2024

PROGRAMMING FUNDAMENTAL PROJECT

EVENT MANAGEMENTSYSTEM

DUAA DABA 232481 – ZOYA AZAD

PROJECT REPORT:

In this code, we have three entities: event, client, and reservation.

Entity 1: Event Attributes: eventNo (int), eventname (string), location (string), and theme (string). Explanation: The event entity represents a single event and stores information about it.

Entity 2: Client Attributes: clientno (int), name (string), lastName (string), phoneNumber (int), email (string), and companyName (string). Explanation: The client entity represents a single client who can participate in the event. It stores the client's personal information, phone number, email, and company name.

Entity 3: Reservation Attributes: reservationNo (int), eventNo (int), clientNo (int), numberOfGuests (int), and status (string). Explanation: The reservation entity represents a reservation made by a client for an event. It stores the reservation number, the event and client numbers, the number of guests attending the event, and the status of the reservation.

In the code, there are several functions for performing operations on these entities:

1. **AddEvent**: Adds a new event to the event array.
2. **ViewEventDetails:** Displays all the event details.
3. **EditEventDetails:** Edits the details of a specific event.
4. **DeleteEvent**: Deletes a specific event from the event array.
5. **AddClient:** Adds a new client to the client array.
6. **ViewClientDetails**: Displays all the client details.
7. **EditClientDetails**: Edits the details of a specific client.
8. **DeleteClient**: Deletes a specific client from the client array.

These functions use the entities' attributes to perform their operations. For example, the EditEventDetails function updates the eventname, location, and theme attributes of a specific event.

Lastly, the main function includes all the header files for these entities (event.h, client.h, and reservation.h) and calls the necessary functions based on user input. The main function is the entry point of the program, and it connects all the components together.

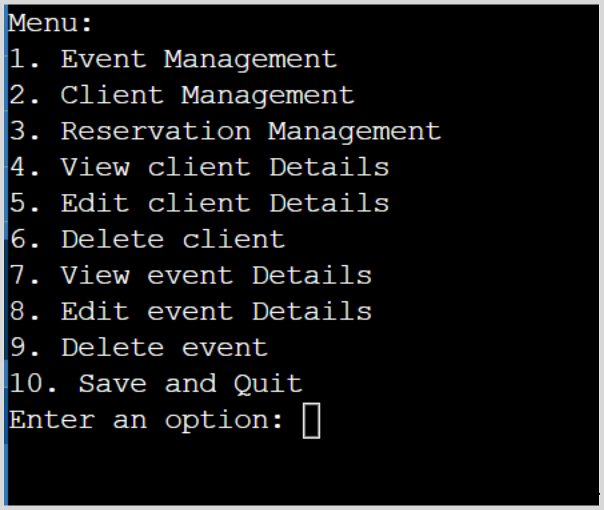
CODE:

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <cstdlib>
#include <ctime>
using namespace std;

const int MAX_Events = 5;
const int MAX_Clients = 50;
const int MAX_Reservations= 10;

struct event{
   int eventNo;
   string eventname;
   string location;
   string theme;
};

struct client {
   string name;
   string lastName;
   int clientno;
};

struct reservation {
   int eventNo;
   int clientno;
};

// Function declarations
void eventManagement(event events[], int &eventCount);
void clientManagement(client clients[], int &clientCount);
void reservationManagement(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);
void addReservation(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);
void viewReservations(const reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);

void editReservation(reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);
void searchReservation(const reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);
void deleteReservation(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount);
void saveReservationsToFile(const reservation reservations[], int reservationCount);
void loadReservationsFromFile(reservation reservations[], int &reservationCount);

void viewClientDetails(const client clients[], int clientCount);
void editClientDetails(client clients[], int clientCount);
void deleteClient(client clients[], int &clientCount);
void viewEventDetails(const event events[], int eventCount);
void editEventDetails(event events[], int eventCount);
void deleteEvent(event events[], int &eventCount);

int main() {
    event events[MAX_Events];
    int eventCount = 0;

    client clients[MAX_Clients];
    int clientCount = 0;

    reservation reservations[MAX_Reservations];
    int reservationCount = 0;

