mohammad. salameh

Database Programming

Final Assignment

Contents

[Part 1 3](#_Toc157381251)

[TASK1 3](#_Toc157381252)

[introduction 3](#_Toc157381253)

[Data Requirements 3](#_Toc157381254)

[User Requirements: 5](#_Toc157381255)

[System Requirements: 6](#_Toc157381256)

[Conceptual model 7](#_Toc157381257)

[Logical model 8](#_Toc157381258)

[Physical MODEL 11](#_Toc157381259)

[TASK 2 : dbms function of the sub-components 13](#_Toc157381260)

[Task3 18](#_Toc157381261)

[dfd 18](#_Toc157381262)

[flowchart 19](#_Toc157381263)

[Task 4 Effectiveness of the database and DBMS 19](#_Toc157381264)

[PART 2 22](#_Toc157381265)

[TASK 1 22](#_Toc157381266)

[An explanation of buffer managers and storage systems' dynamic data structures 34](#_Toc157381267)

[Indexing 35](#_Toc157381268)

[Criticize the process of mapping SQL queries to the developed CRUD statements and suggest improvements to optimize the developed CRUD quires. 36](#_Toc157381269)

[Improvements 37](#_Toc157381270)

[Part 3 38](#_Toc157381271)

[Task 2 39](#_Toc157381272)

[Part 4 41](#_Toc157381273)

[Task 1 41](#_Toc157381274)

[Task 2 41](#_Toc157381275)

[The test plan. 41](#_Toc157381276)

[Flights 44](#_Toc157381277)

[Employee 48](#_Toc157381278)

[Tickets 51](#_Toc157381279)

[Airport 54](#_Toc157381280)

[File loading testing 57](#_Toc157381281)

[Evaluate the improvements 59](#_Toc157381282)

[Reference 60](#_Toc157381283)

# Part 1

## TASK1

### introduction

The current airport system is facing a lot of problems ,like for example in handling passengers or departments , so this may effect it efficiency and also effect the user experience , so we need a robust system that will streamline the airport database system ,and to overcome these challenges and improve the overall airport operations , so our aim and goal is to design The current Airport System is facing various challenges that hinder its efficiency and user a comprehensive Airport Database System that follow up and address issues in the present system , and make sure to meets the specified user and system requirements, additionally the system will be designed to manage various aspects of the airport (Passenger management , Ticket management,…), The system aims to provide a user-friendly interface for efficient management and retrieval of essential data related to airport activities.

### Data Requirements

In addition to these primary components, the system must be user-friendly so that airport personnel can readily access and handle this information. It should also include provisions for efficiently updating and retrieving information to accommodate the dynamic nature of airport operations, by taking all of these factors into account, the Airport Database System will be better able to improve airport operations and the overall experience for both passengers and personnel.

* **Passenger Information** 🡺here in this section it will cover all the information about the users of those who will use the airport system, so it will include their names and contact info, in addition to any other needs they might have. Consider it a thorough directory of everyone who passes through the airport.
* **Flight Details** 🡺same as the passenger but here is for the flights details that will include information for flights, such as their destination and scheduled departure and arrival times airlines and more. This is like a complete timetable that keeps track of all flights.
* **Ticketing Information** 🡺 this will be related directly to the passengers, so it will present their ticket info like ticket numbers, seat allocations.
* **Employee Records**🡺This will include all the information on that works at the airport , for example plane crew to the captain , so it will record their name , positions work schedules , contact information. It's like a human resources system designed exclusively for airport staff.
* **Interactions of these components and interface** 🡺the airport system should interface should be easy to use and attractive for the user , and also to understand how these components interact with one another. For example, passenger information is linked to ticketing (who purchased which ticket) .

**Flight Information:**

* **Flight number**
* **Departure and arrival airports**
* **Departure and arrival times**
* **Aircraft type/model**
* **Status (on time, delayed, canceled)**
* **Gate information**

**Passenger Information:**

* **Passenger First name**
* **Passenger last name**
* **Ticket number**
* **Seat number**
* **phone number.**
* **Passport/ID details**

**Airline Information:**

* **Airline name**
* **Airline code**
* **Contact information**
* **Fleet details**

**Airport Information:**

* **Airport code**
* **Location details**
* **Facilities available**
* **Gate assignments**
* **Security information**

**Crew Information:**

* **Crew member names**
* **Role (pilot, co-pilot, cabin crew)**
* **Contact information**
* **Schedule**

### User Requirements:

**Passengers**:

* Display all the passengers and their information (First Name, last Name, ….).
* The addition of new passengers with their own info’s such as name, passport number, and unique ID.
* The ability to delete a passenger based on their ID.
* The ability to update a passenger based on their ID, info without deleting him the passport number for a specific passenger.
* Display a list of all the passengers.

**Flights**:

* The addition of new flights and add it info such as flight number, destination, and unique ID.
* Delete a flight based on its ID.
* Display a list of all flights.

**Employees**:

* Add a new employee with details such as name, employee ID, and unique ID.
* Delete an employee based on their ID.
* Display a list of all employees.

**Tickets**:

* Add a new ticket with details such as ticket number, passenger ID, and flight ID.
* Delete a ticket based on its number.
* Display a list of all tickets.

**Airports**:

* Add a new airport with details such as name, location, and unique ID.
* Delete an airport based on its ID.
* Display a list of all airports.

**System Operations:**

* Provide a main menu for navigating through different modules.
* Allow users to exit the system.

### System Requirements:

**Structures**:

* Passenger structure with attributes: name, passport number, ID.
* Flight structure with attributes: flight number, destination, ID.
* Employee structure with attributes: name, employee ID, ID.
* Ticket structure with attributes: ticket number, passenger ID, flight ID.
* Airport structure with attributes: name, location, ID.

**Functions**:

* Functions to add, delete, update, and display information for passengers, flights, employees, tickets, and airports.
* Functions to find entities by their respective IDs.
* Main menu function for overall system navigation.

**User Interface:**

* Clear and intuitive menu structures for passengers, flights, employees, tickets, and airports.
* Input validation for user entries to ensure data integrity.
* Exiting functionality to terminate the system.

**Data Storage:**

* Linked lists for storing passenger, flight, employee, ticket, and airport data.
* Memory allocation and deallocation for dynamic data storage.

**Error Handling:**

* Proper error messages for cases where the user enters invalid data or when an entity is not found.

### Conceptual model

A database's conceptual design performs an abstract model that defines what data will be stored and how the data elements , so we want to define it 🡺 it is a high-level and abstract model that does not include any explicit technical specifications, but it focuses on the entities and there identifications , and the flow of the data , inside the system , the main goal for it is to organize data in such a way that it matches the organization's business logic.

We have 6 entities and they are making an airport system , so we have airport passenger ticket department and employee , so for example the ticket have a relation with passenger because passenger must tickets and ticket number so they can enter the flight , and also the ticket a related to a flight so the flight will have a specific ticket number so we can differentiate between other flights, same foe the other entities .

A screenshot of a computer screen

Description automatically generated

### Logical model

* Passenger Table

Attributes: PassengerID, FirstName, LastName, PassportNumber, ContactNumber

Primary Key (PK): PassengerID

Description: This table stores information about the passengers.

* Flight Table

Attributes: FlightID, FlightNumber, Destination, DepartureTime, ArrivalTime, Gate, AirportCode

Primary Key (PK): FlightID

Foreign Key (FK): AirportCode (references Airport table)

Description: Contains details of all the flights.

* Employee Table

Attributes: EmployeeID, FirstName, LastName, ContactNumber

Primary Key (PK): EmployeeID

Foreign Key (FK): Airport id (references airport table)

Description: Records the details of airline or airport employees.

