code;

ALTER TABLE flight

ADD CONSTRAINT fk\_flight\_IATA\_code

FOREIGN KEY (IATA\_code)

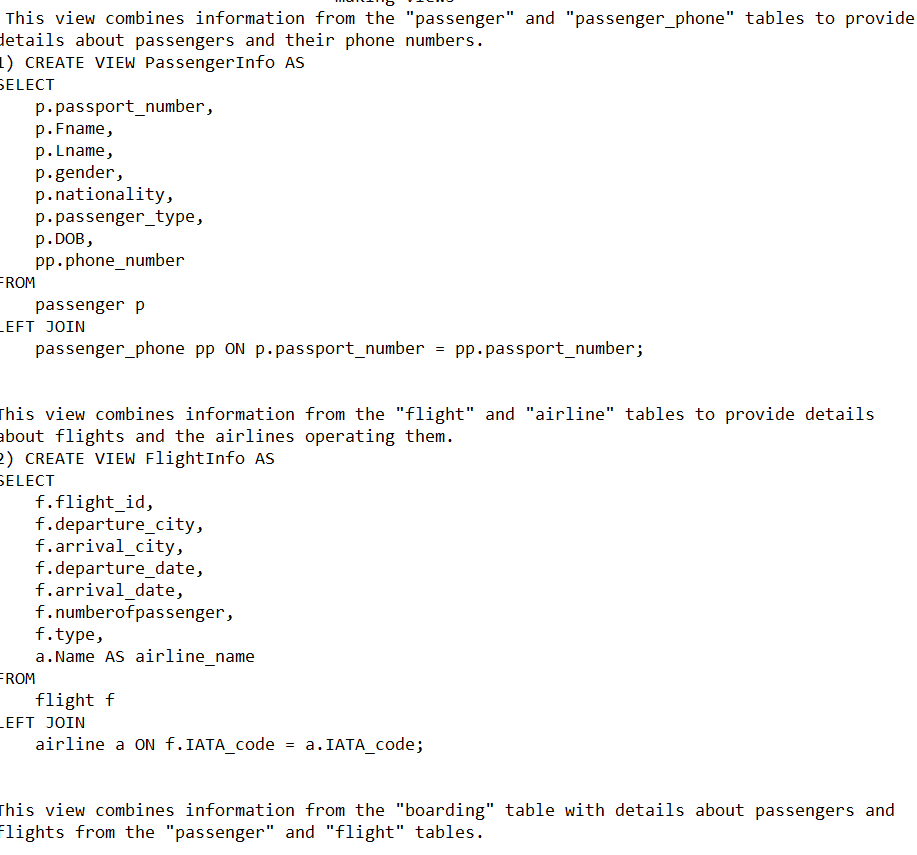
REFERENCES airline(IATA\_code)

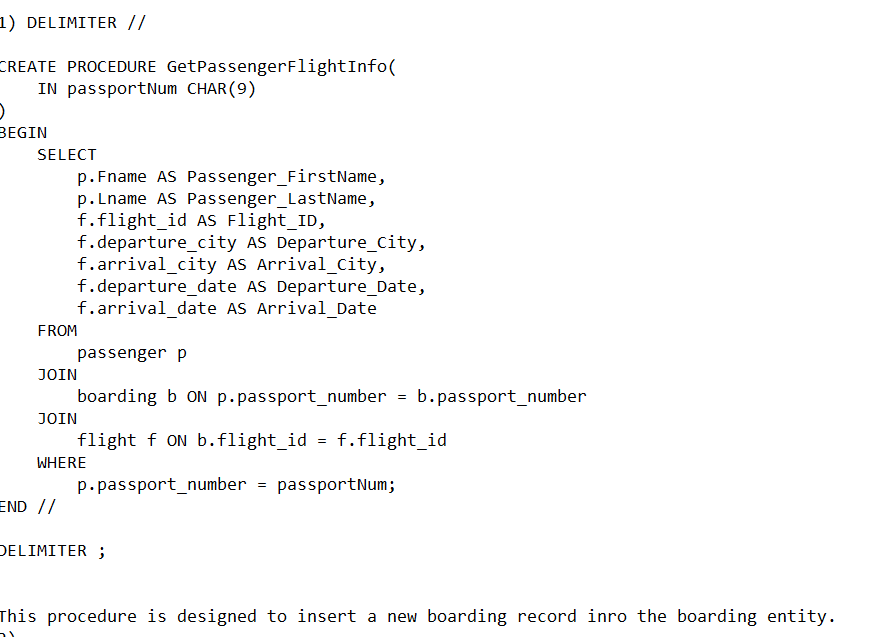
ON UPDATE CASCADE

ON DELETE CASCADE;