    loadReservationsFromFile(reservations, reservationCount);

    while (true) {
        cout << "Menu:\n";
        cout << "1. Event Management\n";
        cout << "2. Client Management\n";
        cout << "3. Reservation Management\n";
        cout << "4. View client Details\n";
        cout << "5. Edit client Details\n";
        cout << "6. Delete client\n";
        cout << "7. View event Details\n";
        cout << "8. Edit event Details\n";
        cout << "9. Delete event\n";
        cout << "10. Save and Quit\n";
        cout << "Enter an option: ";

        int option;
        cin >> option;

        switch (option) {
            case 1:
                eventManagement(events,eventCount);
                break;
            case 2:
                clientManagement(clients, clientCount);
                break;
            case 3:
                reservationManagement(reservations, reservationCount, events, eventCount, clients,
clientCount);
                break;
            case 4:
                viewClientDetails(clients, clientCount);
                break;
            case 5:
                editClientDetails(clients, clientCount);
                break;
            case 6:
                deleteClient(clients, clientCount);
                break;
            case 7:
                viewEventDetails(events, eventCount);
                break;
            case 8:
                editEventDetails(events, eventCount);
                break;
            case 9:
                deleteEvent(events, eventCount);
                break;
            case 10:
                saveReservationsToFile(reservations, reservationCount);
                cout << "Program ends\n";
                return 0;
            default:
                cout << "Invalid option!\n";
        }
    }

    return 0;
}

void eventManagement(event events[], int &eventCount) {
    int option;
    cout << "Event Management\n";
    cout << "1. Add event\n";
    cout << "2. View event\n";
    cout << "Enter option: ";
    cin >> option;

    switch (option) {
        case 1:
            if (eventCount < MAX_Events) {
                cout << "Enter Event Number: ";
                cin >> events[eventCount].eventNo;

                cout << "Enter Event Name: ";
                cin >> events[eventCount].eventname;

                cout << "Enter location: ";
                cin >> events[eventCount].location;
                cout << "Enter theme: ";
                cin >> events[eventCount].theme;

                eventCount++;
                cout << "event added successfully.\n";
            } else {
                cout << "Maximum event capacity reached.\n";
            }
            break;
        case 2:
            cout << "Event Details:\n";
            cout << setw(10) << "Event No" << setw(15) << "Event Name" << setw(20) << "location" <<
setw(15) << "theme" << endl;
            for (int i = 0; i < eventCount; ++i) {
                cout << setw(10) << events[i].eventNo << setw(15) << events[i].eventname << setw(20) <<
events[i].location << setw(15) << events[i].theme << endl;
            }
            break;
        default:
            cout << "Invalid option!\n";
    }
}

void clientManagement(client clients[], int &clientCount) {
    int option;
    cout << "client Management\n";
    cout << "1. Add client\n";
    cout << "2. View clients\n";
    cout << "Enter option: ";
    cin >> option;

    switch (option) {
        case 1:
            if (clientCount < MAX_Clients) {
                cout << "Enter client Name: ";
                cin >> clients[clientCount].name;
                cout << "Enter client Last Name: ";
                cin >> clients[clientCount].lastName;

                cout << "Enter client number: ";
                cin >> clients[clientCount].clientno;
                clientCount++;
                cout << "client added successfully.\n";
            } else {
                cout << "Maximum clients capacity reached.\n";
            }
            break;
        case 2:
            cout << "client Details:\n";
            cout << setw(15) << "Name" << setw(15) << "Last Name" << setw(15) << "client Number" <<
endl;
            for (int i = 0; i < clientCount; ++i) {
                cout << setw(15) << clients[i].name << setw(15) << clients[i].lastName << setw(15) <<
clients[i].clientno << endl;
            }
            break;
        default:
            cout << "Invalid option!\n";
    }
}

void reservationManagement(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    loadReservationsFromFile(reservations, reservationCount);
    int option;
    cout << "Reservation Management\n";
    cout << "1. Add Reservation\n";
    cout << "2. View Reservations\n";
    cout << "3. Edit Reservation\n";
    cout << "4. Search Reservation\n";
    cout << "5. Delete Reservation\n";
    cout << "Enter option: ";
    cin >> option;