* Ticket Table

Attributes: TicketNumber, PassengerID, FlightID, SeatNumber

Primary Key (PK): TicketNumber

Foreign Keys (FKs): PassengerID (references Passenger table), FlightID (references Flight table)

Description: Stores the details of the tickets purchased by passengers.

* Airport Table

Attributes: AirportCode, AirportName, Location

Primary Key (PK): AirportCode

Description: Holds information about the airports.

A screenshot of a computer

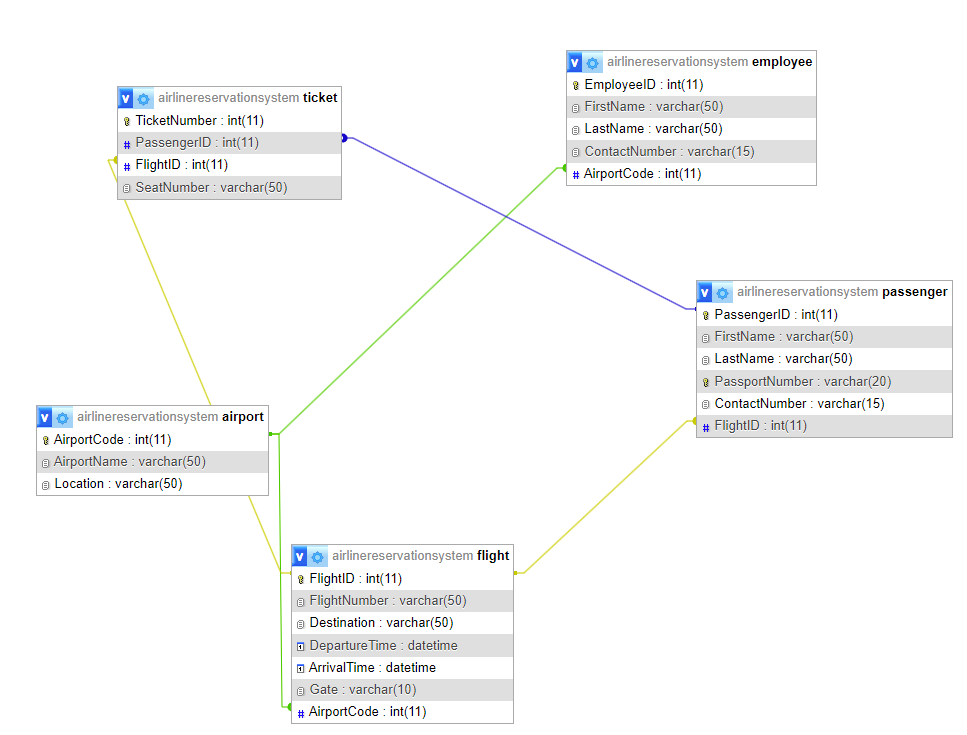
Description automatically generated

### Physical MODEL

The blueprint for how data is kept and handled in a database system is called the physical design of the database. It outlines the links between tables as well as the specifics of each table's structure, including columns, data types, keys, and constraints. The objective is to design a schema that supports the specified queries and transactions, preserves data integrity, maximizes performance, and captures all relevant data.

The purpose of each of these tables is to hold data that corresponds to the relationships and entities specified in the conceptual model. The foreign keys protect the integrity of relationships between entities, while the primary keys guarantee uniqueness within each database. The business operations of the airline reservation system are supported by this structure, which makes it possible to get and manipulate data efficiently.

| **Table** | **Name** | **PK** | **Attributes** | **Description** |
| --- | --- | --- | --- | --- |
| 1. | Passenger | PassengerID | PassengerID (INT), FirstName (VARCHAR(50)), LastName (VARCHAR(50)), PassportNumber (VARCHAR(20), Unique), ContactNumber (VARCHAR(15)) | Stores information about individuals who book flights. **PassengerID** serves as the primary key for unique identification. |
| 2. | Flight | FlightID | FlightID (INT), FlightNumber (VARCHAR(50)), Destination (VARCHAR(50)), DepartureTime (DATETIME), ArrivalTime (DATETIME), Gate (VARCHAR(10)), AirportCode (INT, FK) | Contains details of flight schedules. **FlightID** is the primary key. **AirportCode** is a foreign key referencing the **Airport** table. |
| 3. | Ticket | TicketNumber | TicketNumber (INT), PassengerID (INT, FK), FlightID (INT, FK), SeatNumber (VARCHAR(50)) | Records ticket purchases by passengers. **TicketNumber** is the primary key. It has foreign keys **PassengerID** and **FlightID** that reference the **Passenger** and **Flight** tables, respectively. |
| 4. | Airport | AirportCode | AirportCode (INT), AirportName (VARCHAR(50)), Location (VARCHAR(50)) | Holds information about airports. **AirportCode** acts as the primary key. This table might be referenced by other tables like **Flight** and **Employee** through foreign key relationships. |
| 5. | Employee | EmployeeID | EmployeeID (INT), FirstName (VARCHAR(50)), LastName (VARCHAR(50)), ContactNumber (VARCHAR(15)), AirportCode (INT, FK) | Maintains records of employees working for the airline or at the airport. **EmployeeID** is the primary key. **AirportCode** is a foreign key linking employees to their respective airports. |



## TASK 2 : dbms function of the sub-components

When we want to design a database for an airport management system , it will be built on a several of components , so we can have a fully functional DBMS which different function and each function has it won purpose , so by the use of C programming language and Hash Map data structure and also the sue of other programming and non-programming components those can lead to a good result .

1. **Data Storage** 🡺Data storage is very important DBMS components because in our management system we are required to same various things , like for example entities and it attributes (passenger , flights , .. ), so the data storage component manages how and where the data will be stored and in which forum , so as we said before we have The C programming language and HashMap each one has its own use in the data storage , how ? 🡺 The C is responsible for creating these entities and the hash map is for the storing, it stores the data as a key and value which make it easy to manage, the key will an Identifier (Primary keys), value (other attributes). So briefly the data storage is used for managing how and where the data will be stored , so it is very important component for retrieving data and store , now for the technique that the data storage uses is data serialization and deserialization, memory management, that will use arrays and structs for organizing the data in the memory and b-tress for hierarchical storage and the for storing in pairs we will use the key-value pairing .

I am using this component in my database system , because I need a place where I can store data in it , so I will be able to backup and retrieve data from it , so I can store data for different , and I will use different linked lists to store the data for passengers, flights, staff, tickets, and airports.

1. **Query Compiler**  🡺 in this component if we want to extract or edit some attributes or add data from or into the DBMS by queries it done here, so this method include accepting SQL queries , and it also analyze these queries and then gaining the access to the relevant data from the database , and the same process if we want to add or update things you will grant the access after analyzing the query. When we want to accomplish these queries, we can refer to the C, as in the C there is some techniques that perform the same process this component need like for example parsing in the compiler, and the query execution optimization. so at the end we use the SQL queries for processing and optimizing these queries so that will enable manipulation and retrieval, so for the techniques that the query compiler uses is SQL parsing and Syntax tree so it can easily understand the grammar that the query have, and also for the data structure is will use trees, and graph for query planning and hash tables.
2. **Indexing and Retrieval: 🡺** For speedy data retrieval, indexing is essential. For key-based retrievals, hash maps can provide quick access times. To minimize collisions and guarantee effective retrieval, hash functions must be thoughtfully constructed to disperse keys uniformly over the map. Hash map collisions can be handled using strategies like chaining or open addressing. So, we use the indexing process for faster data access especially when we have a large dataset we need to faster indexing and data accessing, so the indexing using some techniques that make for efficient so it uses the hash functions , and collision resolution , and for the data structure is uses the hash table to have a quick key-based access .

I am also using this for this in my system so I can easily and faster indexing after saving the data or entering the data to the files.

