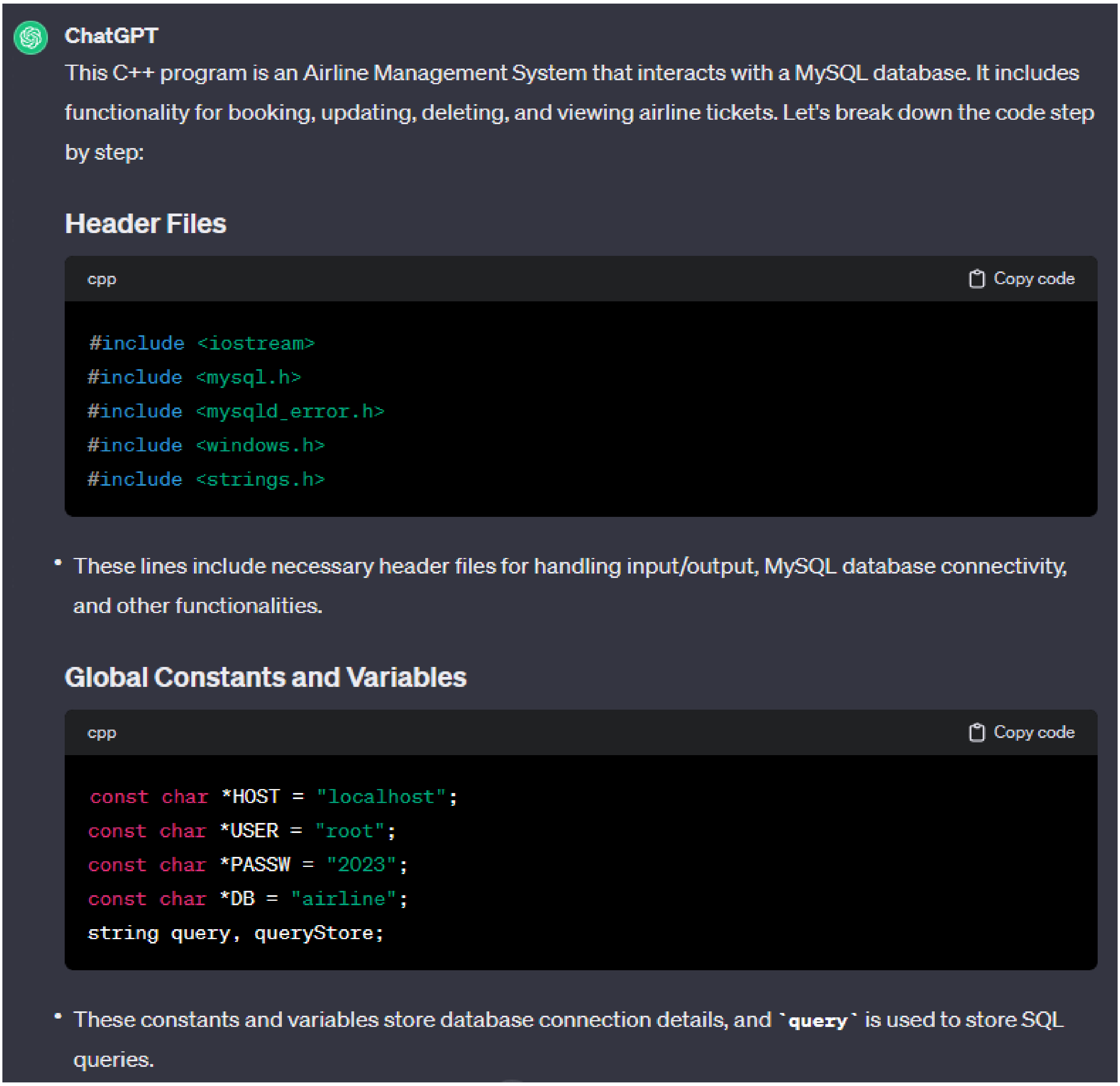


This C++ program is an Airline Management System that interacts with a MySQL database. It includes functionality for booking, updating, deleting, and viewing airline tickets. Let's break down the code step by step:

Header Files

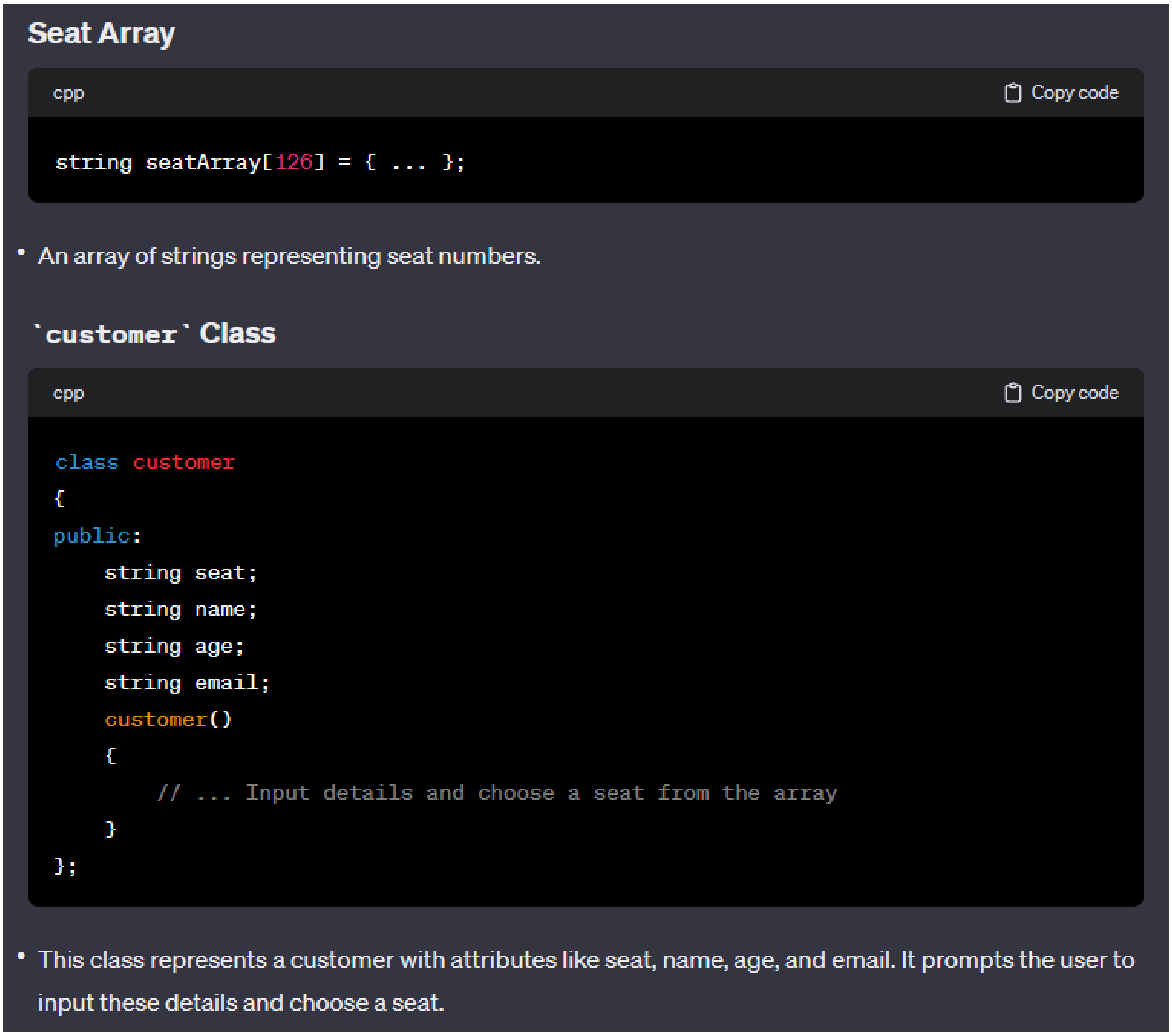
```cpp
#include <iostream>
#include <mysql.h>
#include <mysqld_error.h>
#include <windows.h>
#include <strings.h>
```

- These lines include necessary header files for handling input/output, MySQL database connectivity, and other functionalities.

Global Constants and Variables

```cpp
const char *HOST = "localhost";
const char *USER = "root";
const char *PASSW = "2023";
const char *DB = "airline";
string query, queryStore;
```

- These constants and variables store database connection details, and `query` is used to store SQL queries.



