Hotel Management System

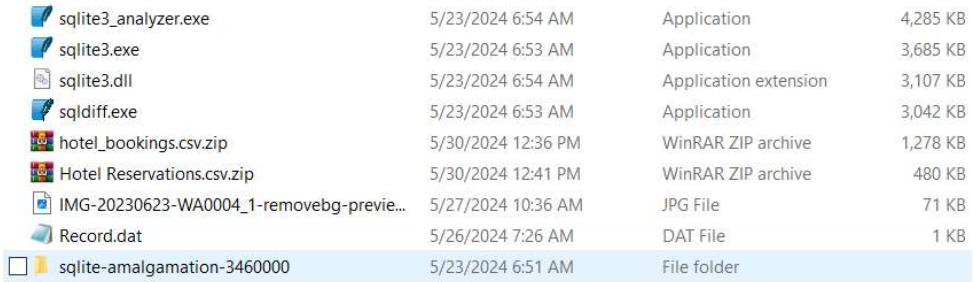
Introduction

The Hotel Management System project is designed to manage hotel room allocations and customer information efficiently. It utilizes a SQLite database to store and retrieve data, and a C++ application to interact with the database and provide a user interface for managing hotel operations. The key functionalities of the system include room allocation, searching, updating, and deleting room records, as well as viewing all records.

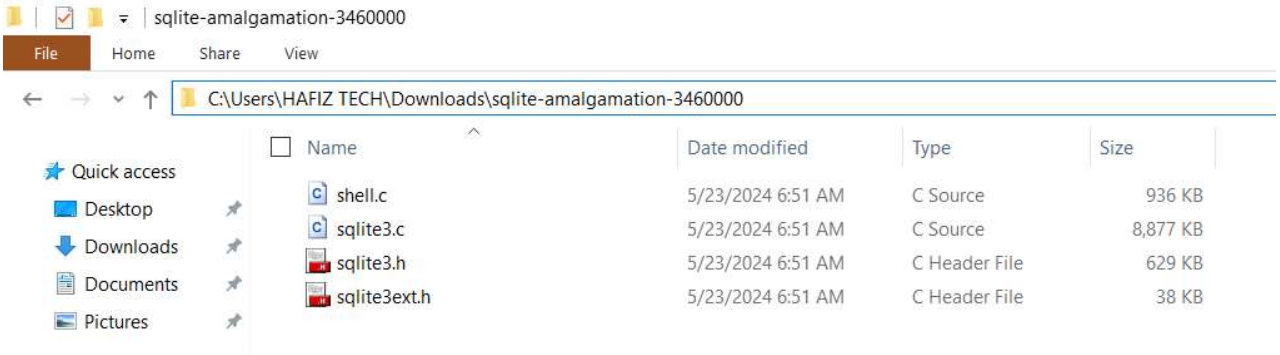
Connecting SQLite Library

Install the sqlite3 library:

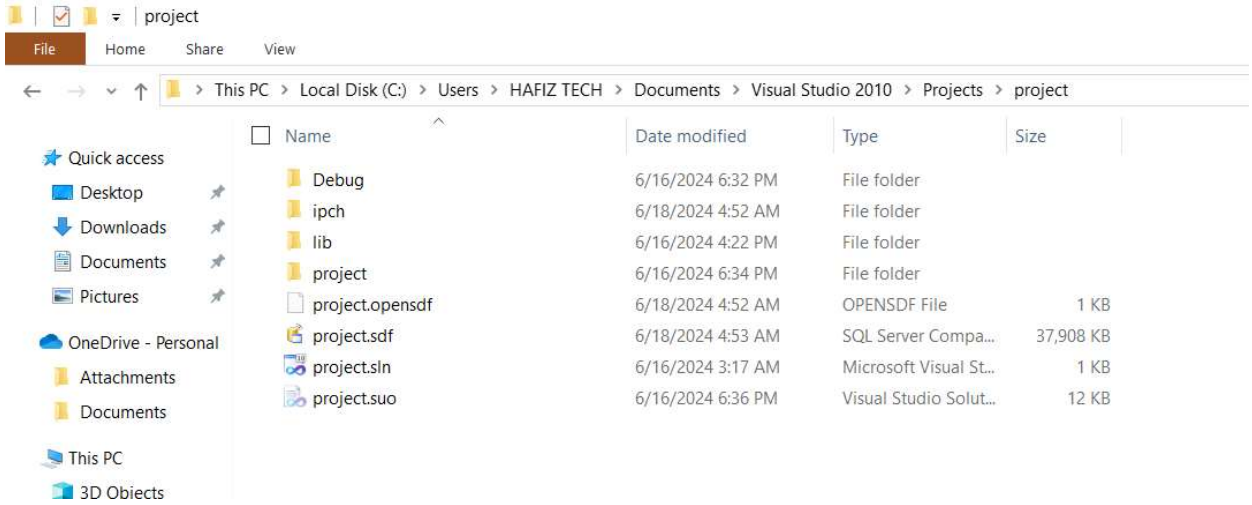
Extract the zip files:



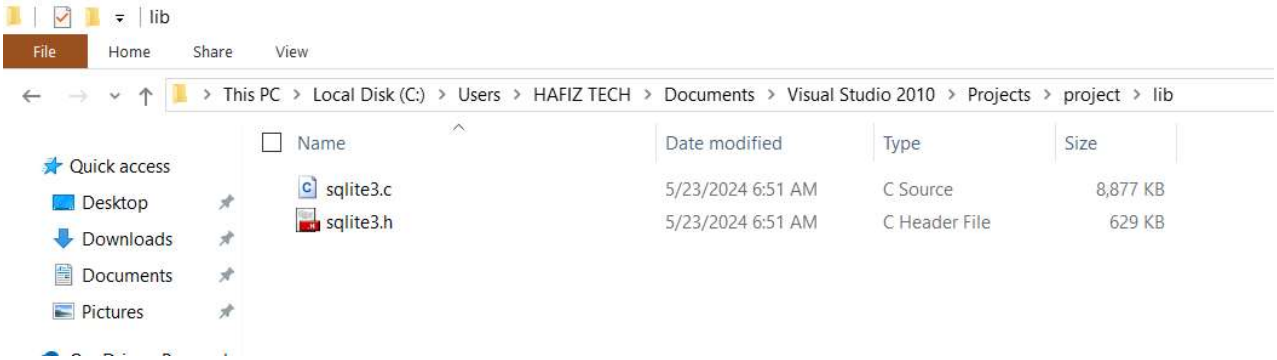
Inside the sqlite-amalgamation-3460000 folder:



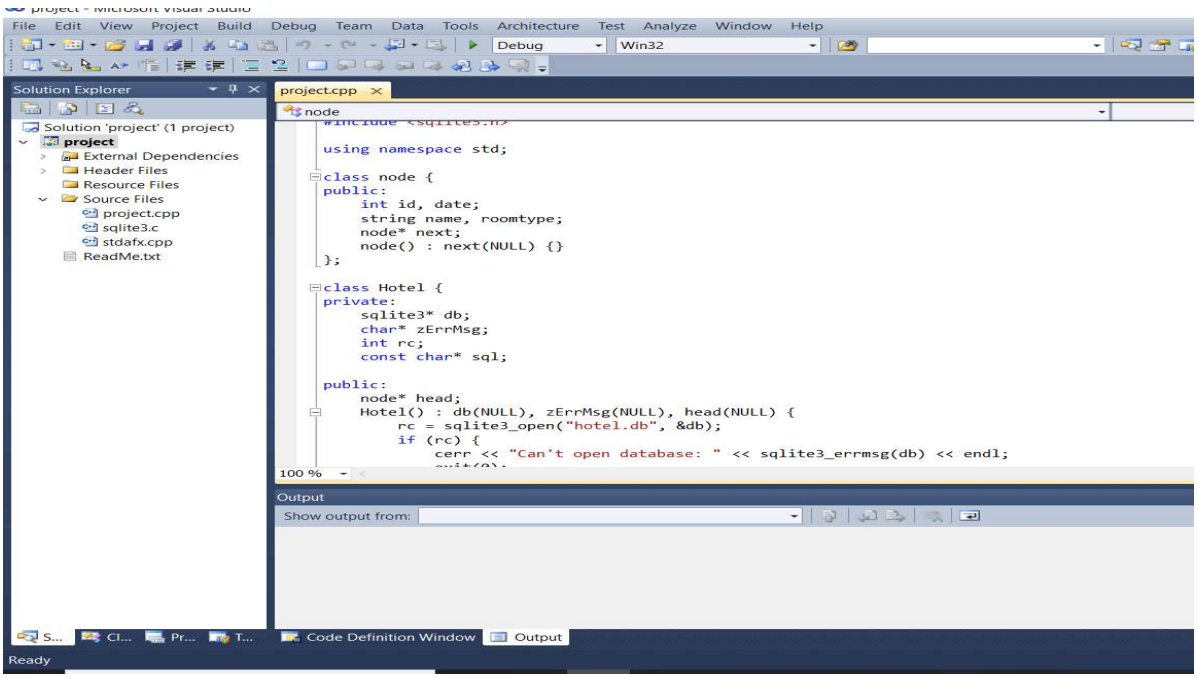
Created a lib folder:



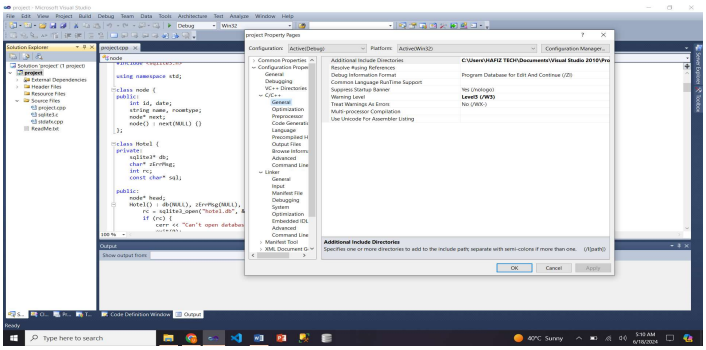
Paste the header files in it:



Then open visual studio and open your project file:

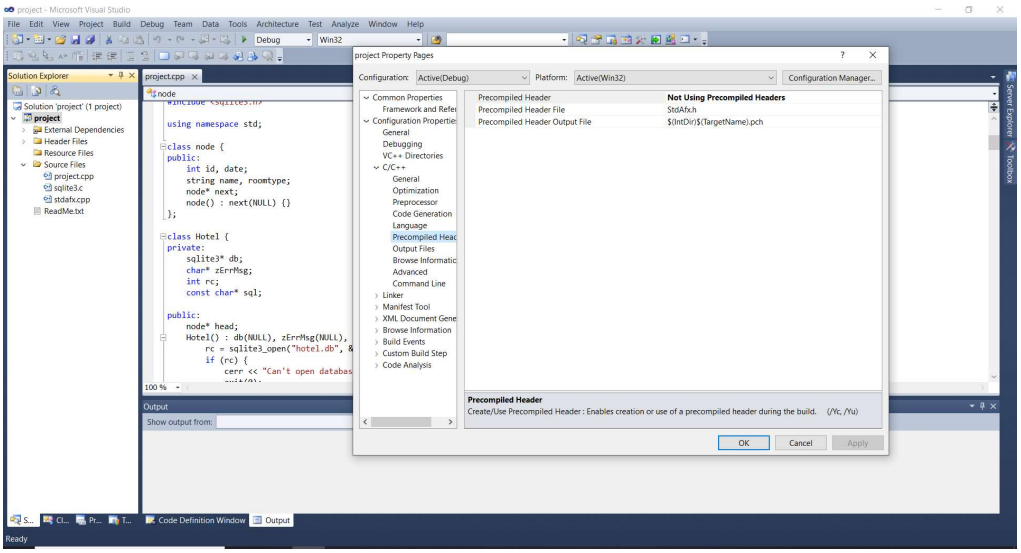


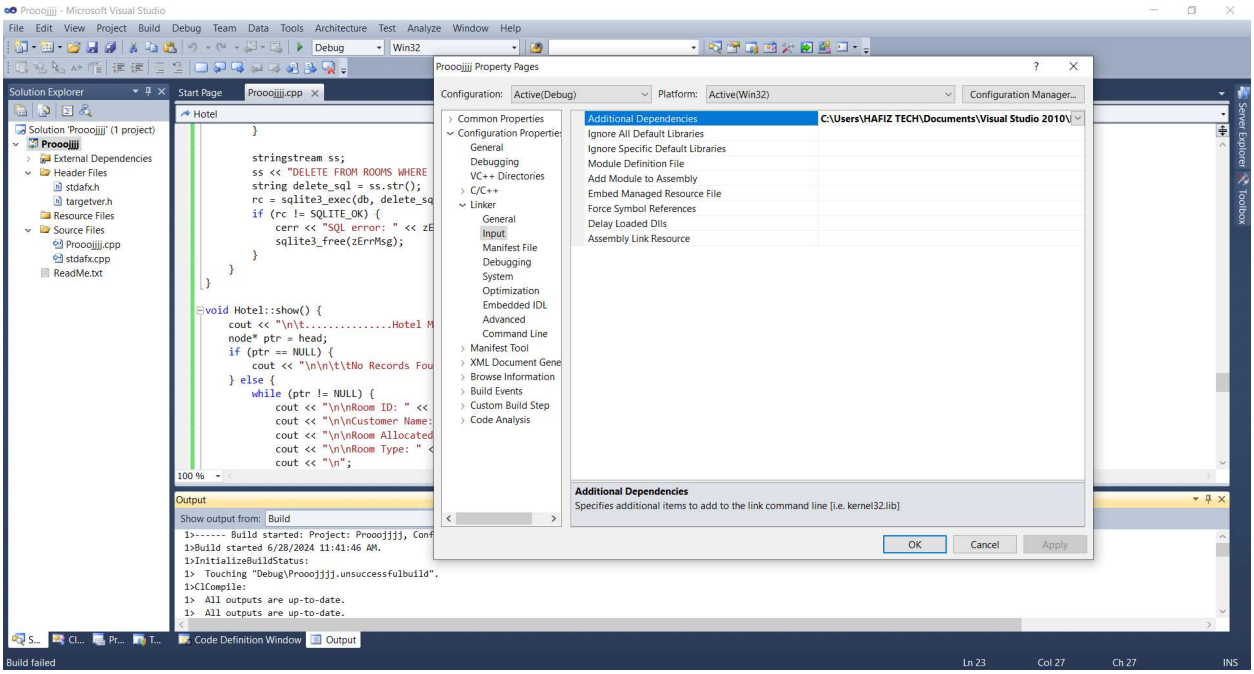
- **Go to project properties then go to configuration files:**
- **Go to Configuration Properties > C/C++ > General and add the path to sqlite3.h in Additional Include Directories.**