1. **Concurrency Control and Transaction Management**🡺 Concurrency with database access is crucial in a multi-user setting such as an airport. An essential part of database management is ensuring the atomicity, consistency, isolation, and durability (ACID) of transactions through techniques like locking mechanisms (e.g., mutex, semaphore) to regulate access to shared data. Careful synchronization of resource access is required when implementing them in C , it is used for managing the simultaneous database access , so it ensures that the transactions are been processed reliably and efficiently , so this will been done by the use of the locking mechanisms techniques , and also the use of the dead lock and transaction isolation levels, for the datastore Techniques: Implementing locking mechanisms, deadlock detection and resolution, transaction isolation levels, for the data structure that will be used is the logs and the lock tables are used for managing the resources access and transaction sequencing in queues.

The use of the thread is to representing the attempt to manage the concurrency, and it is very important to be use In the database so we can ensuring that database activities do not conflict with one another when many threads are active, which is an important part of concurrency control and transaction management.

1. **Buffer Management** 🡺 this component is responsible for memory and data management in memory allocation and the techniques that is used caching and buffering, so it is very important to apply an effective buffering method to control the I/O operations and enhance the overall performance , so this will handled by the programming language ,so in C we have techniques that we perform for memory allocation and buffer management and replacement , so the buffers are used for optimizing the storage and retrieval of the data that is stored in the data structure , so this component is very important for performance enhancement, for the techniques that is used in the buffers is the buffer pool management and buffer replacement algorithms such as (FIFO) and in addition of the caching strategies, for the data structure we use the linked lists and hash maps for the caching strategies.

In my database system I will deal with files and for saving and reading data, so we going to deal with (fopen, fprintf, fscanf, and fclose), These procedures are part of buffer management, which involves temporarily storing data before writing to or reading from a file.

1. **Backup and Recovery 🡺** this components is one of the most significant components, we use this component to prevent data loss, so any management systems need and require a strong data backup and recovery, so it is very important to have a backup strategy that backup data to another place, and putting rollback and recovery methods in place, so at the end we need the back and recovery components for data durability and consistency , and its very important for data integrity by the time , so it will need techniques to that so for the techniques that it uses is Backup strategies, transaction log management, data recovery methods, in addition for the transaction logs and back up files data structures .

We will have functions that save data to files , so we can say we are making backup for data , might be viewed as a basic type of backup.

1. **Transaction manager 🡺** The transaction manager in every DBMS, and we need because it is crucial for maintaining the consistency and integrity of the database. And also it monitor and supervises the database operations and ensures that the (Atomicity, Consistency, Isolation, Durability) guidelines are followed correctly , the Atomicity 🡺 we use it for guarantees that the transaction and its operation will be completed successfully or none at all , and for the Consistency 🡺 it is used for ensuring the movement of the transactions that moves the database from a valid state to another , Durability 🡺 her in case of any system failure , the durability will make sure that the system remain committed . So, for the transaction manager we can create it by using C programming language, we can use commit, that will ensure that the changes are made after a successful transaction, so for the techniques that the transaction manager uses is the transactions coordination ,so that will ensure atomicity, consistency, isolation, and durability , and for the data structure is uses the logs for transaction state , tables for activity tracking .
2. **Logging and lock table 🡺** in the DBMS logging is very important feature for the because it helps in monitoring and recording the flow of operations and events , so it require some protocols since it is responsible for restore the database in case of any failure .so each log contains a detailed a transaction activity , and it should be implemented in every DBMS so we will be able to store and retrieve data easily , now for the lock table 🡺 the purpose for it is for concurrency control , so as a result it will help in preventing any conflicts and ensure the data integrity , and which transaction have the access for the database , Locks can be used at several levels, including the row or table level, and are necessary to enable isolation in transactions , and for the techniques it uses logging database operation so that will manage the lock for concurrency controls , and for the data structure is uses the logfiles for auditing and lock tables for handling the access of the database elements .

The employment of threads for various administration activities (passengers, flights, etc.) emphasizes a requirement for these components to handle concurrency control properly and to log operations for each thread, even though the code doesn't explicitly include a log system or a lock table.

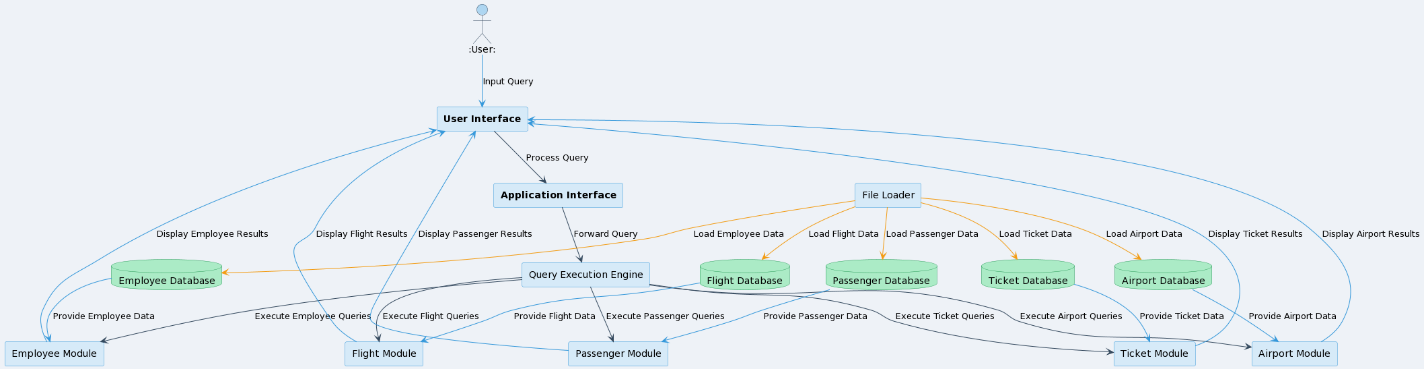
1. **DDL compiler 🡺** The DDL compiler is responsible for reading or interpreting and executing commands written in data definition language , so these commands have a specific purpose to add or edit or delete form and into the database , so then the DDL compile these commands and then trans them into a set of tables , or to metadata , so now let us go for the techniques that the DDL uses it uses the shecma upadate opertaring and also the parsing DDL staments that contain the mann grammers and syntax for the quieris , and for the data structure is use the tables or metadata tables , and also sysntax tree or parsing tree for pasrsing .

## Task3

### dfd

A data flow diagram (DFD) is a picture that shows how data moves through a system. It makes it easier to comprehend how the system functions and how data moves through it by providing examples of how data enters the system, how it is transformed, and where it is kept.

