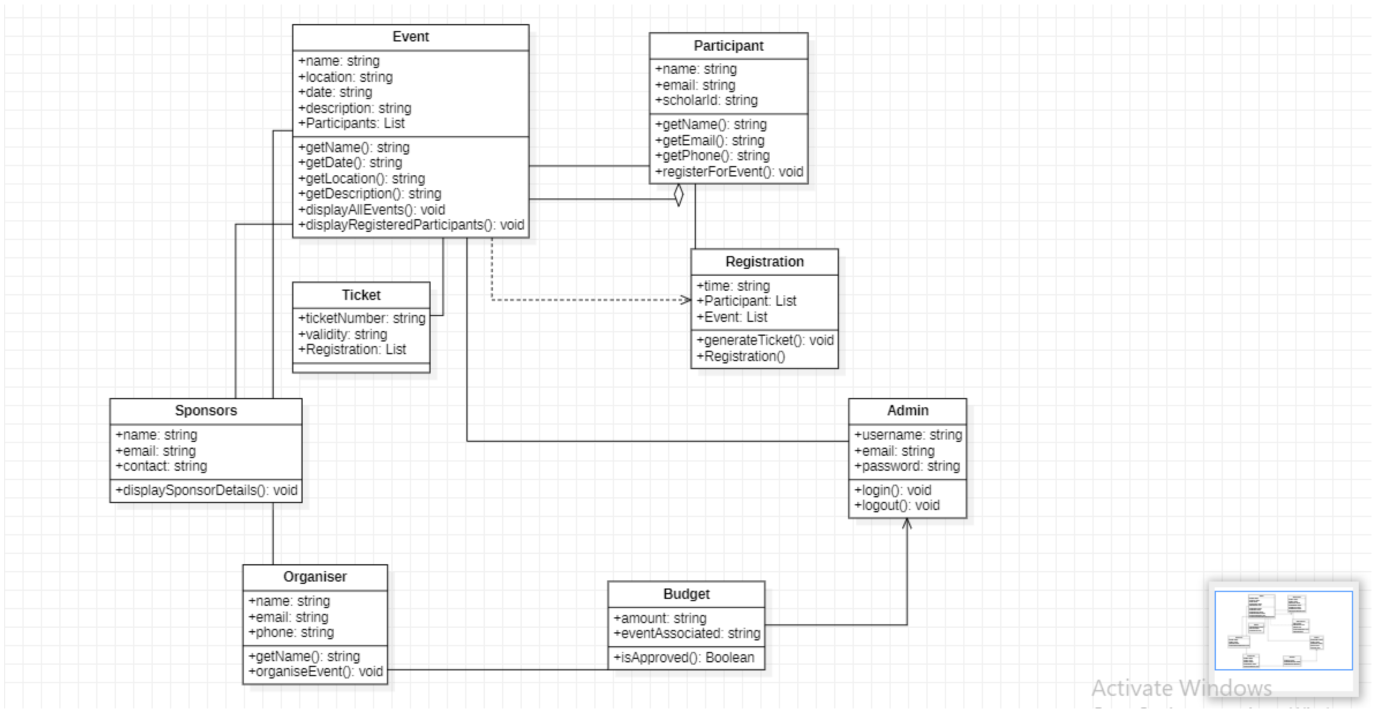
StarUml



Code

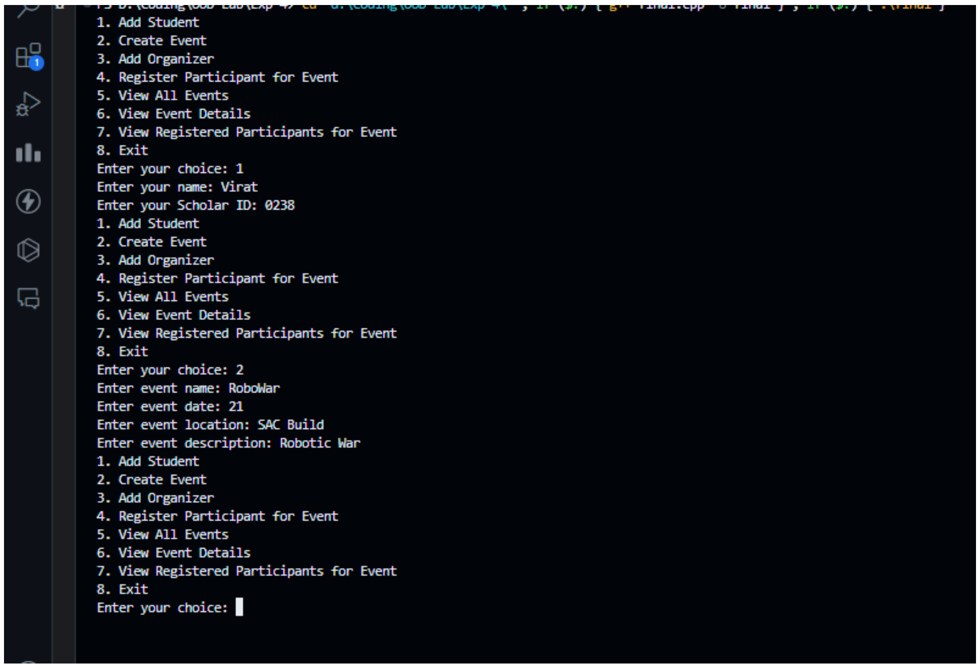
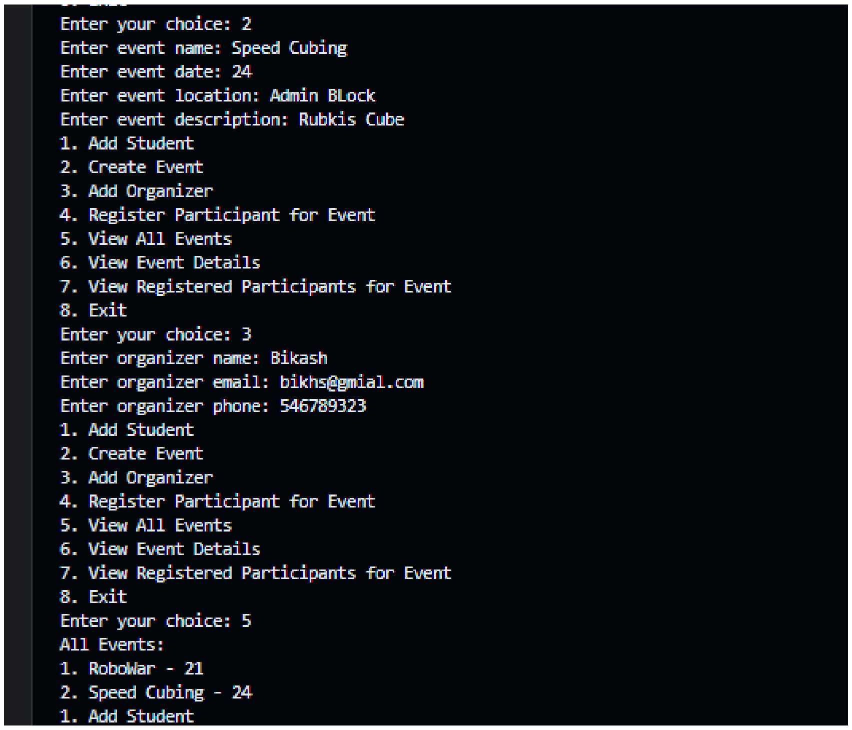
```
#include <iostream>
#include <vector>
#include <string>
#include <ctime>

using namespace std;

// Forward declaration of classes
class Participant;
class Organizer;

class Event
{
   string name;
   string date;
   string location;
   string description;
   vector<Participant *> participants;
   Organizer *organizer;
public:
   Event(string n, string d, string l, string desc, Organizer *org) : name(n), date(d), location(l),
description(desc), organizer(org) {}
   void addParticipant(Participant *p) { participants.push_back(p); }
   void removeParticipant(Participant *p)
   { /* Implement removal logic */
   }
   string getName() const { return name; }
   string getDate() const { return date; }
   string getLocation() const { return location; }
   string getDescription() const { return description; }
   const vector<Participant *> &getParticipants() const { return participants; }
   // string getOrganizer() const { return organizer->getName(); }
};

class Participant
{
   string name;a
   string email;
   string phone;

public:
   Participant(string n, string e, string ph) : name(n), email(e), phone(ph) {}
   void registerForEvent(Event *event) { event->addParticipant(this); }
   const string &getName() const { return name; }
   const string &getEmail() const { return email; }
   const string &getPhone() const { return phone; }
};

class Organizer
{
   string name;
   string email;
   string phone;

public:
   Organizer(string n, string e, string ph) : name(n), email(e), phone(ph) {}
   void organizeEvent(Event *event)
   { /* Implement event organization logic */
   }
   string getName() const { return name; }
};

class Registration
```

```
{
   Participant *participant;
   Event *event;
   time_t registrationDate;

public:
   Registration(Participant *p, Event *e) : participant(p), event(e)
   {
      registrationDate = time(nullptr);
   }
   void generateTicket()
   { /* Implement ticket generation logic */
   }
};

class Ticket
{
   Registration *registration;
   string ticketNumber;
   string validity;

public:
   Ticket(Registration *reg, string num, string valid) : registration(reg), ticketNumber(num), validity(valid)
{}
};

void displayAllEvents(const vector<Event *> &events)