This data is meaningful because it establishes relationships between tables, ensuring that data consistency is maintained across related tables. These foreign keys prevent actions like inserting or updating records that would break these relationships. Also, foreign keys help improve data quality by ensuring that only valid, existing values are used in related tables.

Views and procedures





The views are very useful and meaningful data because it is allowed to abstract complex underlying data structures into simplified virtual tables =, Users can query a view without needing to understand the underlying table structure. Also, the data security because the view enables to implement security mechanisms by restricting access to sensitive data.

The procedures are very important also because they helped me to improve performance and manage transactions and ensuring the data consistency and integrity.

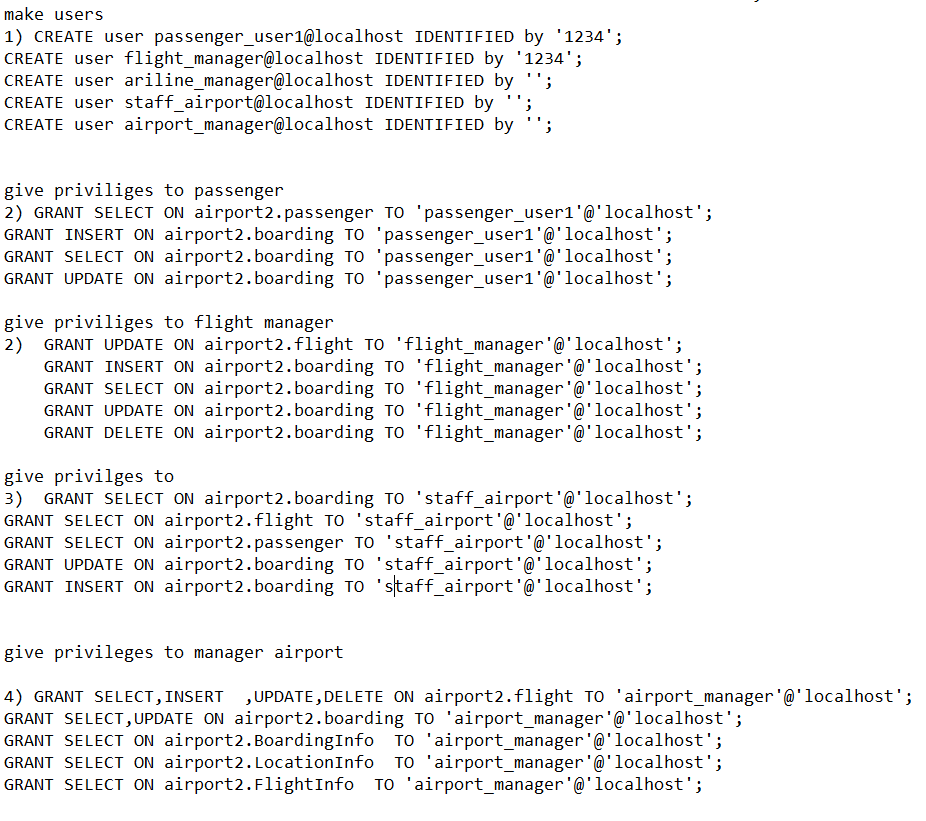
**Insertion data**

A computer screen shot of a computer code

Description automatically generated

This is so important process and this information that inserted into the tables was meaningful because they are the data that the tables have been built to and these details. These data fill up the tables and the users will use the interfaces to manipulate this data.

Privileges and users



These data include the users account that will use the system with the privileges that they have so these data is meaningful and so helpful and built depending on the users requirements and it manages the usage of the database so it ensure that everyone just could reach the information that he supposed to reach.