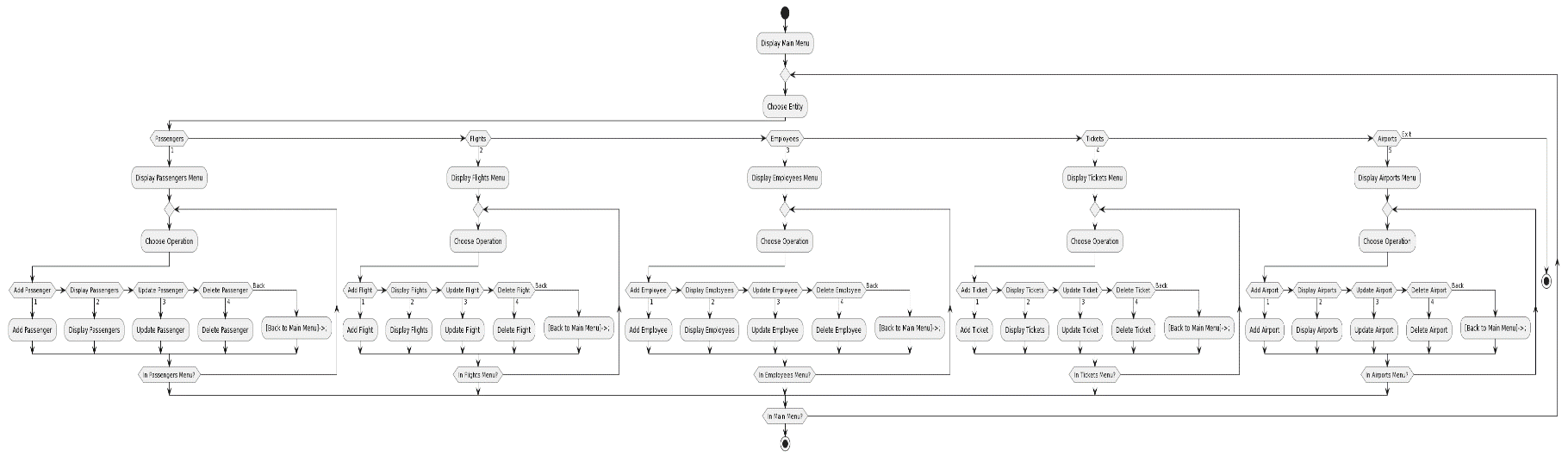
A visual depiction of the flow of data through a system is offered by DFDs. This covers the sources, destinations, and intermediate data transformation procedures for data. It is helpful to comprehend how data enters, gets processed, and is stored in the system while discussing databases.



#### Evaluate the ability to transfer data between different DBMS components

As we saw in the DFD the Passengers, Flights, Employees, Tickets, and Airports to Management Systems is directly connected to its entities , so that will applies a straightforward mechanism for the data modification in the database , and indicating a good flow of the data so that will improve the management layer .Create, Update, Delete Operations: as in the provided DFD the CRUD operations have a clear paths of managing the components to the respective databases for creating and deleting and updating , so that shows that the system contain an effective and clear modification for the data within the database , but for the read operation it will have a reserved arrow as shown in the diagram , so that indicates that the system can read or retrieve data efficiently , and this includes the fetching processes .Data Integration Across Different DBMS Components: Inter-Component Communication: in the DFD the data flow between several databases , such as the relation between flight database and the employee data baes is not specially depicted in the DFD , data flow pathways may be required if such interactions are needed (e.g., cross-referencing staff assignments with flight itineraries). System Scalability🡺 The modular nature of the system (separate databases and administration components for each entity) suggests that it may be easily scaled. It should be feasible to expand or add new functions without affecting other system components. Flexibility to Facilitate New Technology Integration: Even though everything is set up nicely now, it's crucial to ensure that the system design is adaptable enough to allow for the inclusion of new parts or technologies in the future.

### flowchart



Note 🡺 I will show this in the oral discussion

## Task 4 Effectiveness of the database and DBMS

* **User Requirements Evaluation**
* Passengers Module:

Effectiveness: they are able to handle and deal with all the operations that is related to there data , or in the whole system (other tables ), so they have the ability to do add , delete , update and display passengers info and more .

Assessment: we can test each operation so we can assess each function for adding and delete so this how we can make so of the integrity and privacy of the data , and how the effective they can use the system .

* Flights Module:

Effectiveness 🡺 we have a flight management system that the users will be able to manage the system by adding or deleting flights, and also they are able to display the flight lists and manage them.

* Employees Module:

Effectiveness 🡺 the employee is capable to manage the employee and there data effectively, and all the operation that it is related to it, covering additions, deletions, updates, and displaying of employee information and other information’s.

* Tickets Module:

Effectiveness 🡺 same for the other attributes but we have a weak entity so the ticket so we will have two entities are connected to this entity the passenger and the flight so the ticket will be connected to these , and it must have the access to add or update delete info

* System Operations:

Effectiveness: The system should provide simple and dependable system exit functionality together with easy navigation across several modules.

Evaluation: Conduct usability tests with an emphasis on the way users interact with the menu and exit functionality. Keep track of the time it takes to finish routine tasks and get user feedback.

* **System Requirements Evaluation:**
* Structures and Functions:

Effectiveness: we will have in our system, including passengers, planes, staff, tickets, and airports, so we must have a well-defined structure for functions, and also have a clear and effective function to handle different type of operations , such as adding ,removing , updating, and presenting data.

Evaluation: Examine the coherence of these structures as well as the functionality of the associated functions. Testing should include confirming that each function handles the data correctly and efficiently.

* User Interface:

Effectiveness: this will ensure the data integrity of the data, and the interface is simple and easy to use and to deal with, and it is a straightforward to the point , and also to fulfil the requirements for the user needs and system so that mean the user can do the operation that he must do .

Assessment: Examine the interface's usability to determine how responsive, understandable, and simple it is to use. Make that the interface design makes it easy to enter data accurately, navigate the system, and properly validate user input to prevent errors.

* Data Storage:

Effectiveness: this will ensure the dynamic data handling with a efficient memory allocation and deallocation, so we include the data storage techniques , so we make these so the data are stored in an efficient and it we be backed up and retrieved in more efficient way .

Evaluation: Examine the memory usage efficiency, retrieval speed, and overall data management performance of the data storage system. This entails evaluating the system's capacity to handle enormous volumes of data and its reaction times.

* Error Handling:

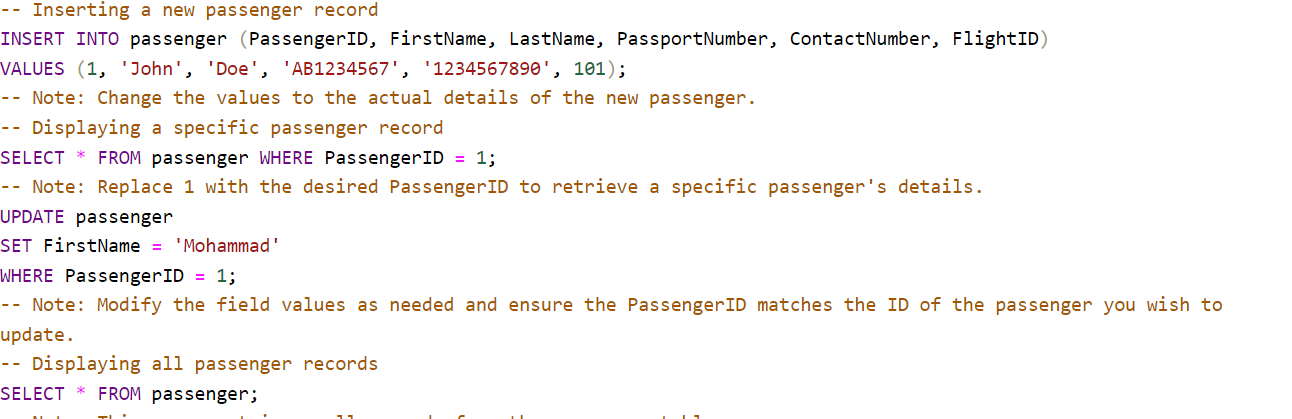
Effectiveness: Comprehensive error-handling tools should be built into the system so that, if a user enters data incorrectly or that an entity cannot be discovered, the system may provide them with constructive and intelligible feedback.

Evaluation: Run through several mistake scenarios to see how the system reacts. This involves confirming that the system reacts correctly to improper situations and that error messages are precise and understandable.

