Team: loop99

Project topic: Restaurant menu viewer.

System requirements:

1. **Mysql version 8.0.35 or higher.**
2. **Some MySql related modules**

Steps to be followed:

1. **Please create a user in mysql with username: "root1" and password: "54325" in "localhost" (recommended)**

   Or

   Change the username, password and host in the code in functions.cpp file as per your system requirements, this file has 4 functions .

2. **Grant all permissions to the user.**

3. **MySQl:**

   Step 3.1: Create a database named "restaurant".

   Step 3.2: Create table "hotels". Below is the structure and sql syntax of the table "hotels"

```
CREATE TABLE hotels (
    name varchar(255),
    ratings decimal(3,2),
    address varchar(255),
    contact_number varchar(15),
    type varchar(50),
    id varchar(5)
);
```

   Step 3.3: Create table "menu". Below is the structure and sql syntax of the table "menu"

```
CREATE TABLE menu (
    id INT AUTO_INCREMENT PRIMARY KEY,
    hotel_name VARCHAR(255) NOT NULL,
    starters VARCHAR(255) NOT NULL,
    main_courses VARCHAR(255) NOT NULL,
```

```
    desserts VARCHAR(255) NOT NULL,
    beverages VARCHAR(255) NOT NULL,
    starter_price DECIMAL(10, 2) NOT NULL,
    main_price DECIMAL(10, 2) NOT NULL,
    dessert_price DECIMAL(10, 2) NOT NULL,
    beverages_price DECIMAL(10, 2) NOT NULL
);
```

Step 3.4: Insert all the data from csv file to mysql into the table "menu". The c++ code to insert data into table "menu" in the database "restaurant" from the file menu.csv has been given in the file insert.cpp. Run the code to insert the data.

Or

Copy the mysql file "restaurant" to your mysql database path given in the folder.

4. **After the set-up of MySql, compile main.cpp and functions.cpp, link them and run. (While linking, some std c++ library and mysql library files might be required. The two which we used are "-lstdc++" and "-lmysqlcppconn")**

How to read the code

Main.cpp

Header and Namespace:
The code includes necessary header files (iostream, limits, string) for input/output operations, handling limits, and using strings. The using namespace std; statement allows using standard
C++ identifiers without prefixing them with std::

Function Prototypes:

1. fetchHotelData(const string &hotelType): Fetches hotel data based on the hotel type.
2. selectMenuByHotelID(const string &hotelID): Selects a menu based on the hotel ID.
3. book_table(const string &id, const string &dineDate): Books a table based on the ID and date.
4. search(string itemName, double minPrice, double maxPrice, double minRating): Searches based on item name, price range, and minimum rating.

Class Definition (HotelManager):

Encapsulates functionality related to managing a hotel system.

- Private members: hotelID, date, start (flag for controlling the loop), and choice (for user input).
- Private member functions: Displaymainmenu, getValidChoice, handleRestaurantMenu, displayRestaurantMenu, handleHotelType, displayHotelOptions, handleHotelMenu, displayBookTableOptions, bookTable, confirmBooking, displayConfirmationOptions, handleSearch, getUserInputs.
- Public member function run(): Main function to execute the hotel management system.

Main Loop (run() function):

The program runs in a loop controlled by the start flag. Displays the main menu, prompts for a choice, and performs input validation for non-numeric inputs.

Main Menu Options:

1. If the user chooses option 1, it calls handleRestaurantMenu().
2. If the user chooses option 2, it calls handleSearch().
3. If the user chooses option 3, the program exits.

Input Validation (getValidChoice() function):
Ensures user input is a valid integer. Clears the error flag and discards invalid input.

Restaurant Menu Handling (handleRestaurantMenu() function):
Displays a sub-menu for selecting restaurant types (Veg, Non-veg). Calls handleHotelType() based on the user's choice.

Hotel Type Handling (handleHotelType() function):
Displays hotel options based on the selected type (Veg, Non-veg). Calls fetchHotelData() to get hotel information. Calls handleHotelMenu() or bookTable() based on user choice.

Hotel Menu Handling (handleHotelMenu() function):
Asks the user to enter a hotel ID and calls selectMenuByHotelID() to display the menu. Allows the user to book a table.

Booking Table (bookTable() function):
Takes user input for hotel ID and date, then calls book_table() and confirmBooking().

Confirmation Handling (confirmBooking() function):
Asks the user to confirm the booking and displays a success message if confirmed.

Search Handling (handleSearch() function):
Allows the user to search for items based on name, price range, and minimum rating.
Calls bookTable() if the user chooses to book a table.

User Input Function (getUserInputs() function):
Takes input for item name, minimum price, maximum price, and minimum rating.