All of these details is already explained in this report and that is why these data are important and meaningful.

## Assess the effectiveness of testing.

The testing is the most important process before trying the new system and move it from a project to actual database system. First i started with testing the physical design so after creating all the tables i showed the diagram that shows all the information about my database such as the tables, attributes, and the data type of the data. Then i made view and procedure and tested them the commands that gives me the ability to know if the views and the procedures’ function or not. After that, i started to test the tables.

Then i made users and gave them privileges and I tested all the privileges using some commands to make sure that the privileges work as expected or not so i started to test them by showing the commands and make sure that the commands inserted are as expected.

I did this operation for the flight\_manager, airport\_manager, passnger\_user1 and the staff\_airport and all the information. After making the sql maestro i tested the functionality of the user off all the tables which are flight, boarding, LocationInfo, BoardingInfo, FlightInfo and tested the privileges that I gave by trying to view, update, delete and insert in and from the tables.

Then in the data validation, I tested all the cases that could be tested in the primary key so i tries to put a null value in the passenger table in the primary key field and then i tried to insert a duplicate value for the primary key and I tried this operation in all the tables. I faced some issues while testing the tables, but I fixed these bugs.

After that I tested the foreign key by insert invalid foreign key because the foreign key is a primary key in another table. Then I tried to update the foreign key with invalid value in the essential table of the foreign key. After that I tested the on-update cascade query by changing the value of the foreign key in the primary key and check in the tables that has it as a foreign key. Then i tested the on delete cascade by deleting a primary key and check if it is deleted in the foreign key. Some of these operations didn’t function from the first time so the testing was very helpful in changing the data to a better sorted data.

Then I tested the default by inserting the flight table without putting the type and it is default.

Then I tested the not null by inserting a null value into a not null entity which is gate number in the airline table.

Then I tested the output validation by testing the insert, delete, update, and simple query to test the functionality of the database and make sure that it gives the best result expected.

Then I tested the privileges again by trying all the privileges in each user.

Finally, I tested the GUI by trying the privileges of the user in all the tables such as insert, delete, select, update and so on.

I then tested that if any data changed in the GUI the data will change in the sql and the opposite to make sure that the system work successfully or not and to test the relation and links.

All the results of the testing are noted in my technical documentation in details.

# Evaluation of database solution

## Effectiveness of the database solution based on user and system requirement.

This project is consider the actual buit of the database after designing the database in the previous document. In this project I solved the database and made all the changes to make alignment with the user expectations. I removes the relationships that is unlogical such as the relation ip between the plane and the passenger and I made a better version of the database. Any unlogical information has been distracted from the database in the actual implementation. I made the users and system requirements as it was expected from the previous implemnetaion such as the passenger, staff, flight manager and the airport manager. I al;so gave all the priveliges on depening on the requirements for example.

1. Passenger:

* Access the system using a unique identifier (passport\_number, name) to make a boarding.
* The passenger could use the system to view the boarding details.
* The passenger could use the system to see the roadmap of the airport procedures.
* The passenger could use the system to update the boarding details such as baggage.

1. Flight management department:

* Have the access to edit and update the flight information such as the date and arrival city.
* Assign passengers to flight.
* View the boarding details for different flights.
* Manage the baggage in the plane.

1. Airline management:

* Manage airline information, including location, IATA code, email, and associated planes.
* Add new flights and link them with corresponding airline.
* Monitor boarding operation and passenger details.
* Gives the ability to match the passengers to their seats and to the plane to be in its right gate to make it easy to passengers to reach their plane.
* View all the boarding details and the baggage component by the system.

1. Airport staff:

* View a summary of passenger boarding and flight schedule.
* Assist passenger with any issues related to boarding and flight.
* Update boarding details based on passenger requests.
* Make seat allocation and ensure that the passengers have boarding in this plane.
* View the passenger information such as passport details and any information about the passengers.

1. The airport management:

* Match the planes with gates and places to stay on.
* Manage the passengers with their flight.
* Check all the airline’s location and the procedure from the arrival of the passenger to the airport until he leaves,
* View the departments of the airport in order to stop any bad behaviour inside the airport.