{
   cout << "All Events:" << endl;
   for (size_t i = 0; i < events.size(); ++i)
   {
      cout << i + 1 << ". " << events[i]->getName() << " - " << events[i]->getDate() << endl;
   }
}

void displayEventDetails(const vector<Event *> &events)
{
}
void displayRegisteredParticipants(const vector<Event *> &events)
{
}

int main()
{
```

```
vector<Event *> events;
vector<Participant *> participants;
vector<Organizer *> organizers;

// Main loop for user interaction
while (true)
{
    cout << "1. Add Student" << endl;
    cout << "2. Create Event" << endl;
    cout << "3. Add Organizer" << endl;
    cout << "4. Register Participant for Event" << endl;
    cout << "5. View All Events" << endl;
    cout << "6. View Event Details" << endl;
    cout << "7. View Registered Participants for Event" << endl;
    cout << "8. Exit" << endl;
    cout << "Enter your choice: ";
    int choice;
    cin >> choice;

    switch (choice)
    {
    case 1:
    {
        string name, email, phone;
        cout << "Enter your name: ";
        cin.ignore();
        getline(cin, name);
        email = name + "21_ug@cse.nits.ac.in";
        cout << "Enter your Scholar ID: ";
        getline(cin, phone);
        Participant *participant = new Participant(name, email, phone);
        participants.push_back(participant);
        break;
    }
    case 2:
    {
        string name, date, location, description;
        cout << "Enter event name: ";
        cin.ignore(); // Clear buffer
        getline(cin, name);
        cout << "Enter event date: ";
        getline(cin, date);
        cout << "Enter event location: ";
        getline(cin, location);
    cout << "Enter event description: ";
    getline(cin, description);
    Organizer *org = new Organizer("Organizer Name", "org@example.com", "1234567890");
    Event *event = new Event(name, date, location, description, org);
    events.push_back(event);
    break;
}
case 3:
{
    string name, email, phone;
    cout << "Enter organizer name: ";
    cin.ignore(); // Clear buffer
    getline(cin, name);
    cout << "Enter organizer email: ";
    getline(cin, email);
    cout << "Enter organizer phone: ";
    getline(cin, phone);
    Organizer *organizer = new Organizer(name, email, phone);
    organizers.push_back(organizer);
    break;
}
case 4:
{