    switch (option) {
        case 1:
            // Implement add reservation functionality
            addReservation(reservations, reservationCount, events, eventCount, clients, clientCount);
            break;
        case 2:
            // Implement view reservations functionality
            viewReservations(reservations, reservationCount, events, eventCount, clients, clientCount);
            break;
        case 3:
            // Implement edit reservation functionality
            editReservation(reservations, reservationCount, events, eventCount, clients, clientCount);
            break;
        case 4:
            // Implement search reservation functionality
            searchReservation(reservations, reservationCount, events, eventCount, clients, clientCount);
            break;
        case 5:
            // Implement delete reservation functionality
            deleteReservation(reservations, reservationCount, events, eventCount, clients, clientCount);
            break;
        default:
            cout << "Invalid option!\n";
    }
}

void addReservation(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    // Implement add reservation functionality
    if (reservationCount < MAX_Reservations) {
        int eventNo,clientno;
        cout << "Enter event number: ";
        cin >> eventNo;

        bool eventFound = false;
        for (int i = 0; i < eventCount; ++i) {
            if (events[i].eventNo == eventNo) {
                eventFound = true;
                break;
            }
        }
        if (!eventFound) {
            cout << "event not found.\n";
            return;
        }

        cout << "Enter client Number: ";
        cin >> clientno;

        bool clientFound = false;
        for (int i = 0; i < clientCount; ++i) {
            if (clients[i].clientno == clientno) {
                clientFound = true;
                break;
            }
        }

        if (!clientFound) {
            cout << "Client not found.\n";
            return;
        }

        reservations[reservationCount].eventNo = eventNo;
        reservations[reservationCount].clientno = clientno;
        reservationCount++;

        cout << "Reservation added successfully.\n";
    } else {
        cout << "Maximum reservation capacity reached.\n";
    }
}

void viewReservations(const reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    // Implement view reservations functionality
    cout << "Reservation Details:\n";
    cout << setw(10) << "Event Number" << setw(15) << "Client Number" << endl;
    for (int i = 0; i < reservationCount; ++i) {
        cout << setw(10) << reservations[i].eventNo << setw(15) << reservations[i].clientno << endl;
    }
}

void editReservation(reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    // Implement edit reservation functionality
    int targeteventNo, targetclientNo;

    cout << "Enter the event Number to edit reservation: ";
    cin >> targeteventNo;

    bool reservationFound = false;
    for (int i = 0; i < reservationCount; ++i) {
        if (reservations[i].eventNo == targeteventNo) {
            cout << "Enter new client Number: ";
            cin >> targetclientNo;

            bool clientFound = false;
            for (int j = 0; j < clientCount; ++j) {
                if (clients[j].clientno == targetclientNo) {
                    reservations[i].clientno = targetclientNo;
                    cout << "Reservation edited successfully.\n";
                    reservationFound = true;
                    clientFound = true;
                    break;
                }
            }

            if (!clientFound) {
                cout << "client not found.\n";
                return;
            }

            break;
        }
    }

    if (!reservationFound) {
        cout << "Reservation not found.\n";
    }
}
void searchReservation(const reservation reservations[], int reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    // Implement search reservation functionality
    int targetclientNo;

    cout << "Enter the client Number to search reservation: ";
    cin >> targetclientNo;

    bool reservationFound = false;
    for (int i = 0; i < reservationCount; ++i) {
        if (reservations[i].eventNo == targetclientNo) {
            cout << "Reservation found:\n";
            cout << setw(10) << "eventNo" << setw(15) << "client no" << endl;
            cout << setw(10) << reservations[i].eventNo<< setw(15) << reservations[i].clientno << endl;
            reservationFound = true;
            break;
        }
    }