# PART 2

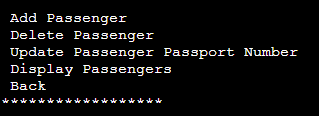
## TASK 1

* Passenger:



A white background with black and orange text

Description automatically generated



1. Insert

A screenshot of a computer screen

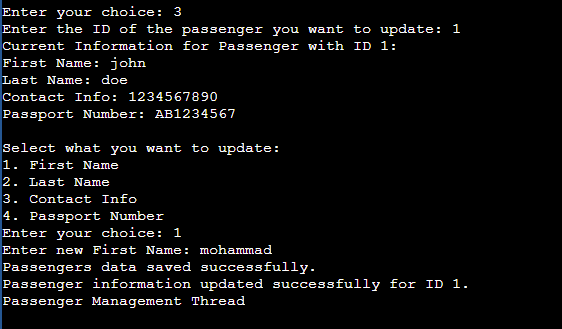
Description automatically generated

1. Display all

A screen shot of a computer

Description automatically generated

1. Update

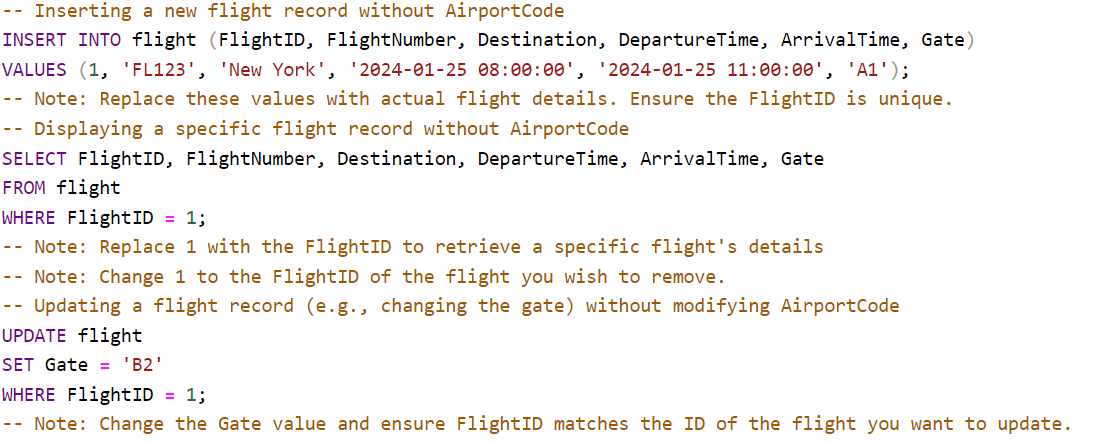


1. Delete

A screen shot of a computer

Description automatically generated

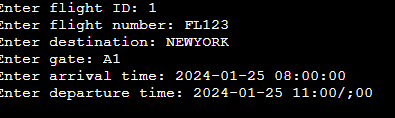
* Flights



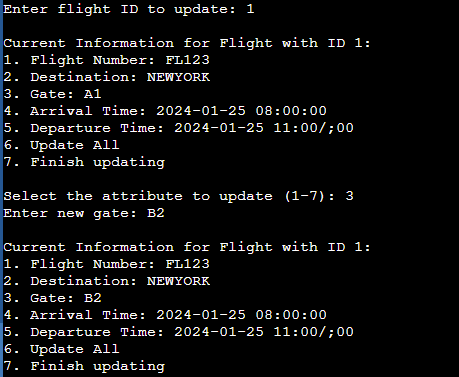
A close up of a computer screen

Description automatically generated

1. Insert



1. Update

7

1. Display

A screenshot of a computer program

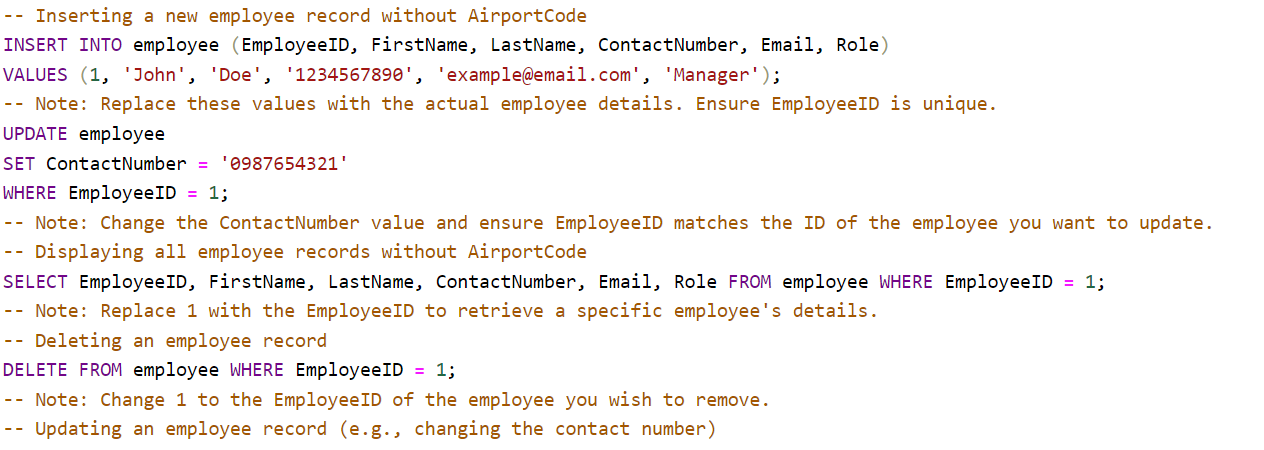
Description automatically generated

1. Delete

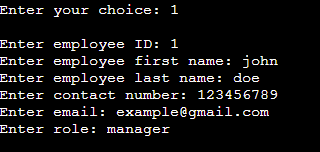
A black screen with white text

Description automatically generated

* **Employee**



1. Insert

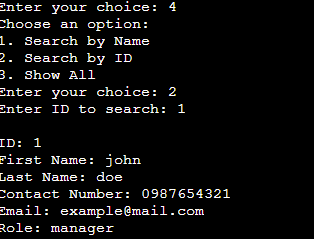


1. Update

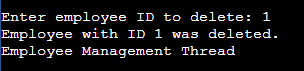
A computer screen shot of a black screen

Description automatically generated

1. Display



1. Delete

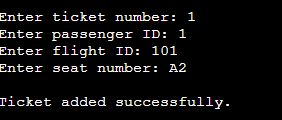


Ticket 🡺

A screen shot of a computer

Description automatically generated

* Add



* Update

A screen shot of a computer

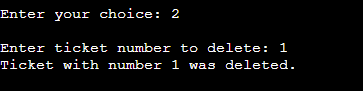
Description automatically generated

* Display

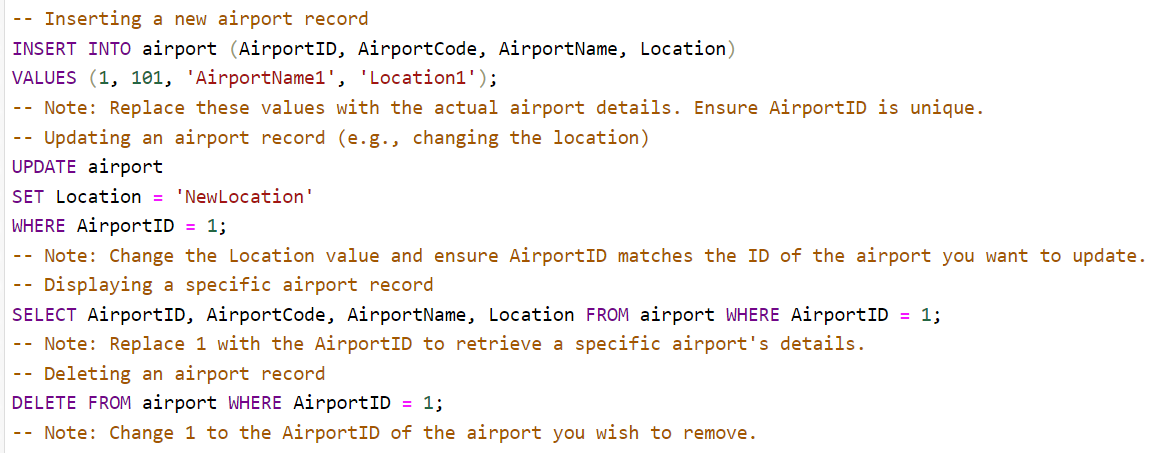
A screen shot of a computer

Description automatically generated

* Delete



Airport 🡺



* Insert

A black screen with white text

Description automatically generated

* Update

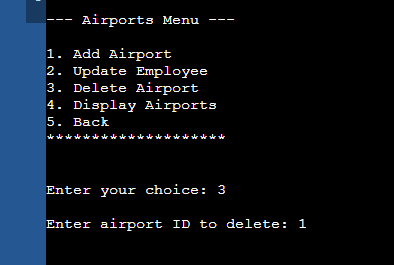
A screenshot of a computer

Description automatically generated

* display

A screenshot of a computer

Description automatically generated  
  
  
Delete



## An explanation of buffer managers and storage systems' dynamic data structures

The Airport Management System is made to manage several different things, including travelers, planes, staff, passes, and airports. Effective data administration and rapid retrieval processes are essential to the system's functionality. Sophisticated indexing techniques and dynamic data structures are used to accomplish this.