    // Assuming events, participants, and organizers exist
    // and you have appropriate checks in place
    int eventIndex, participantIndex;
    cout << "Enter index of event: ";
    cin >> eventIndex;
    cout << "Enter index of participant: ";
    cin >> participantIndex;
    participants[participantIndex]->registerForEvent(events[eventIndex]);
    break;
}
case 5:
    displayAllEvents(events);
    break;
case 6:
    displayEventDetails(events);
    break;
case 7:
    displayRegisteredParticipants(events);
    break;
case 8:
    for (Event *event : events)
```

```
            delete event;
          for (Participant *participant : participants)
            delete participant;
          for (Organizer *organizer : organizers)
            delete organizer;
          return 0;
        default:
          cout << "Invalid choice. Please try again." << endl;
        }
    }

    return 0;
}
```

OUTPUT


```
1. Add Student
2. Create Event
3. Add Organizer
4. Register Participant for Event
5. View All Events
6. View Event Details
7. View Registered Participants for Event
8. Exit
Enter your choice: 1
Enter your name: Virat
Enter your Scholar ID: 0238
1. Add Student
2. Create Event
3. Add Organizer
4. Register Participant for Event
5. View All Events
6. View Event Details
7. View Registered Participants for Event
8. Exit
Enter your choice: 2
Enter event name: RoboWar
Enter event date: 21
Enter event location: SAC Build
Enter event description: Robotic War
1. Add Student
2. Create Event
3. Add Organizer
4. Register Participant for Event
5. View All Events
6. View Event Details
7. View Registered Participants for Event
8. Exit
Enter your choice: 
```

```
Enter your choice: 2
Enter event name: Speed Cubing
Enter event date: 24
Enter event location: Admin BLock
Enter event description: Rubkis Cube
1. Add Student
2. Create Event
3. Add Organizer
4. Register Participant for Event
5. View All Events
6. View Event Details
7. View Registered Participants for Event
8. Exit
Enter your choice: 3
Enter organizer name: Bikash
Enter organizer email: bikhs@gmial.com
Enter organizer phone: 546789323
1. Add Student
2. Create Event
3. Add Organizer
4. Register Participant for Event
5. View All Events
6. View Event Details
7. View Registered Participants for Event
8. Exit
Enter your choice: 5
All Events:
1. RoboWar - 21
2. Speed Cubing - 24
1. Add Student
```