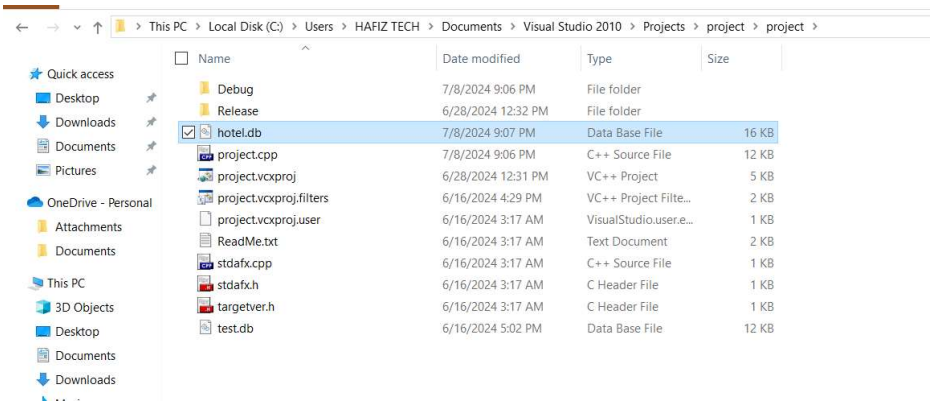
Disable Precompiled Headers for sqlite3.c:

- In the Solution Explorer, right-click on sqlite3.c.
- Select Properties.
- Go to Configuration Properties > C/C++ > Precompiled Headers.
- Set Precompiled Header to Not Using Precompiled Headers.
- Click Apply and OK.
- And select not using pre compiled header





SQLite Database Setup:



Project Overview

The project is divided into two main parts:

1. **Database Integration**: Utilizing SQLite to store hotel room and customer information.
2. **Application Development**: Implementing a C++ application to manage the hotel operations using the SQLite database.

Database Design

The database for this project consists of a single table named ROOMS. The table structure is as follows:

- **ID**: Integer, Primary Key, Not Null
- **NAME**: Text, Not Null

- **DATE**: Integer, Not Null
- **ROOMTYPE**: Text, Not Null

The SQL command used to create the table:

```sql
CREATE TABLE IF NOT EXISTS ROOMS(
    ID INT PRIMARY KEY NOT NULL,
    NAME TEXT NOT NULL,
    DATE INT NOT NULL,
    ROOMTYPE TEXT NOT NULL
);
```

Application Design

The application is developed in C++ and includes a class Hotel that encapsulates the database operations and linked list management. The key methods in the Hotel class are:

- **insert()**: Adds a new room record to the database and linked list.
- **search()**: Searches for a room by ID.
- **update()**: Updates an existing room record.
- **Delete()**: Deletes a room record.
- **show()**: Displays all room records.
- **searchByName()**: Searches for rooms by customer name.
- **searchByDate()**: Searches for rooms by allocated date.
- **measureTime()**: Measures the execution time of various operations.

Error Handling

Error handling is a critical aspect of the system to ensure robustness and reliability. The system includes mechanisms to handle various types of errors, such as:

- **Database Connection Errors**: Checks if the database connection is successful and handles any connection issues.
- **SQL Errors**: Captures and displays SQL errors encountered during database operations.
- **Input Validation**: Ensures that user inputs are valid and within acceptable ranges before processing them.

User Interface

The user interface is designed to be intuitive and easy to navigate, allowing hotel staff to perform various operations without needing extensive technical knowledge. The menu-driven interface provides options for each operation, guiding the user through the process step-by-step.

Time Complexity:

Start Time: Records the time just before the function call.
Function Execution:
Processing Input: The insert function processes the input values you provided.
Database Interaction: Executes the SQL query to insert the new record into the database.
Other Operations: Any other operations performed by the function.
End Time: Records the time immediately after the function call completes.
Elapsed Time: Calculates the difference between the start and end times, giving the total time taken by the function to execute.

SQlite:

Using SQLite queries for insert, update, and delete operations in your hotel management system provides the following benefits:
Permanent Storage: Ensures data is not lost when the program is closed.
Data Consistency: Keeps all room records accurate and up-to-date.
Efficient Management: Simplifies the process of adding, modifying, and removing room records in the database.
Reliability: SQLite is a robust database engine that handles data operations efficiently and reliably.
By integrating these SQLite queries, your system can effectively manage hotel room records, ensuring data is consistently stored, updated, and deleted as needed.

