King Saud University
College of Computer and Information Sciences

Event Management System

NAME	WORK
فاطمة محمد	Bulid and design Event class and eventmangement system class implements the data structure and Gui
جمانة ابراهيم المشهراوي	Bulid and design Organizer class and Person class eventmangement system class implements the exceptions and Gui
شيماء حمود قاسم	Bulid and design Participant class and VIPParticipant class and eventmangement system implement files and Gui

Project Description:

This event management system is built around a core Person class, which defines the basic details for everyone in the system. From this foundation, we have three specific roles: Organizers who create and run events, regular Participants who attend them, and VIPParticipants who get special status. The heart of the system is the Event class, which represents a single event and is created by an Organizer. All of this is brought together in the main EventManagementSystem program, which presents a simple menu that lets an organizer register, create an event, add or remove attendees, and view everyone who's signed up. To help you get started, the system comes with a few sample attendees already loaded in.

Modifications:

Graphical User Interface (GUI): The system now uses **Java Swing** components to improve user interaction, making the process of creating events, adding attendees, and viewing lists much more intuitive compared to the console version.

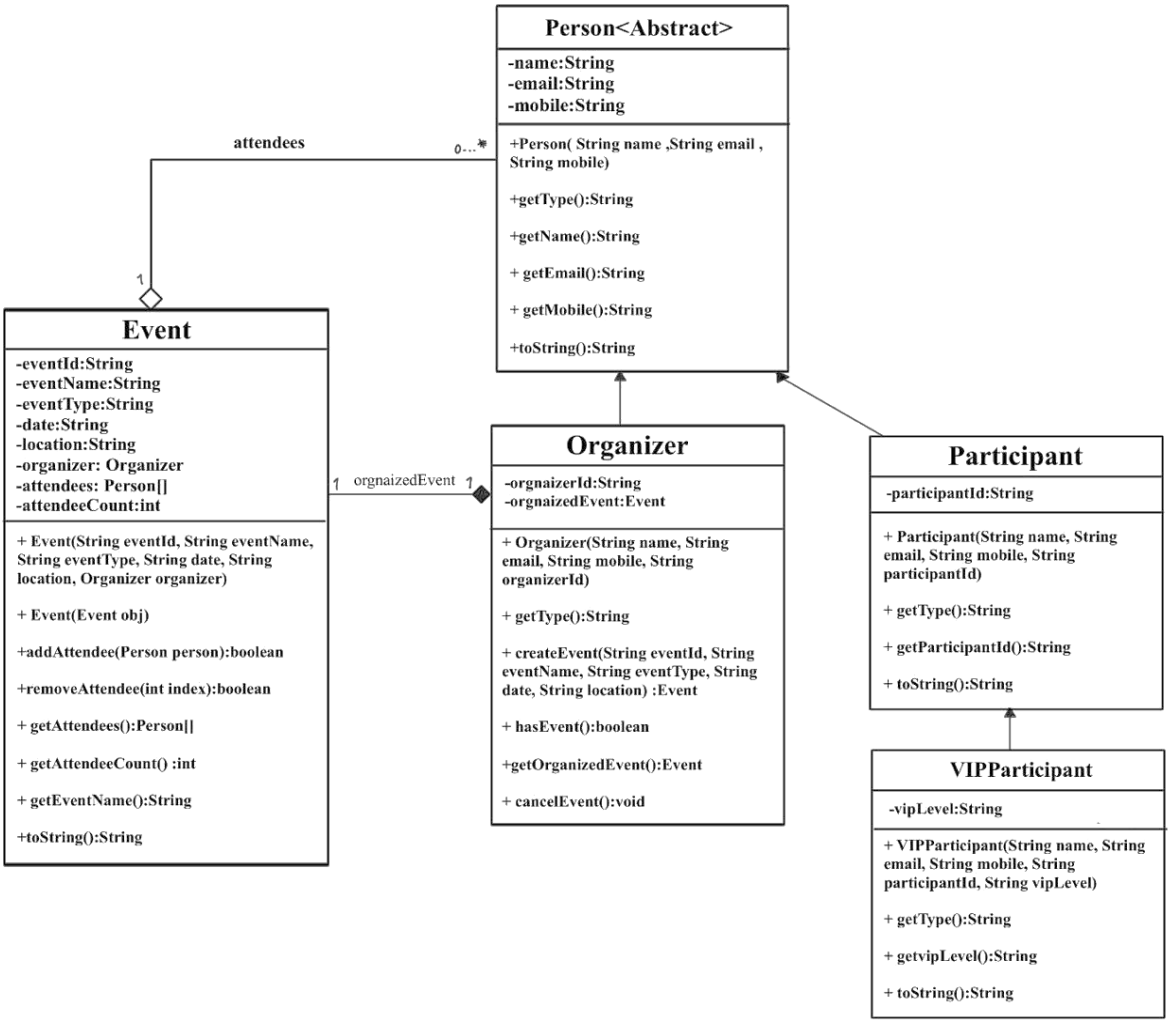
Data Handling: File management is implemented to save and load organizer and event records, ensuring **data persistence**. Files like Organizers.dat and Events.dat are used for storing system information securely.