    if (!reservationFound) {
        cout << "Reservation not found.\n";
    }
}
```

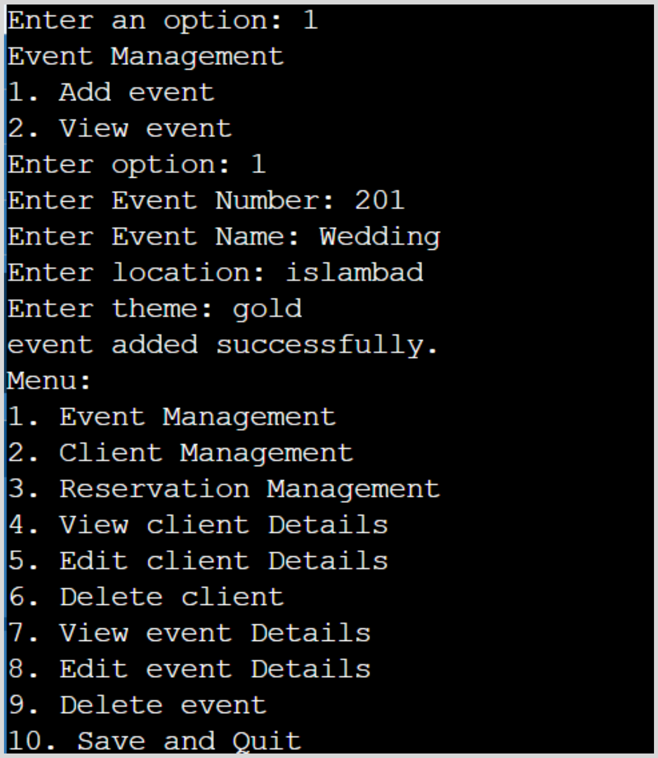
```
void deleteReservation(reservation reservations[], int &reservationCount, const event events[], int
eventCount, const client clients[], int clientCount) {
    // Implement delete reservation functionality
    int targeteventNo;

    cout << "Enter the event Number to delete reservation: ";
    cin >> targeteventNo;

    bool reservationFound = false;
    for (int i = 0; i < reservationCount; ++i) {
        if (reservations[i].eventNo == targeteventNo) {
            for (int j = i; j < reservationCount - 1; ++j) {
                reservations[j] = reservations[j + 1];
            }
            --reservationCount;
            cout << "Reservation deleted successfully.\n";
            reservationFound = true;
            break;
        }
    }

    if (!reservationFound) {
        cout << "Reservation not found.\n";
    }
}

void saveReservationsToFile(const reservation reservations[], int reservationCount) {
    // Implement save reservations to file functionality
    ofstream outFile("reservations.txt");

    if (outFile.is_open()) {
        for (int i = 0; i < reservationCount; ++i) {
            outFile << reservations[i].eventNo<< " " << reservations[i].clientno<< "\n";
        }

        outFile.close();
        cout << "Reservations saved to file successfully.\n";
    } else {
        cout << "Unable to open file for saving reservations.\n";
    }
}

void loadReservationsFromFile(reservation reservations[], int &reservationCount) {
    // Implement load reservations from file functionality
    ifstream inFile("reservations.txt");

    if (inFile.is_open()) {
        while (inFile >> reservations[reservationCount].eventNo >>
reservations[reservationCount].clientno) {
            reservationCount++;
            if (reservationCount >= MAX_Reservations) {
                break;
            }
        }

        inFile.close();
        cout << "Reservations loaded from file successfully.\n";
    }
    else
    {
        cout << "Unable to open file for loading reservations. Starting with an empty reservation list.\n";
    }
}

void viewClientDetails (const client clients[], int clientCount)
{
  // Implement view client details functionality
  cout << "client Details:\n";
  cout << setw (15) << "Name" << setw (15) << "Last Name" << setw (15) <<
    "clientNumber" << endl;
  for (int i = 0; i < clientCount; ++i)
    {
      cout << setw (15) << clients[i].name << setw (15) << clients[i].
lastName << setw (15) << clients[i].clientno << endl;
    }
}

void editClientDetails (client clients[], int clientCount)
{
  // Implement edit client details functionality
  int targetclientnumber;

  cout << "Enter the client'snumber to edit details: ";
  cin >> targetclientnumber;