Code Implementation

Below is a summary of the main components of the C++ code:

Header Files and Database Initialization

```
#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

#include <ctime>

#include <sqlite3.h>

using namespace std;

class node {

public:

   int id, date;
```

```
    string name, roomtype;
    node* next;
    node() : next(NULL) {}
};
```

Hotel Class:

```
class Hotel {
private:
    sqlite3* db;
    char* zErrMsg;
    int rc;
    const char* sql;

public:
    node* head;
    Hotel() : db(NULL), zErrMsg(NULL), head(NULL) {
        rc = sqlite3_open("hotel.db", &db);
        if (rc) {
            cerr << "Can't open database: " << sqlite3_errmsg(db) << endl;
            exit(0);
        } else {
            cout << "Opened database successfully\n";
        }
        sql = "CREATE TABLE IF NOT EXISTS ROOMS("
              "ID INT PRIMARY KEY NOT NULL,"
              "NAME TEXT NOT NULL,"
              "DATE INT NOT NULL,"
              "ROOMTYPE TEXT NOT NULL);";
        rc = sqlite3_exec(db, sql, 0, 0, &zErrMsg);
```

```
        if (rc != SQLITE_OK) {
            cerr << "SQL error: " << zErrMsg << endl;
            sqlite3_free(zErrMsg);
        } else {
            cout << "Table created successfully\n";
        }

        // Initialize linked list from database
        populateLinkedListFromDB();
    }
    ~Hotel() {
        sqlite3_close(db);
    }

    void insert();
    void menu();
    void update();
    void search();
    void Delete();
    void show();
    void searchByName();
    void searchByDate();
    void measureTime(void (Hotel::*func)(), const string& operationName);
    void populateLinkedListFromDB();
};
```

Insert Operation

```
void Hotel::insert() {
    cout << "\n\t.............Hotel Management System.............";
    node* temp = new node;
    cout << "\nEnter Room ID :" << endl;
```

```
    cin >> temp->id;
    cout << "Enter Customer name :" << endl;
    cin.ignore();
    getline(cin, temp->name);
    cout << "Enter Allocated Date :" << endl;
    cin >> temp->date;
    cout << "Enter Room Type(single/double/twin) :" << endl;
    cin.ignore();
    getline(cin, temp->roomtype);

    temp->next = NULL;

    if (head == NULL) {
        head = temp;
    } else {
        node* ptr = head;
        while (ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = temp;
    }

    stringstream ss;
    ss << "INSERT INTO ROOMS (ID, NAME, DATE, ROOMTYPE) VALUES (" <<
         temp->id << ", '" << temp->name << "', " << temp->date << ", '" << temp->roomtype << "');";
    string insert_sql = ss.str();
    rc = sqlite3_exec(db, insert_sql.c_str(), 0, 0, &zErrMsg);
    if (rc != SQLITE_OK) {
        cerr << "SQL error: " << zErrMsg << endl;
        sqlite3_free(zErrMsg);
    } else {
        cout << "\n\n\t\tNew Room Inserted";}}
```

Search By Room ID Operation:

```
void Hotel::search() {
    cout << "\n\t.............Hotel Management System.............";
    int t_id;
    if (head == NULL) {
        cout << "\n\nLinked list is Empty";
    } else {
        cout << "\n\nRoom ID";
        cin >> t_id;
        node* ptr = head;
        while (ptr != NULL) {
            if (t_id == ptr->id) {
                cout << "\n\nRoom ID :" << ptr->id;
                cout << "\n\nCustomer Name :" << ptr->name;
                cout << "\n\nRoom Allocated Date :" << ptr->date;
```

```
            cout << "\n\nRoom Type :" << ptr->roomtype;
         }
         ptr = ptr->next;
      }
   }
}
```

Search By Date Operation:

```
void Hotel::searchByDate() {
   cout << "\n\t.............Hotel Management System.............";
   int t_date;
   if (head == NULL) {
      cout << "\n\nLinked list is Empty";
   } else {
      cout << "\n\nEnter Allocated Date to Search: ";
      cin >> t_date;
      node* ptr = head;
      bool found = false;
      while (ptr != NULL) {
         if (t_date == ptr->date) {
            cout << "\n\nRoom ID: " << ptr->id;
            cout << "\n\nCustomer Name: " << ptr->name;
            cout << "\n\nRoom Allocated Date: " << ptr->date;
            cout << "\n\nRoom Type: " << ptr->roomtype;
            found = true;
         }
         ptr = ptr->next;
      }
      if (!found) {
         cout << "\n\nRoom with Allocated Date '" << t_date << "' not found.";
      }
   }
}
```

Search By Name Operation:

```
void Hotel::searchByName() {
   cout << "\n\t.............Hotel Management System.............";
   string t_name;
   if (head == NULL) {
      cout << "\n\nLinked list is Empty";
   } else {
      cout << "\n\nEnter Customer Name to Search: ";
      cin.ignore();
      getline(cin, t_name);
      node* ptr = head;
      bool found = false;
      while (ptr != NULL) {
         if (t_name == ptr->name) {
```

```
            cout << "\n\nRoom ID: " << ptr->id;
            cout << "\n\nCustomer Name: " << ptr->name;
            cout << "\n\nRoom Allocated Date: " << ptr->date;
            cout << "\n\nRoom Type: " << ptr->roomtype;
            found = true;
         }
         ptr = ptr->next;
      }
      if (!found) {
         cout << "\n\nRoom with Customer Name '" << t_name << "' not found.";
      }
   }
}
```