Seat Array

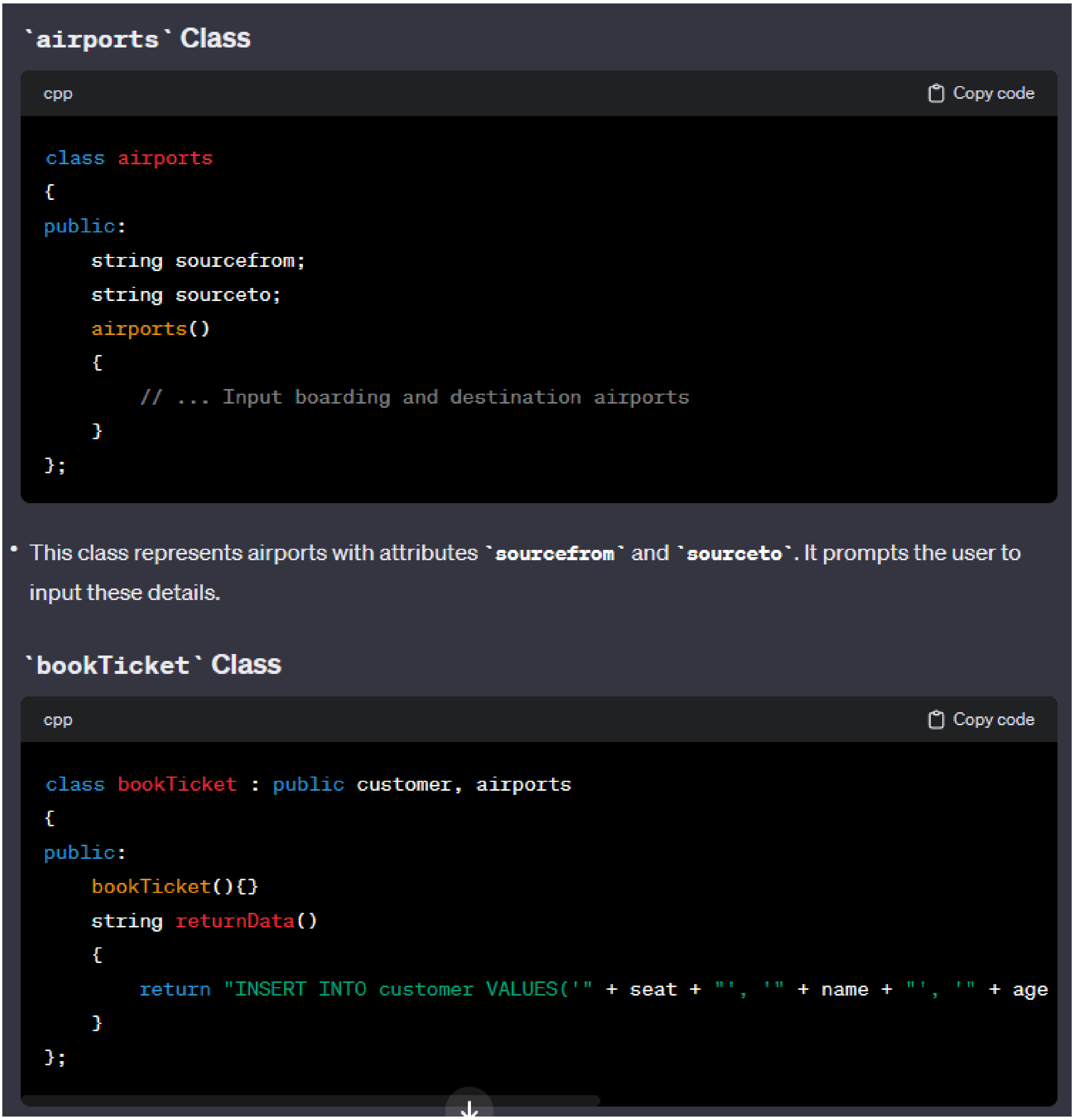
```cpp
string seatArray[126] = { ... };
```

- An array of strings representing seat numbers.

`customer` Class

```cpp
class customer
{
public:
    string seat;
    string name;
    string age;
    string email;
    customer()
    {
        // ... Input details and choose a seat from the array
    }
};
```

- This class represents a customer with attributes like seat, name, age, and email. It prompts the user to input these details and choose a seat.