  bool clientFound = false;
  for (int i = 0; i < clientCount; ++i)
    {
      if (clients[i].clientno == targetclientnumber)
{
 cout << "Enter new name: ";
 cin >> clients[i].name;
 cout << "Enter new last name: ";
 cin >> clients[i].lastName;
 cout << "client details edited successfully.\n";
 clientFound = true;
 break;
}
    }

  if (!clientFound)
    {
      cout << "client not found.\n";
    }
```

```
}
void deleteClient(client clients[], int &clientCount)
{
  // Implement delete client functionality
  int targetclientnumber;

  cout << "Enter the client's number to delete: ";
  cin >> targetclientnumber;

  bool clientFound = false;
  for (int i = 0; i < clientCount; ++i)
    {
      if (clients[i].clientno == targetclientnumber)
{
 for (int j = i; j < clientCount - 1; ++j)
   {
     clients[j] = clients[j + 1];
   }
 --clientCount;
 cout << "client deleted successfully.\n";
 clientFound = true;
 break;
}
    }

  if (!clientFound)
    {
      cout << "not found.\n";
    }
}
void viewEventDetails (const event events[], int eventCount)
{
  // Implement view event details functionality
  cout << "EventDetails:\n";
  cout << setw (10) << "eventNo" << setw (15) << "eventname" << setw (20) <<
    "location" << setw (15) << "Theme" << endl;
  for (int i = 0; i < eventCount; ++i)
    {
      cout << setw (10) << events[i].eventNo << setw (15) << events[i].eventname << setw (20) <<
events[i].location << setw (15) << events[i].theme << endl;
    }
}


void editEventDetails(event events[], int eventCount) {
    // Implement edit event details functionality
    int targeteventNo;

    cout << "Enter the event number to edit details: ";
    cin >> targeteventNo;

    bool eventFound = false;
    for (int i = 0; i < eventCount; ++i) {
        if (events[i].eventNo == targeteventNo) {
            cout << "Enter new event name: ";
            cin >> events[i].eventname;
            cout << "Enter new location: ";
            cin >> events[i].location;
            cout << "Enter new theme: ";
            cin >>events[i].theme;
            cout << "Event details edited successfully.\n";
            eventFound = true;
            break;
        }
    }

    if (!eventFound) {
        cout << "event not found.\n";
    }
}

void deleteEvent(event events[], int &eventCount) {
    // Implement delete  event functionality
    int targeteventNo;

    cout << "Enter the eventnumber to delete: ";
    cin >> targeteventNo;

    bool eventFound = false;
    for (int i = 0; i < eventCount; ++i) {
        if (events[i].eventNo == targeteventNo) {
            for (int j = i; j < eventCount - 1; ++j) {
             events[j] = events[j + 1];
            }
            --eventCount;
            cout << "event deleted successfully.\n";
            eventFound = true;
            break;
        }
    }