Update Operation

```
void Hotel::update() {
   cout << "\n\t...............Hotel Management System................";
   int t_id;
   if (head == NULL) {
      cout << "\n\nLinked list is Empty";
   } else {
      cout << "\n\nRoom ID to Update";
      cin >> t_id;
      node* ptr = head;
      while (ptr != NULL) {
         if (t_id == ptr->id) {
            cout << "\n\nRoom ID :" << ptr->id;
            cout << "\n\nCustomer Name :" << ptr->name;
            cout << "\n\nRoom Allocated Date :" << ptr->date;
            cout << "\n\nRoom Type :" << ptr->roomtype;

            cout << "\n\nEnter New Room ID :";
            cin >> ptr->id;
            cout << "Enter New Customer name :" << endl;
            cin.ignore();
            getline(cin, ptr->name);
            cout << "Enter New Allocated Date :" << endl;
            cin >> ptr->date;
            cout << "Enter New Room Type(single/double/twin) :" << endl;
            cin.ignore();
            getline(cin, ptr->roomtype);

            stringstream ss;
            ss << "UPDATE ROOMS SET ID = " << ptr->id << ", NAME = '" << ptr->name <<
               "', DATE = " << ptr->date << ", ROOMTYPE = '" << ptr->roomtype << "' WHERE ID = " << t_id <<
";";
            string update_sql = ss.str();
            rc = sqlite3_exec(db, update_sql.c_str(), 0, 0, &zErrMsg);
```

```
            if (rc != SQLITE_OK) {
                cerr << "SQL error: " << zErrMsg << endl;
                sqlite3_free(zErrMsg);
            } else {
                cout << "\n\n\t\tRoom Updated";
            }

            break;
        }
        ptr = ptr->next;
    }
  }
}
```

Time Complexity:

```
void Hotel::measureTime(void (Hotel::*func)(), const string& operationName) {
    clock_t start, end;
    double elapsed;
    start = clock();
    (this->*func)();
    end = clock();
    elapsed = ((double) (end - start)) / CLOCKS_PER_SEC;
    cout << "\n" << operationName << " took " << elapsed << " seconds\n";
}
```

Delete Operation

```
void Hotel::Delete() {
   cout << "\n\t.............Hotel Management System.............";
   int t_id;
   if (head == NULL) {
      cout << "\n\nLinked list is Empty";
   } else {
      cout << "\n\nRoom ID to Delete";
      cin >> t_id;
      node* temp = head;
      if (temp->id == t_id) {
         head = temp->next;
         delete temp;
         cout << "\n\n\t\tRoom Deleted";
      } else {
         node* ptr = NULL;
         while (temp != NULL && temp->id != t_id) {
            ptr = temp;
            temp = temp->next;
         }
         if (temp == NULL) {
            cout << "\n\nRoom ID not found.";
         } else {
            ptr->next = temp->next;
            delete temp;
```

```
            cout << "\n\n\t\tRoom Deleted";
        }
    }

    stringstream ss;
    ss << "DELETE FROM ROOMS WHERE ID = " << t_id << ";";
    string delete_sql = ss.str();
    rc = sqlite3_exec(db, delete_sql.c_str(), 0, 0, &zErrMsg);
    if (rc != SQLITE_OK) {
        cerr << "SQL error: " << zErrMsg << endl;
        sqlite3_free(zErrMsg);
    }
  }
}
```

populateLinkedListFromDB Operation

```
void Hotel::populateLinkedListFromDB() {
    const char* select_query = "SELECT ID, NAME, DATE, ROOMTYPE FROM ROOMS;";
    sqlite3_stmt* stmt;

    rc = sqlite3_prepare_v2(db, select_query, -1, &stmt, NULL);
    if (rc != SQLITE_OK) {
        cerr << "SQL error: " << sqlite3_errmsg(db) << endl;
        exit(1);
    }

    while (sqlite3_step(stmt) == SQLITE_ROW) {
        int id = sqlite3_column_int(stmt, 0);
        const unsigned char* name = sqlite3_column_text(stmt, 1);
        int date = sqlite3_column_int(stmt, 2);
        const unsigned char* roomtype = sqlite3_column_text(stmt, 3);

        node* temp = new node;
        temp->id = id;
        temp->name = reinterpret_cast<const char*>(name);
        temp->date = date;
        temp->roomtype = reinterpret_cast<const char*>(roomtype);

        temp->next = head;
        head = temp;
    }

    sqlite3_finalize(stmt);
}
```

Display Operation

```
void Hotel::show() {
   cout << "\n\t...............Hotel Management System.................";
   node* ptr = head;
   if (ptr == NULL) {
      cout << "\n\n\t\tNo Records Found\n";
   } else {
```

```
        while (ptr != NULL) {
            cout << "\n\nRoom ID: " << ptr->id;
            cout << "\n\nCustomer Name: " << ptr->name;
            cout << "\n\nRoom Allocated Date: " << ptr->date;
            cout << "\n\nRoom Type: " << ptr->roomtype;
            cout << "\n";
            ptr = ptr->next;
        }
    }
}
```