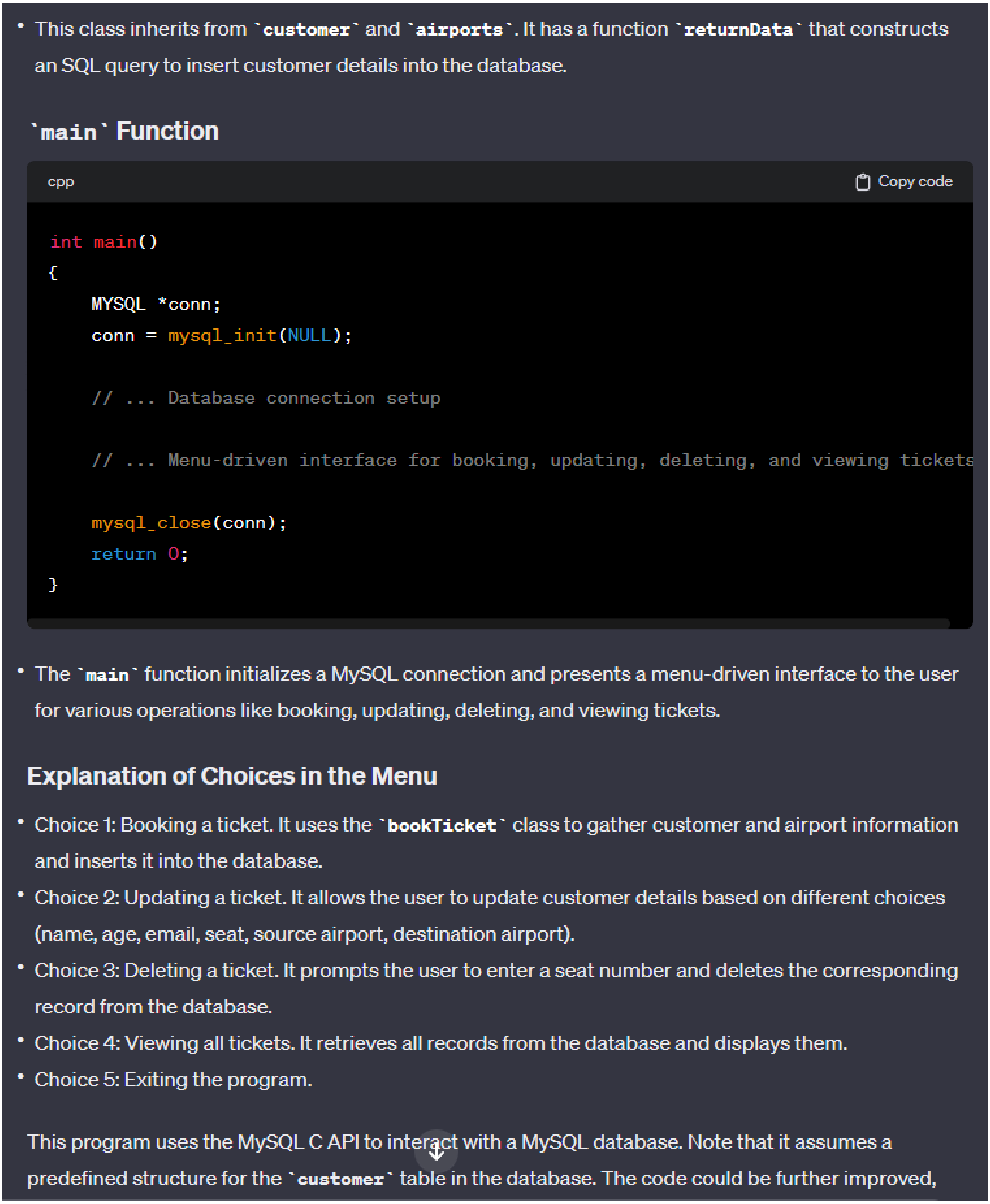
`airports` Class

```cpp
class airports
{
public:
    string sourcefrom;
    string sourceto;
    airports()
    {
        // ... Input boarding and destination airports
    }
};
```

- This class represents airports with attributes `sourcefrom` and `sourceto`. It prompts the user to input these details.

`bookTicket` Class

```cpp
class bookTicket : public customer, airports
{
public:
    bookTicket(){}
    string returnData()
    {
        return "INSERT INTO customer VALUES('" + seat + "', '" + name + "', '" + age
    }
};
```



- This class inherits from `customer` and `airports`. It has a function `returnData` that constructs an SQL query to insert customer details into the database.