    if (!eventFound) {
        cout << "event not found.\n";
    }
}
```

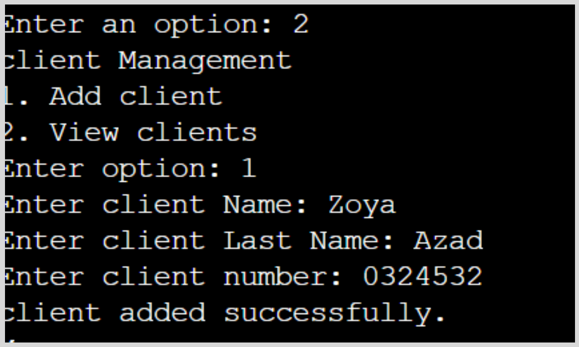
OUTPUT:



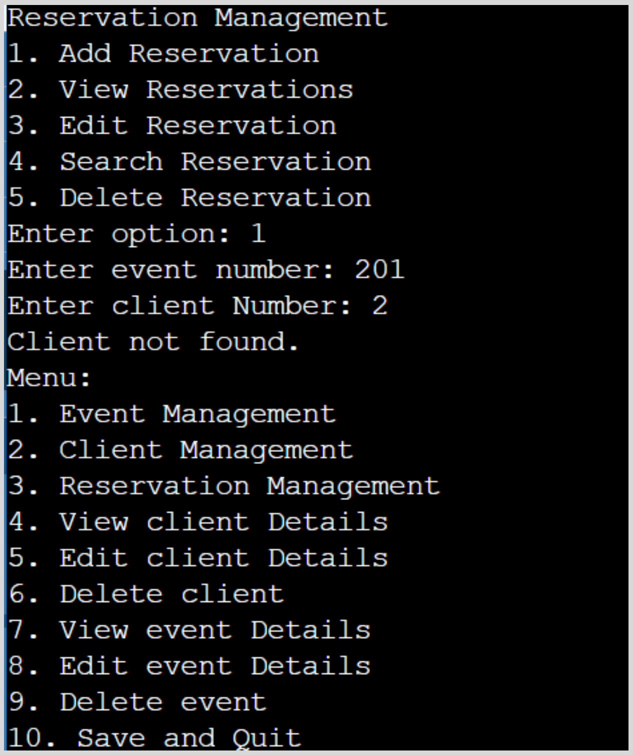
```
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
Enter an option: 
```

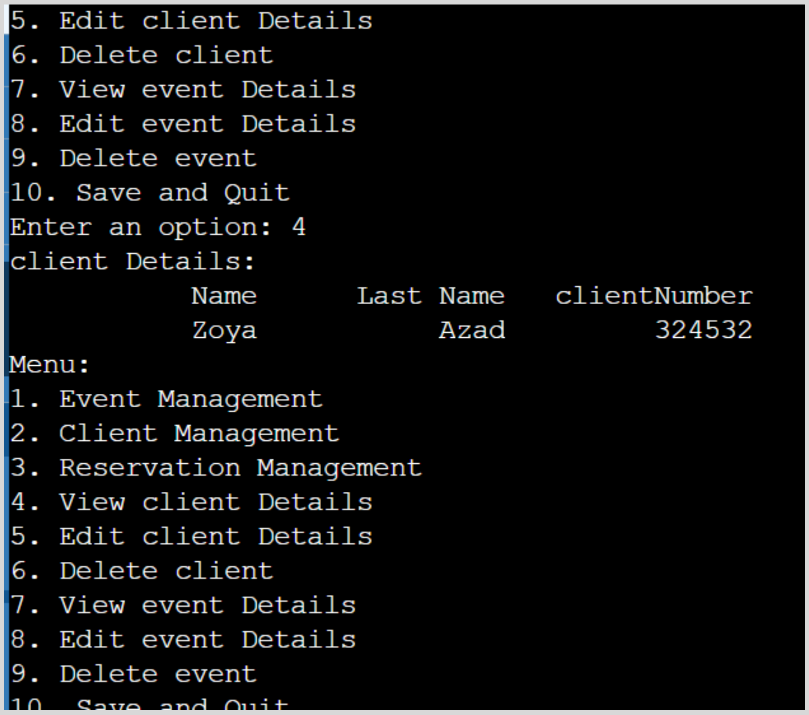


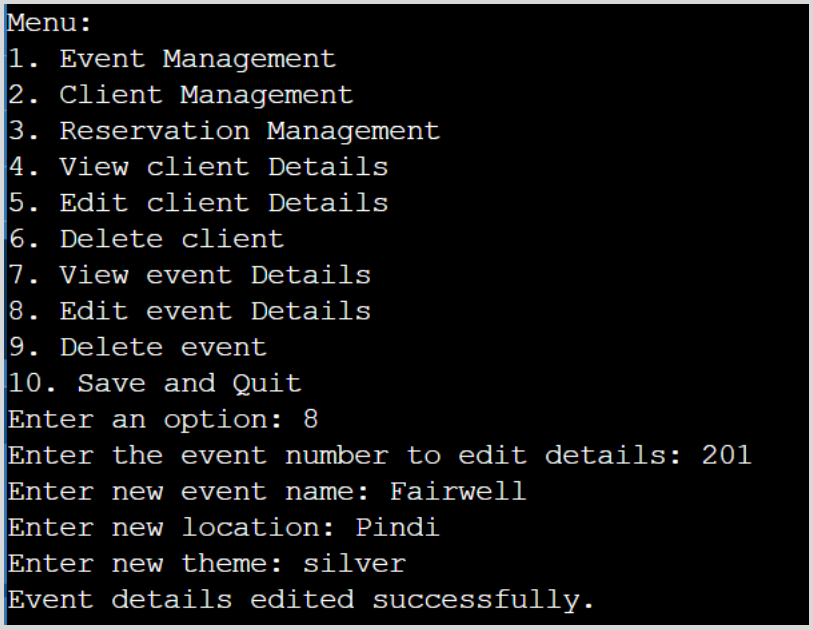
```
Enter an option: 1
Event Management
1. Add event
2. View event
Enter option: 1
Enter Event Number: 201
Enter Event Name: Wedding
Enter location: islambad
Enter theme: gold
event added successfully.
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
```



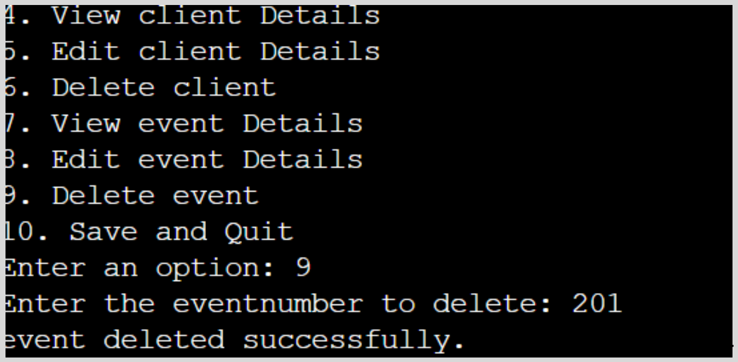
```
Enter an option: 2
client Management
1. Add client
2. View clients
Enter option: 1
Enter client Name: Zoya
Enter client Last Name: Azad
Enter client number: 0324532
client added successfully.
```