* Linked Lists:

The linked list is my primary data structure , and the main reason for using it is to handle each entity in the system and store them in sequence while using nodes, so that as a result will make the data processing and traversal more efficient so that will increase the overall efficiency.

And when talking about the allocation in the memory, in the linked list a lot of memory is allocated dynamically, which makes it easier to allocate memory at runtime according to actual data needs.

When using the data structure provides us with more data management and the system become more adaptable and effective.

* Dynamic Allocation:

The new nodes will be created and uses the dynamic memory allocation (malloc), that will help in guarantees that memory is allocated at runtime based on the actual data requirements.

Every node in the chain has a pointer to the next node and data fields pertinent to its entity. This makes it possible to traverse and manipulate the list (addition, deletion, update) efficiently.

## Indexing

Hash tables are the main indexing mechanism utilized in your application to maximize the efficiency of simulated SQL queries. Here's a thorough breakdown of how this indexing method is used:

* Structure: in my DBMS I have a number of entities such as passenger, flight, employee, ticket, or airport, so the hash table is linked to each one of them, so hash table is an array of pointers to hash nodes, each of which stands for a distinct record.
* Hash Function: when we are dealing with the hash tables, we need it is essential point and component to be added is the hash function. So it purpose is to compute an index in array for the given hash table given a key, in your example, an integer ID. Effective computation is ensured by the modulo operation's simplicity (key % HASH\_TABLE\_SIZE).
* Effective Lookups: the biggest benefits for the hash table is their ability to provide a quick search in the recodes , and the hash table have the access to index where the data is saved (average case O(1) time complexity) rather than having to gradually search through a linked list (O(n) time complexity).

## Criticize the process of mapping SQL queries to the developed CRUD statements and suggest improvements to optimize the developed CRUD quires.

The system I developed for a database management system emphasizes the need for increased functionality and flexibility, especially with respect to the Read, Update, Delete (CRUD) framework components, so expanding the query terms is necessary to improve the read or select function, this for sure will allow a wide range of queries as well as restring queries to a small number of pre-defined conditions , so by using a join query such as inner and right , this they will give a great improvement to the system over all and retrieval by merging data from multiple rows according to similar properties.

The Flexibility is limited by the current system's prohibitions on deletions for primary key operations, if the user could eliminate recorders based on the on factor other than only unique identification , it this will be the biggest improvements , and will make the system able to make possible for users to remove records that meet specific criteria more successfully, and also it is very important to make improvements to the update feature, the systems must now change the character based on the it main key .

The system can also be enhanced by adding sorting functions, that will include a ascending and descending order, this will help in organizing and displaying data, use techniques that make the system which is a boundary condition will allow you control over how many entries are obtained when working with large data collections.

The focus in creating an a more flexible so that , this will enhance aim to rectify current inadequacies and get the DBMS closer to the capabilities of complex SQL-based databases.

## Improvements

1. Complex Query Structures 🡺 for future use the users may do a complex SQL queries to handle the more complex data retrieve , , this kind of intricacy can make the code more complex but can be used for more usages This kind of intricacy can make the code harder to comprehend and maintain, which makes debugging and future changes more challenging.
2. Performance Bottlenecks Due to Query Inefficiencies🡺change some dependencies that may improve the performance , because the inefficient SQL queries have a high potential to seriously hinder system performance, and slow down the whole system , and Misuse of indexes, dependence on "SELECT \*", this are a common mistake and not effect usage and non-optimized subqueries are common mistakes, and slow the response time of the system over all , so we can do some queries that have the same data retrieved but with lower performance and also the system also should accept these queries
3. Challenges with Data Growth: in the future sure the system will handle new data with high quantities increase, CRUD activities that were not built with scalability in mind may become unmanageable. The overall performance of the system may be impacted because of these activities seeing longer execution times and higher resource needs as the dataset grows, so here we should make the CRUD operation accept these expanding of data .
4. Issues with Hardcoded Elements 🡺Any modification to the database structure that is hardcoded into SQL queries, such as column names, may result in query failures. Because of this rigidity, changing the schema may require significant rewrites of the code.

# Part 3

When several transactions are carried out concurrently in a Database Management System (DBMS), concurrency control is essential to preserving the consistency and integrity of the data. The main objectives of concurrency control strategies are to guarantee the safe execution of database transactions and to avert problems such as deadlocks and inconsistent data. Some common methods for concurrency control in DBMSs are listed below:

1. **Lock-Based Protocols:**

* Binary Locks 🡺 the lock state have two states (locked unlocked), and before accessing the database there should be a basic type of lock, when a transaction it will obtain a lock that will be released after the operation, and the drawback of the binary lock is that it cannot distinguish between write and read operations.
* Shared/Exclusive Locks (S/X Locks) 🡺 here in this approach the distinguishes locks based on the transaction's goal, there will be a shared lock that will allow transaction to access data while preventing other transactions to publish it at the same time, we will use exclusive locks in writing data or preventing other operations from reading or writing the same data.

1. **Timestamp-Based Protocols**

* So while using this protocol each transaction will take a specific timestamp based on its start time, and this time stamp indicates a specific order in which the transaction can ideally access the database.
* Timestamp Ordering: The sequence in which transactions access database entries is determined by their timestamps. A transaction is rolled back if it attempts to access data in a manner that is inconsistent with timestamp ordering (e.g., writing data that has already been read by a subsequent transaction).

1. **Multiversion Concurrency Control (MVCC):**

In the DBMS while using concurrency every data will be maintained in many versions by the MVCC ,and the non-blocking will be done easily because the transaction that reads the data items will access the version that was current at the beginning of the transaction , and the database will keep track of these versions , while keeping multiple versions of data necessitates more storage , so the MVCC is very important and effective in the context of reading .

1. **Validation-Based Protocols or** **Optimistic Concurrency Control (OCC ) :**

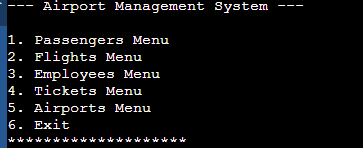
And these controls allows transactions to be executed without acquiring locks , so while compiling a validation step will be performed to make sure that the transaction is not interfered with other concurrent transactions , and if the validation failed the transaction will rolled back , so this control is very important and very useful for the databases that contain transactions that mostly non-conflicting , but the drawbacks of this control will when the transaction has conflicts because that will cause an overhead on the environment .