All the privileges Was implemented depending on these requirements. As I told in this document.

Some of the requirement was not achieved because it was analogical in the actual building of the database and the actual implementation of the project.

## Suggested improvements

1. Further normalization of the present architecture is necessary to reduce duplication and enhance data integrity. Consider creating separate columns for features like country and gender and building foreign key links rather than keeping them directly in the "passenger" table.
2. Additional Entities: Consider introducing new entities to represent concepts not currently addressed. For instance, I might want to include entities for airport staff, airport facilities, and passenger bookings to manage reservations and ticketing more comprehensively.
3. I also should try to manage the privileges of the users more accurately so I should give more privilege on the flight table to the flight manager, also I should give more privileges to the passengers to manage his own information in the tables.
4. I also should try to build the data with less completion because I have 8 entities that include sometimes irrelevant data and the views in my table reduce this confliction a little bit and I need to make more views and procedure to manage the data more efficiently and manage the tables such as flight, boarding, passenger more accurately.
5. I should use a better method of data recovery because my backway and data recovery is so simple because I used notepad and I imported the database in more than one device but I should make it more efficient.
6. I should make user interface for all the users which are passenger, staff, and flight manager because I just made the interface for the airport manager, and I should build a better interface.
7. Documentation: My document is so complicated such as the technical and user documentation, so I need to make more detailed documents about the queen alia airport database that include more specific details.

## Evaluation based on improvements needed.

The recommended changes, such as proper entity relationships and data normalization, can improve data integrity. This ensures that the information stored in the database is accurate and consistent, reducing errors and improving the overall quality of services provided to passengers. The system can reply to user requests more swiftly by streamlining the data storage and retrieval operations. Passengers will benefit from quicker check-in, boarding, and access to flight information, which will improve their airport experience. The suggested user interfaces can make it simpler for passengers, airlines, and airport workers to engage, which will make it simpler to manage data and procedures. Increased user pleasure and efficiency may result from this. By giving each user access solely to the data necessary for their position, each stakeholder is given the proper permissions. This makes the system more user-friendly and improves security and data privacy. The enhancements to database design allow for expansion of airlines, flights, and passenger counts. The technology can manage larger data loads without noticeably degrading performance as the airport grows. A well-structured database is easier to maintain and update. This reduces the time and effort required for database maintenance and minimizes the risk of data corruption or loss. With a more organized and comprehensive database, it becomes easier to analyze trends, passenger behaviors, and flight performance. This information can be used to make data-driven decisions, optimize airport operations, and improve services. Moreover, The updated database is capable of advanced reporting features, allowing stakeholders to create bespoke reports and learn more about numerous airport operations-related topics. Implementing these improvements may require additional development time, especially if there are significant changes to the existing system. Balancing the need for quick implementation with the desire for a well-structured database is essential. Depending on the complexity of the changes, additional resources such as skilled database administrators, developers, and testers may be necessary. Adequate resource allocation is crucial to ensure a successful implementation. It can be difficult to migrate data to the new structure if there is an existing database. In order to achieve a smooth transfer without data loss or errors, careful preparation and testing are required. Lastly, Users and staff will need training to adapt to the new interfaces and workflows. Adequate training and support should be provided to minimize disruption during the transition.

References

1. [*https://www.oracle.com/database/technologies/security/db-authn-authz.htm*](https://www.oracle.com/database/technologies/security/db-authn-authz.htm).
2. [*https://www.geeksforgeeks.org/interfaces-in-dbms/*](https://www.geeksforgeeks.org/interfaces-in-dbms/).
3. [*https://budibase.com/blog/tutorials/database-gui/*](https://budibase.com/blog/tutorials/database-gui/).
4. [*https://www.yurishwedoff.com/database-testing-for-ui-various-tools-to-check-data-integrity-and-accuracy/*](https://www.yurishwedoff.com/database-testing-for-ui-various-tools-to-check-data-integrity-and-accuracy/).
5. [*https://www.w3schools.com/sql/sql\_datatypes.asp*](https://www.w3schools.com/sql/sql_datatypes.asp).