Generic Linked List: Attendees are now stored using a custom **Generic Linked List (PLinkedList<T>)** structure instead of fixed-size arrays, providing more efficient memory management and flexibility to handle an unlimited number of participants.

The system also **handles exceptions** to ensure smoother operation and prevent crashes:

Exception Handling & Data Persistence: To ensure system robustness, we implemented custom exceptions (EventAlreadyExistsException, EmptyFieldException) for logical validation, alongside handling standard exceptions like IOException. Furthermore, a FileHandler class was integrated to manage data persistence, enabling the saving and loading of system records via the event_data.dat file,

UML:


Person<Abstract>
-name:String
-email:String
-mobile:String
+Person(String name ,String email , String mobile)
+getType():String
+getName():String
+ getEmail():String
+ getMobile():String
+toString():String
attendees
0...*
1
Event
-eventId:String
-eventName:String
-eventType:String
-date:String
-location:String
-organizer: Organizer
-attendees: Person[]
-attendeeCount:int
+ Event(String eventId, String eventName, String eventType, String date, String location, Organizer organizer)
+ Event(Event obj)
+addAttendee(Person person):boolean
+removeAttendee(int index):boolean
+ getAttendees():Person[]
+ getAttendeeCount() :int
+ getEventName():String
+toString():String
1
orgnaizedEvent
1
Organizer
-orgnaizerId:String
-orgnaizedEvent:Event
+ Organizer(String name, String email, String mobile, String organizerId)
+ getType():String
+ createEvent(String eventId, String eventName, String eventType, String date, String location) :Event
+ hasEvent():boolean
+getOrganizedEvent():Event
+ cancelEvent():void
Participant
-participantId:String
+ Participant(String name, String email, String mobile, String participantId)
+ getType():String
+ getParticipantId():String
+ toString():String
VIPParticipant
-vipLevel:String
+ VIPParticipant(String name, String email, String mobile, String participantId, String vipLevel)
+ getType():String
+ getvipLevel():String
+ toString():String

Classes and methods headers:

Person class

- Person(name, email, mobile)
  Constructor to create person with basic details
- getType()
  Abstract Method must return person type
- getName()
  Returns person's name
- getEmail()
  Returns email address
- getMobile()
  Returns mobile number
- toString()
  Returns a string representation of the object

Organizer class

- Organizer(name, email, mobile, organizerId)
  Constructor to create organizer with ID
- getType()
  Inherited Abstract method returns "Organizer"
- createEvent()
  Creates new event (composition)
- hasEvent()
  Checks if organizer has an event
- getOrganizedEvent()
  Returns current event
- cancelEvent()
  Removes the current event

Participant Class

- Participant(name, email, mobile, participantId)
  Constructor to create participant with ID
- getType()
  Inherited Abstract returns "Regular Participant"
- getParticipantId()
  Returns participant ID

- toString()

Adds ID to person details

VIP Participant Class

- VIPParticipant(name, email, mobile, participantId, vipLevel)
  Constructor to create VIP participant with level

  - getType()

  Inherited Abstract returns "VIP Participant (level)"

- getVipLevel()
  Returns VIP level

  - toString()
    Adds VIP level to participant details

Event Class:

- **public Event (String eventId, String eventName, String eventTyp,String date, String location, Organizer organizer)**
  Event constructor used to initialize the event ID, name, type, date and location, along with the Organizer object responsible for creating and managing the event

- **public Event (Event obj)**
  A copy constructor that creates a new instance of the Event class by copying the attributes from an existing Event object.