Functions.cpp

Header Files:
Includes various MySQL and C++ headers for database connectivity and standard input/output operations.

Namespace:
The using namespace std; statement allows the use of standard C++ identifiers without prefixing them with std::.

Schema (Database) Selection:
con->setSchema("restaurant");: Sets the active database schema to "restaurant."

Prepared Statements:
Prepared statements (sql::PreparedStatement) are used to execute SQL queries with placeholders.
The first prepared statement retrieves the hotel name based on the given hotel ID.
The second prepared statement retrieves menu details based on the same hotel ID.

Result Set Processing:
The result set (sql::ResultSet) is used to store the results of the executed queries.
If the hotel is found (if (res->next())), the hotel details are displayed along with the menu in a formatted table.
The individual fields of the result set are accessed using res->getString and res->getDouble, and the information is printed in a tabular format.

Error Handling:

If an SQL exception occurs, it is caught and an error message is displayed.

Resource Cleanup:
delete statements are used to free the memory allocated for the result set, prepared statements, and the database connection.

Function Signature:
void fetchHotelData(const string &hotelType): Defines a function named fetchHotelData that takes a constant reference to a string (hotelType) as a parameter and does not return any value (void).

Try-Catch Block:
Encloses the code within a try-catch block to handle potential SQL exceptions.

Statement Creation:
Create a statement object for executing SQL queries: sql::Statement *stmt = con->createStatement();.

Query Execution and Column Width Determination:
Execute an SQL SELECT query to retrieve hotel data based on the provided hotel type: string query = "SELECT id, name, ratings FROM hotels WHERE type = '" + hotelType + "'"; and sql::ResultSet *res = stmt->executeQuery(query);.
Determine maximum column widths by iterating over the result set and finding the maximum lengths for each column.

Result Set Rewind:
Rewind the result set to the beginning for actual printing: res->beforeFirst();.

Results Printing (Header):
Print the header for the results table with proper formatting, using the determined maximum column widths.

Results Printing (Data):
Iterate over the result set and print the actual hotel data in a tabular format, using the determined maximum column widths.

Resource Cleanup:
Release the memory allocated for the result set, statement, and connection: delete res;, delete stmt;, delete con;.

Exception Handling (if SQL Exception occurs):
If a SQL exception occurs, catch it and print an error message containing details such as file name, function name, line number, error message, error code, and SQL state.

Function Definition (selectMenuByHotelID):
Displays the menu for a given hotel ID.

- Enclosed in a try-catch block to handle SQL exceptions.
- Establishes a connection to a MySQL database, sets the active schema, and uses prepared statements to retrieve hotel and menu details.
- Processes the result set, displaying hotel details along with the menu in a formatted table.
- Handles SQL exceptions and performs resource cleanup.

fetchHotelData Function Signature:
Defines a function named fetchHotelData that takes a constant reference to a string (hotelType) as a parameter and does not return any value (void).

- Establishes a connection to a MySQL database, sets the active schema, and uses a try-catch block to handle potential SQL exceptions.
- Creates a statement object, executes an SQL SELECT query to retrieve hotel data based on the provided hotel type, and determines maximum column widths.
- Prints the header and actual hotel data in a tabular format.
- Performs resource cleanup and handles SQL exceptions.

book_table Function:
- Purpose: Books a table at a hotel based on the provided hotel ID and dining date.
- Database Interaction: Connects to a MySQL database to retrieve information about the hotel using the provided hotel ID.
- Booking Information: Prints booking information if the hotel is found, including the hotel name, ID, a randomly generated table number, dine date, address, and contact number.
- Exception Handling: Throws a runtime error if the provided hotel ID is invalid. Catches and handles SQL exceptions, printing an error message if they occur.

search Function:

- Purpose: Performs a search for hotels and their menu items based on user-defined criteria, including item name, price range, and minimum rating.
- Database Interaction: Connects to a MySQL database to perform multi-step filtering based on the provided search criteria.
- Search Criteria: Utilizes temporary tables to filter menu items based on a relatively matched item name, price range, and minimum rating.
- Results Display: Prints hotel information and menu items that match the search criteria.
- Exception Handling: Catches and handles SQL exceptions, printing an error message if they occur.

Some links to ChatGpt prompts

1. https://chat.openai.com/share/b5b835f2-2839-4248-89b8-be4aab2b7c17
2. https://chat.openai.com/share/343b2789-9821-45a6-b947-c4134c9a423d
3. https://chat.openai.com/share/a4bf5183-c534-4bba-a6ee-ee412ce3847a
4. https://chat.openai.com/share/34d0066c-7261-4da5-8e3a-5a88712e6280
5. https://chat.openai.com/share/2132f095-2af1-4023-a168-d02d9be3bec1