## Task 2

POSIX threads in C have been used to enable parallel data processing features in the developed DBMS for an airport management system, making the system more responsive and efficient. This is especially helpful in areas with a lot of traffic, like airports. The system allows for concurrent operations since each main component—Passengers, Flights, Employees, Tickets, and Airports—is managed by a separate thread. These modules can function independently and concurrently since they employ pthread\_create and pthread\_join for thread management. For instance, two threads can process flight information and passenger data simultaneously. The system's capacity to handle numerous processes at once is greatly improved by this architecture, which lowers wait times and boosts throughput overall. For data permanence, the system additionally includes file operations, with each module having its own specific text file, like passengers.txt, for passenger data, guaranteeing fast and well-organized data handling. Because of its ability to handle multiple activities at once, this threading strategy makes the database management system (DBMS) very suitable for the demanding and dynamic world of airport administration.

# Part 4

## Task 1



## Task 2

**Test plan**

1. **Unit Testing**

This entails evaluating certain elements of a database system, such as triggers, stored procedures, and functions. Test cases are generated from distinct testing databases or dummy data for a variety of scenarios. Efficiency can be increased by using automation tools like salt for SQL Server or papa for PostgreSQL. The main advantage is in promptly identifying and resolving problems at the lowest possible system level.

1. **integration Testing**

This phase, which comes after unit testing, focuses on how the DBMS's integrated components or modules interact with one another. Ensuring reliable information flow and proper operation of various components is the goal. To simulate and validate these interactions—a crucial step in identifying and resolving interface flaws—tools such as JUnit, Unknit, or TestNG are utilized.

1. **Performance Testing**

Testing is crucial for assessing how a DBMS behaves under different load scenarios since it gauges responsiveness, scalability, and reliability. It consists of load testing to evaluate capacity under average and peak conditions, as well as stress testing to identify system limits. Multiple users are simulated using tools such as LoadRunner or Apache JMeter to make sure the system operates smoothly under anticipated traffic and transaction levels.

1. **Backup/Recovery Testing**

This evaluates how well data backup systems guard against corruption or loss of data. To test recovery techniques, data loss or corruption is first introduced and then backups are created. The objective is to evaluate the system's resilience and disaster recovery method integrity in order to guarantee that data can be reliably and rapidly restored.

### The test plan.

#### Passenger

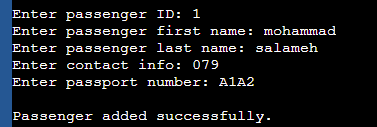
1. Creating new passengers and save the result to file.

* Inserting from the code 🡺 the code should show the inserted values

Test 🡺 id = 1 , first name = mohammad , last name = salameh , contact info = 079, passport number = A1A2.

Result :-

1.The code first will show the input in the code



2. Result to File 🡺 the inserted data should be saved to the file.



* And when insert a new passenger the file should be able to handle multiple inputs 🡺 the file should handle more than one entry so , in each time a passenger is inserted the file should handle it .

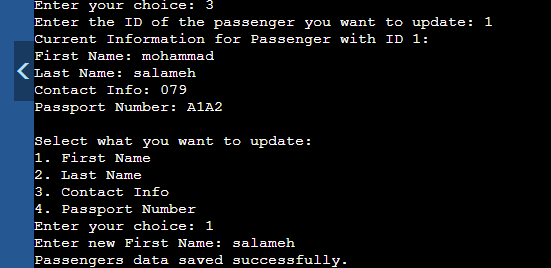


* When inserting two passengers with the same id the system should not accept the new passenger.

We already have a passenger with the id = 1 , the code should not allow this entry .



1. Updating the passenger:

Update the passenger info based on the given ID , so the user can update any attributes or update all and when doing this process this will reflect on the files .  
  
this is how the code will handle this update 🡺  


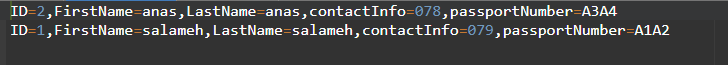
This is how the file will look after editing the last name 🡺



1. Display

Here I want to test the display functionality when I enter the display menu he will ask me about the id and just display the data for the entered id .

This the file content 🡺

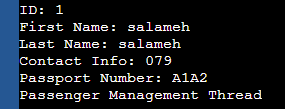


Search by the ID and this is the what will appear , and it is the right data for the ID .

A black screen with white text

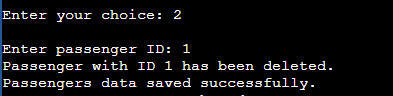
Description automatically generated

Search by the name and this is the what will appear , and it is the right data for the name .



1. Delete

When applying the delete function, the user enter the id of the of the passenger and it will be deleted from the files and a message will appear .



And the passenger as we see will be deleted from the files also.



### Flights

Data

Flight ID 🡺 101

Flight number 🡺az1

Destination 🡺amman

Gate 🡺 01

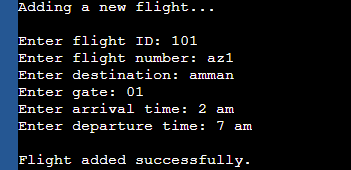
Arrival time 🡺 2 am

Departure time 🡺7 am

* Insert

Create a new flight and add it to the files 🡺

The first point the code will appear the first look the flight is created and inserted.



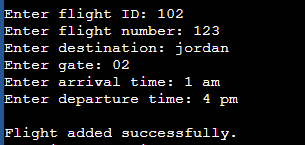
And also, when making new flight the data should be print File 🡺 and this is the file result after creating the new flights



Adding more than one flights 🡺 the files should handle more than one flight, so when adding new flight it will also be saved in the file .

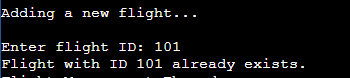
Test 🡺 Flight ID 🡺 101 , Flight number 🡺az1, Destination 🡺amman , Gate 🡺 01 ,Arrival time 🡺 2 am Departure time 🡺7 am

Result



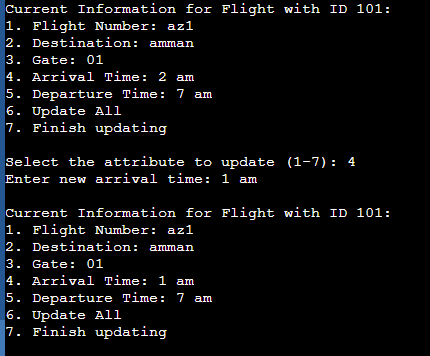


Inserting flights with same ID number 🡺 the expected result after this entry the system should not allow this entry .



* Update

Update the flight info based on the given ID , so the user can update any attributes or update all and when doing this process this will reflect on the files .  
  
this is how the code will handle this update 🡺change the flight with the 101 id number arrival time form 2 a.m. to 1 am



The arrival time has been changed successfully.

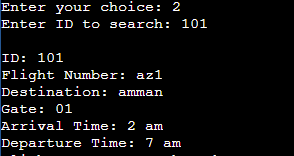
* Display

This is the file content for the flight entity 🡺



Displaying the right items for the entered id 🡺

Id = 101 , the attributes that have the id =101 id number will be printed.



* Delete

Here I want to test if the flight will be deleted also from the file free space , so we already have two flights , so I want to delete the 102 flight and it attribute , and also to make all the recod of the flight after deleting it will be deleted .

Expected result

Is we will only have one record one recod with 101 id and its data .



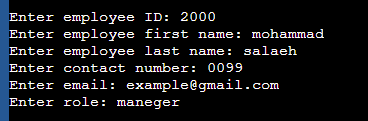
And the file 🡺



### Employee

Insert

Create a new employee and add it to the files 🡺 the expected result will be that the employee should be successfully added to the files and a successful message will appear



File 🡺 and this is the file result after creating the new employee  


Adding more than one employee  
  
A computer screen with white text

Description automatically generated  
  

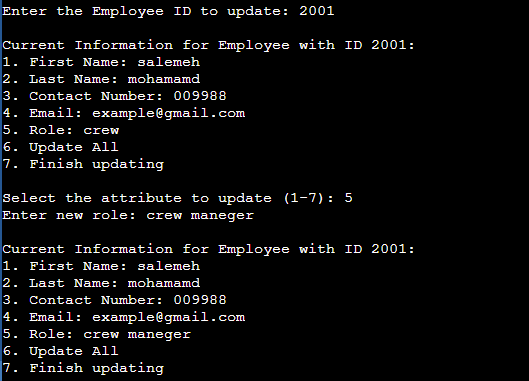



Inserting employee with same ID number

A black background with white text

Description automatically generated

Update

Update the role for the 2001 employee to crew maneger , and that should updatye the employee with id 2001 role , and also this update should also change the role for that employee in the files .  