```
Reservation Management
1. Add Reservation
2. View Reservations
3. Edit Reservation
4. Search Reservation
5. Delete Reservation
Enter option: 1
Enter event number: 201
Enter client Number: 2
Client not found.
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
```

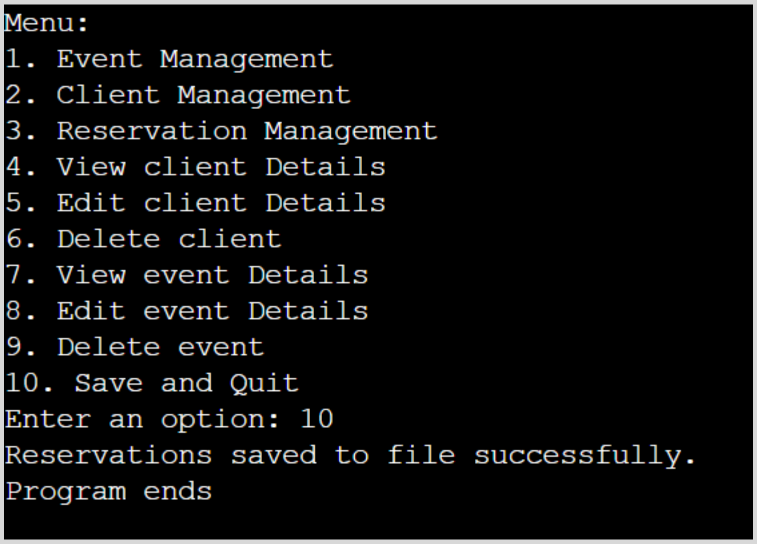

```
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
Enter an option: 4
client Details:
           Name      Last Name   clientNumber
           Zoya           Azad         324532
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
```



```
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
Enter an option: 8
Enter the event number to edit details: 201
Enter new event name: Fairwell
Enter new location: Pindi
Enter new theme: silver
Event details edited successfully.
```



```
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
Enter an option: 9
Enter the eventnumber to delete: 201
event deleted successfully.
```



```
Menu:
1. Event Management
2. Client Management
3. Reservation Management
4. View client Details
5. Edit client Details
6. Delete client
7. View event Details
8. Edit event Details
9. Delete event
10. Save and Quit
Enter an option: 10
Reservations saved to file successfully.
Program ends
```