`main` Function

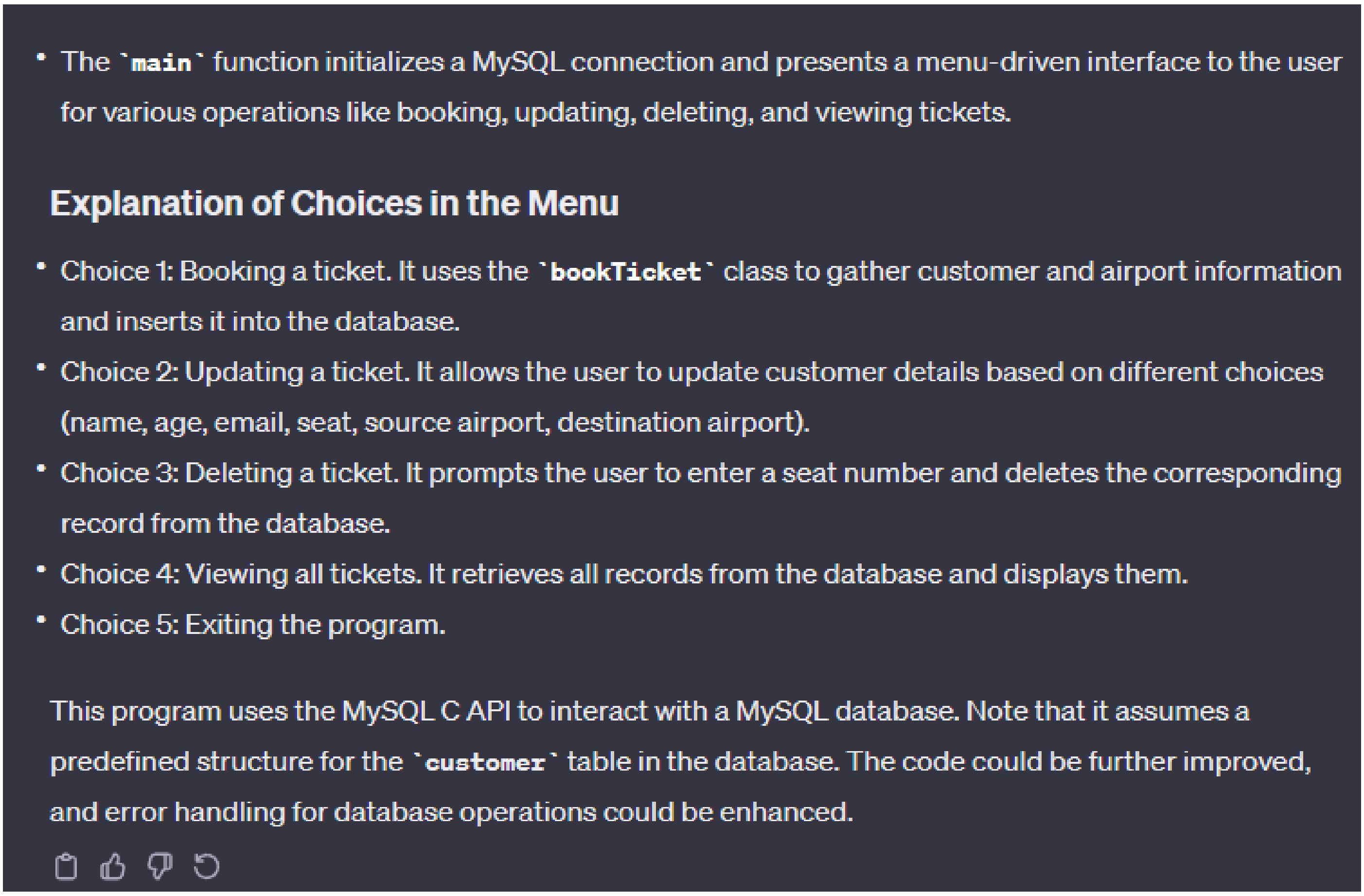
```cpp
int main()
{
    MYSQL *conn;
    conn = mysql_init(NULL);

    // ... Database connection setup

    // ... Menu-driven interface for booking, updating, deleting, and viewing tickets

    mysql_close(conn);
    return 0;
}
```

- The `main` function initializes a MySQL connection and presents a menu-driven interface to the user for various operations like booking, updating, deleting, and viewing tickets.

Explanation of Choices in the Menu

- Choice 1: Booking a ticket. It uses the `bookTicket` class to gather customer and airport information and inserts it into the database.
- Choice 2: Updating a ticket. It allows the user to update customer details based on different choices (name, age, email, seat, source airport, destination airport).
- Choice 3: Deleting a ticket. It prompts the user to enter a seat number and deletes the corresponding record from the database.
- Choice 4: Viewing all tickets. It retrieves all records from the database and displays them.
- Choice 5: Exiting the program.

This program uses the MySQL C API to interact with a MySQL database. Note that it assumes a predefined structure for the `customer` table in the database. The code could be further improved,



- The `main` function initializes a MySQL connection and presents a menu-driven interface to the user for various operations like booking, updating, deleting, and viewing tickets.

Explanation of Choices in the Menu

- Choice 1: Booking a ticket. It uses the `bookTicket` class to gather customer and airport information and inserts it into the database.
- Choice 2: Updating a ticket. It allows the user to update customer details based on different choices (name, age, email, seat, source airport, destination airport).
- Choice 3: Deleting a ticket. It prompts the user to enter a seat number and deletes the corresponding record from the database.
- Choice 4: Viewing all tickets. It retrieves all records from the database and displays them.
- Choice 5: Exiting the program.

This program uses the MySQL C API to interact with a MySQL database. Note that it assumes a predefined structure for the `customer` table in the database. The code could be further improved, and error handling for database operations could be enhanced.