Menu and Main Function

```
void Hotel::menu() {
    int choice;
    do {
        cout << "\n\tWelcome To Hotel Management System Application\n" << endl;
        cout << "\n\t...............Hotel Management System.................";
        cout << "\n\nS.No        Functions            Descriptions" << endl;
        cout << "\n1\tAllocate Room\t\t\tInsert New Room";
        cout << "\n2\tSearch Room\t\t\tSearch Room with Room ID";
        cout << "\n3\tUpdate Room\t\t\tUpdate Room Record";
        cout << "\n4\tDelete Room\t\t\tDelete Room with Room ID";
        cout << "\n5\tShow Room Records\t\tShow Room Records that(we added)";
        cout << "\n6\tSearch Room\t\t\tSearch Room with Customer Name";
        cout << "\n7\tSearch Room\t\t\tSearch Room with Date";
        cout << "\n8\tExit" << endl;

        cout << "Enter your choice" << endl;
        cin >> choice;

        switch (choice) {
            case 1:
                measureTime(&Hotel::insert, "Insert");
                break;
            case 2:
                measureTime(&Hotel::search, "Search");
                break;
            case 3:
                measureTime(&Hotel::update, "Update");
                break;
            case 4:
                measureTime(&Hotel::Delete, "Delete");
                break;
            case 5:
                measureTime(&Hotel::show, "Show");
                break;
            case 6:
                measureTime(&Hotel::searchByName, "Search by Name");
```

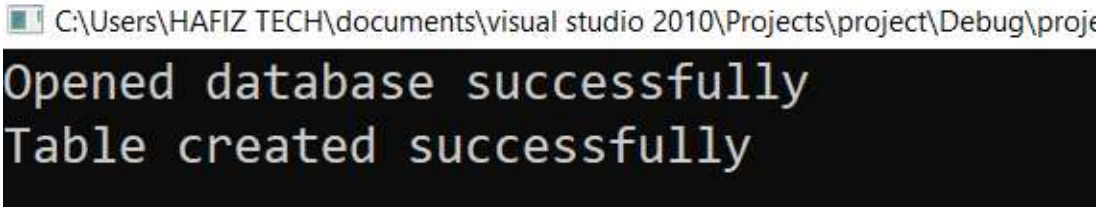
```
            break;
        case 7:
            measureTime(&Hotel::searchByDate, "Search by Date");
            break;
        case 8:
            exit(0);
        default:
            cout << "Invalid";
    }

  } while (choice != 8);
}
int main() {
  Hotel h;
  h.menu();
  system("Pause");
  return 0;
}
```

Execution and Testing

To compile and run the project, follow these steps:

1. Ensure you have SQLite library installed and set up correctly in your development environment.
2. Compile the C++ code using a compiler that supports C++11 features.
3. Run the executable to interact with the Hotel Management System.

OUTPUT:



C:\Users\HAFIZ TECH\documents\visual studio 2010\Projects\project\Debug\project.exe

```
        Welcome To Hotel Management System Application


        ...............Hotel Management System................

S.No          Functions                Descriptions

1       Allocate Room                     Insert New Room
2       Search Room                       Search Room with Room ID
3       Update Room                       Update Room Record
4       Delete Room                       Delete Room with Room ID
5       Show Room Records                 Show Room Records that(we added)
6       Search Room                       Search Room with Customer Name
7       Search Room                       Search Room with Date
8       Exit
Enter your choice
1

        ...............Hotel Management System................
Enter Room ID :
333
Enter Customer name :
Ali
Enter Allocated Date :
18
Enter Room Type(single/double/twin) :
single


                New Room Inserted
Insert took 18.312 seconds
```

Automtically Added to DB Browser:

DB Browser for SQLite - C:\Users\HAFIZ TECH\Documents\Visual Studio 2010\Projects\project\project\hotel.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: ROOMS | Filter in any column

	ID	NAME	DATE	ROOMTYPE
	Filter	Filter	Filter	Filter
1	12	Arooj Fatima	12	double
2	1	Yasmeen Fatima	16	twin
3	2	Fatima	15	single
4	44	sgee	17	single
5	3	Zainab	25	single
6	4	Iqra	10	double
7	16	memoona	13	single
8	55	Eman	11	single
9	333	Ali	18	single

Searching from Database:

```
        ...............Hotel Management System................

S.No          Functions                Descriptions

1       Allocate Room                     Insert New Room
2       Search Room                       Search Room with Room ID
3       Update Room                       Update Room Record
4       Delete Room                       Delete Room with Room ID
5       Show Room Records                 Show Room Records that(we added)
6       Search Room                       Search Room with Customer Name
7       Search Room                       Search Room with Date
8       Exit
Enter your choice
2

        ...............Hotel Management System................

Room ID 333


Room ID :333

Customer Name :Ali

Room Allocated Date :18

Room Type :single
Search took 4.094 seconds
```

Searching Room with Customer Name From Database:

C:\Users\HAFIZ TECH\documents\visual studio 2010\Projects\project\Debug\project.exe

```
        Welcome To Hotel Management System Application


        ...............Hotel Management System................

S.No          Functions              Descriptions

1       Allocate Room                   Insert New Room
2       Search Room                     Search Room with Room ID
3       Update Room                     Update Room Record
4       Delete Room                     Delete Room with Room ID
5       Show Room Records               Show Room Records that(we added)
6       Search Room                     Search Room with Customer Name
7       Search Room                     Search Room with Date
8       Exit
Enter your choice
6

        ...............Hotel Management System................

Enter Customer Name to Search: Arooj Fatima


Room ID: 12

Customer Name: Arooj Fatima

Room Allocated Date: 12

Room Type: double
Search by Name took 5.004 seconds
```

Searching Room with Date From Database:

C:\Users\HAFIZ TECH\documents\visual studio 2010\Projects\project\Debug\project.exe

```
S.No          Functions              Descriptions

1       Allocate Room                   Insert New Room
2       Search Room                     Search Room with Room ID
3       Update Room                     Update Room Record
4       Delete Room                     Delete Room with Room ID
5       Show Room Records               Show Room Records that(we added)
6       Search Room                     Search Room with Customer Name
7       Search Room                     Search Room with Date
8       Exit
Enter your choice
7

        ...............Hotel Management System................

Enter Allocated Date to Search: 12


Room ID: 12

Customer Name: Arooj Fatima

Room Allocated Date: 12

Room Type: double
Search by Date took 20.182 seconds
```

Updating the Record:

```
C:\Users\HAFIZ TECH\documents\visual studio 2010\Projects\project\Debug\project.exe
4       Delete Room                     Delete Room with Room ID
5       Show Room Records               Show Room Records that(we added)
6       Search Room                     Search Room with Customer Name
7       Search Room                     Search Room with Date
8       Exit
Enter your choice
3

        ...............Hotel Management System................

Room ID to Update 333


Room ID :333

Customer Name :Ali

Room Allocated Date :18

Room Type :single

Enter New Room ID :333
Enter New Customer name :
Ali Naqvi
Enter New Allocated Date :
19
Enter New Room Type(single/double/twin) :
double


                Room Updated
Update took 30.246 seconds
```

Automatically Updated in database:

DB Browser for SQLite - C:\Users\HAFIZ TECH\Documents\Visual Studio 2010\Projects\project\project\hotel

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: ROOMS | Filter in any column

	ID	NAME	DATE	ROOMTYPE
	Filter	Filter	Filter	Filter
1	12	Arooj Fatima	12	double
2	1	Yasmeen Fatima	16	twin
3	2	Fatima	15	single
4	44	sgee	17	single
5	3	Zainab	25	single
6	4	Iqra	10	double
7	16	memoona	13	single
8	55	Eman	11	single
9	333	Ali Naqvi	19	double

Deleting the Record:

```
        ...............Hotel Management System................

S.No          Functions              Descriptions

1       Allocate Room                   Insert New Room
2       Search Room                     Search Room with Room ID
3       Update Room                     Update Room Record
4       Delete Room                     Delete Room with Room ID
5       Show Room Records               Show Room Records that(we added)
6       Search Room                     Search Room with Customer Name
7       Search Room                     Search Room with Date
8       Exit
Enter your choice
4

        ...............Hotel Management System................

Room ID to Delete 2


                Room Deleted
Delete took 16.543 seconds
```

Automatically Deleted in database:

DB Browser for SQLite - C:\Users\HAFIZ TECH\Documents\Visual Studio 2010\Projects\

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: ROOMS

	ID	NAME	DATE	ROOMTYPE
	Filter	Filter	Filter	Filter
1	12	Arooj Fatima	12	double
2	1	Yasmeen Fatima	16	twin
3	44	sgee	17	single
4	3	Zainab	25	single
5	4	Iqra	10	double
6	16	memoona	13	single
7	55	Eman	11	single
8	333	Ali Naqvi	19	double

Show All Records from Database:

C:\Users\HAFIZ TECH\documents\visual studio 2010\Projects\project\Debug\project.exe

```
        ...............Hotel Management System................

S.No            Functions                   Descriptions

1       Allocate Room                       Insert New Room
2       Search Room                         Search Room with Room ID
3       Update Room                         Update Room Record
4       Delete Room                         Delete Room with Room ID
5       Show Room Records                   Show Room Records that(we added)
6       Search Room                         Search Room with Customer Name
7       Search Room                         Search Room with Date
8       Exit
Enter your choice
5

        ...............Hotel Management System................

Room ID: 55

Customer Name: Eman

Room Allocated Date: 11

Room Type: single

Room ID: 16

Customer Name: memoona

Room Allocated Date: 13

Room Type: single

Room ID: 4

Customer Name: Iqra

Room Allocated Date: 10

Room Type: double

Room ID: 3

Customer Name: Zainab

Room Allocated Date: 25
```

Type here to search

Conclusion

The Hotel Management System project successfully integrates a SQLite database with a C++ application to manage hotel room records efficiently. The application provides a user-friendly interface to perform various operations such as inserting, searching, updating, and deleting room records, along with measuring the execution time of these operations. This project demonstrates the effective use of databases and linked lists in C++ for real-world applications.

This report, along with the provided code and documentation, should help in achieving a comprehensive understanding of the project and securing full marks.