- **public boolean addAttendee (Person person)**
  A method adds a Person object in the first empty location in the attendee's array and returns true if the addition was successful and false otherwis

- **public boolean removeAttendee(int index)**
  method that removes an attendee from the event based on the given index. It replaces the removed attendee with the last attendee in the list to maintain continuity and returns true if the removal is successful, and false otherwise

- **public Person[] getAttendees()**
  Returns an array of Person objects representing all the attendees of this event

- **public int getAttendeeCount()**
  Returns the total number of attendees registered for this event

- **public String getEventName()**
  Returns the name of the event as a String

public String toString()

Returns a string representation of the Event object, showing key details such as the event ID, name, type, date, location, and the organizer's name

EventmangementSystem Class:

public static void main(String[] args):

The main method serves as the entry point of the Event Management System program

1. It begins by registering an event organizer by collecting their basic information such as name, email, mobile number, and ID. Using this information, it creates an Organizer object to represent the registered organizer

2. After registration, the program automatically adds participants and VIP participants of the event to the system

3. Once the setup is complete, the method displays a menu that allows the organizer to manage events

4. The menu provides options to create a new event, add attendees, remove attendees, display attendees by type, cancel an event, or exit the program

5. After each action, the menu is shown again, allowing the organizer to perform multiple operations during the same session

6. The program keeps running in this loop until the organizer chooses the "Exit" option, at which point a goodbye message is displayed, and the program ends

Sample Run:

Orgnizer register and menu option 1(create new event)

```
=== Organizer Registration ===
Enter organizer name: fatima
Enter organizer email: fatim332@gmail.com
Enter organizer mobile: 05688877
Enter organizer ID: 22
Welcome, fatima! (ID: 22)
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 1
Create New Event
Enter event name: Cyber Security Fundamentals
Enter event type: tech
Enter event date: 5/10/2025
Enter event location: Riyadh
Event created successfully: Cyber Security Fundamentals
Event Details: Event ID: EVT6, Name: Cyber Security Fundamentals, Type: tech, Date: 5/10/2025, Location: Riyadh
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
```

menu option 2(add attendee)

```
Enter your choice: 2
Add Attendee
Available People:
1. Sara - Regular Participant - sara@email.com
2. Khalid - Regular Participant - khalid@email.com
3. Nora - VIP Participant (GOLD) - nora@email.com
4. Mohammed - VIP Participant (PLATINUM) - mohammed@email.com
Enter person number to add: 2
Successfully added Khalid to the event!
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 2
Add Attendee
Available People:
1. Sara - Regular Participant - sara@email.com
2. Khalid - Regular Participant - khalid@email.com
3. Nora - VIP Participant (GOLD) - nora@email.com
4. Mohammed - VIP Participant (PLATINUM) - mohammed@email.com
Enter person number to add: 1
Successfully added Sara to the event!
***** Event Organizer System *****
```

menu option 3(Display attendee by type)

```
Enter person number to add: 1
Successfully added Sara to the event!
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 3
Display Attendees
Show (V: VIP, R: Regular, A: All): v
VIP Attendees
No VIP attendees found.
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 3
Display Attendees
Show (V: VIP, R: Regular, A: All): a
All Attendees
```

menu option 4(Remove atteende)

```
Show (V: VIP, R: Regular, A: All): a
All Attendees
- Khalid - Regular Participant - khalid@email.com
- Sara - Regular Participant - sara@email.com
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 4
Remove Attendee
Current Attendees:
1. Khalid - Regular Participant - khalid@email.com
2. Sara - Regular Participant - sara@email.com
Enter attendee number to remove: 2
Successfully removed Sara from the event!
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
```

menu option 5,6(cancel event and exit)

```
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 5
Are you sure you want to cancel event: Cyber Security Fundamentals? (yes/no): yes
Event cancelled successfully!
***** Event Organizer System *****
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit
Enter your choice: 6
Goodbye
BUILD SUCCESSFUL (total time: 10 minutes 36 seconds)
```

Node Class *(Generic <T>)*

- **Node(T obj)** Constructor to create a node with specific data and null next reference.
- **Node(T obj, `Node<T> node`)** Constructor to create a node with data and a reference to the next node.
- **T getData()** Returns the data stored in the node.
- **setData(T data)** Updates the data stored in the node.
- **Node <T> getNext()** Returns the reference to the next node in the list.
- **setNext(`Node<T> next`)** Updates the reference to the next node.