Display   
 we want to display all info about the employee with the 2000 data , and this should only print that employee data only.

A screen shot of a computer

Description automatically generated



Delete

Remove the employee with id 2001, and also it should be remove d from files

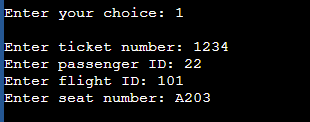
A black screen with white text

Description automatically generated  


### Tickets

Insert

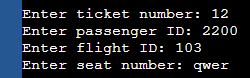
Create a new ticket and add it to the files 🡺 as same as the other entities the when inserting a new ticket the code should show that the tickets has been added , and also saved to the file



File 🡺 and this is the file result after creating the new flights



Adding more than one flights 🡺 expected result is that the file should handle for than one input.

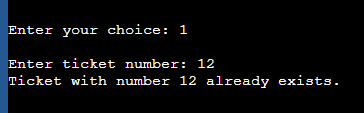


And un the file 🡺

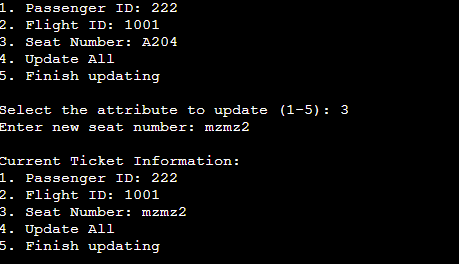
A screen shot of a computer

Description automatically generated

Inserting flights with same ID number , expected result is the when a system user to add new ticket it will not allow it.



Update : the update should change a specific record , and the new data will be printed on the main menu and edited in files , how I will test it I I will change the steat number of the ticket ID , will the seat number



A screen shot of a computer

Description automatically generated

As we can the result is passed and data aare changed .

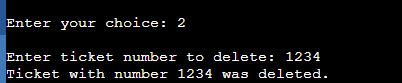
Display 🡺

I will display all ticket data that have the 1234 ticket numbers that is mainly , and it should only show the write data only that is related to that ticket number .

A screen shot of a computer

Description automatically generated  
A close up of a sign

Description automatically generated

Delete 🡺  
when deleting and ticket it should ask me about the ticket number and then it will delete the ticket number and it data (passenger id , flight id , seat number ), from the files .  
  


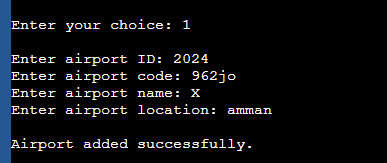
A close up of a sign

Description automatically generated

### Airport

Insert   
Create a new airport and add it to the files 🡺 as same as the other entities the when inserting a new airport the code should show that the airport has been added , and also saved to the file

Case 🡺 inserting an new airports with these attributes 🡺



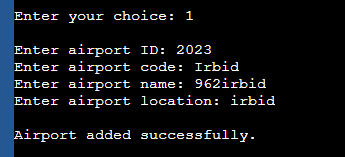
Result the code will show that the airport was added successfully and the added airport will be saved in files 🡺



Case 🡺 we already added a new airport and I want to add a new one , so here I will test if the files can handle more than one entry 🡺 expected result is that the file should handle for than one input.

A screen shot of a computer

Description automatically generated



And the file 🡺

A screenshot of a computer

Description automatically generated

Inserting flights with same ID number, expected result is the when a system user to add new ticket it will not allow it.  
  
A black screen with white text

Description automatically generated

update

Update the flight info based on the given ID , so the user can update any attributes or update all and when doing this process this will reflect on the files .  
  
this is how the code will handle this update 🡺change the airport location with the 2023 id number from irbid to aqaba , the expected result changing the location and the updated infor should be saved in the files with it new location .

A black background with white text

Description automatically generated

display   
   
displaying a specific airport info , so when we enter the airport id only the info that is related to that id will be printed , and the other data with different id wont be printed 🡺

A screenshot of a computer

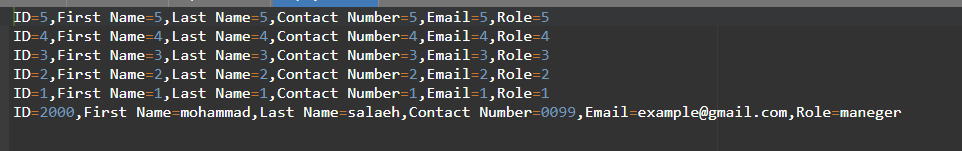
Description automatically generated

A black background with white text

Description automatically generated

## File loading testing

In this test plan I want to test the files stsemtrs , when starting the code the files files should be loaded scussfully , so that will ensure the data retrieval foe files , and as a way of backup.

Test 🡺 we have a employee files with this info , and I will make a select all query that will show every content in the file , without reentering them .  
  
  
  
expected result 🡺 the is that all the file contetnt will displayed for the user .  
  
A black screen with white text

Description automatically generated  
A screenshot of a computer

Description automatically generated

## Evaluate the improvements

The "My Airport DBMS" is intended to perform better and be more scalable, secure, and maintainable with the proposed changes. These improvements aim to create a more reliable, effective, and safe database system by tackling certain issues that arise in a dynamic, high-volume data environment. Performance measurements, code reviews, security audits, and operational efficiency evaluations can all be used to gauge how well these changes are working.

* Refactoring Complex Query Structures

1. Impact: when the SQL queries is improved, and optimized, this can affect in a positive way on the readability and maintainability of the code, and this as a result will make the updating the and the fixing and troubleshooting more easier and simpler and at the end this will increase the overall effectiveness of the database, and for the evaluation for the query structure 🡺and this as a result will Enhance the code readability , and less time time for query execution time and shorter query execution times, as determined by code audits or peer reviews, are indicators of success.

* Optimizing Query Performance

1. Impact: we are working on a system that will be relied on for the airport management, so this a result will greatly improve the speed of the system response time and will increase the overall performance in general. and this will keep an eye on the performance metrics CPU and memory use, as well as query response times and the evaluation of this point. This is especially important in high-data volume contexts, such as airports.

* Scalability in Data Growth

1. Impact: by making the a scalable database and can extend depending on the requirements , so after this the system will be able to handle growing data quantities without having any type of delaying in the performance , and this will ascertain whether the system can withstand peak data loads and growth trends without suffering from a decline in performance.

* Improved Database Error Handling

1. Impact: having an mistake and reporting and handling problems this will give the system more precise insights into the problems , and make the system make a quicker fixes and boosting dependability .

# Reference

* <https://www.appdynamics.com/topics/database-management-systems#:~:text=Optimizing%20database%20performance-,What%20is%20DBMS%3F,delete%20data%20in%20the%20database>.
* <https://www.spiceworks.com/tech/cloud/articles/database-management-systems-dbms/>
* <https://codeinstitute.net/global/blog/database-programming/>
* <https://learn.org/articles/What_Does_a_Database_Programming_Professional_Do.html>
* <https://www.scaler.com/topics/dbms/dbms-architecture/>
* <https://www.tutorialspoint.com/explain-the-architecture-of-dbms>
* <https://www.javatpoint.com/threads-in-operating-system>