PLinkedList Class *(Generic <T>)*

- **PLinkedList()** Constructor to initialize an empty linked list (head is set to null).
- **isEmpty()** Checks if the list is empty (returns true if head is null).
- **insertAtFront(item)** Adds a new item to the beginning of the list .
- **remove(item)** Searches for an item using `equals()` and removes it from the list .
- **print()** Traverses the list and prints all stored items.
- **getHead()** Returns the first node to allow traversal in other classes

Exceptions:

Define Exception:

1. **EventAlreadyExistsException** :
   Purpose: Thrown when an organizer tries to create a new event while already having an active event.
   Location: createNewEvent() method → checkEventExists() method
   Message: "You already have an active event! Current event: [Event Name]"
   How to trigger: Try to create a second event without canceling the first one.

2. **EmptyFieldException :**
   Purpose: Thrown when any required field is left empty during event creation.
   Location: createNewEvent() method → validateEventFields() method
   Message:
   "Event name cannot be empty!"
   "Event type cannot be empty!"
   "Event date cannot be empty!"
   "Event location cannot be empty!"
   How to trigger: Leave any field empty or enter invalid email during organizer registration.

3. **AttendeeNotFoundException** :
   Purpose: Thrown when trying to add a participant that doesn't exist in the system.
   Location: addAttendee() method → findParticipant() method
   Message: "Attendee '[name]' not found! Please check spelling."
   How to trigger: Try to add an attendee with a name that doesn't exist in the available participants list.

Checked Exception:

IOException :
Purpose: Thrown when file writing operation fails while saving event data.
Location: saveEventToFile() method (try-catch inside same method)
Message: "Failed to save event to file: [error details]"
How to trigger: Create an event successfully (automatic save to file). If file system has issues, this exception will be caught.

Unchecked Exception:

IllegalArgumentException :
Purpose: Thrown when organizer registration fields are invalid or empty.
Location: Thrown in validateRegistration() method, caught in registerBtn.addActionListener()
Messages:
"Name cannot be empty!"
"Invalid email format!"
"Mobile cannot be empty!"
"Organizer ID cannot be empty!"
How to trigger: Leave any field empty or enter invalid email during organizer registration.

Files:

FileHandler Class

This class acts as the memory for our Event Management system. It's responsible for safely saving all the program's data to a file when you're done and loading it all back when you start again.

Saving the Data (saveData)
This is the "save" button for the entire program. You give it your current list of people and events, and it quietly packages everything up and stores it in a file named event_data.dat for safekeeping. If something goes wrong during the save, it will let you know.

Loading the Data (loadData)
This is the "load" function. When the program starts, it checks for that same event_data.dat file. If it finds a previous save, it unpacks everything and hands back your complete list of people and events, ready to go. If this is your first time using the program or the file is missing, it simply starts fresh.

The Data File (DATA_FILE)
This is simply the name of the file where all our information is kept, like a digital filing cabinet.

Built-in Safety Net (Error Handling)
The class is designed to be resilient. If it runs into any problems—like not being able to find the file or having trouble reading it—it won't crash your program. Instead, it will gracefully handle the issue and tell you what went wrong, allowing you to continue without losing your work.

GUI (Main Frame):

Purpose: The primary interface for the Event Organizer System. It initializes the system data (Organizer and People list) and provides a centralized console-like environment for managing events and attendees.

Components:

Buttons:
Create Event Prompts the user to enter event details via dialogs and creates a new event.
Add Attendee Displays available participants and adds a selected person to the current event.
Display Shows a detailed list of all attendees currently registered in the event.
Remove Removes a specific attendee from the event by name.
Exit Closes the application.

Text Area:
consoleScreen A read-only text area used to display system logs, operation results, and lists to the user.

Updated Methods description

• EventManagementGUI()

Constructor that initializes the main GUI window. Sets up the frame properties, card layout for screen switching, and adds registration and main menu panels.

• createRegistrationPanel()

Creates and returns the organizer registration panel with input fields for name, email, mobile, and organizer ID. Handles user registration and initializes sample data.

• validateRegistration(String name, String email, String mobile, String id)

Validates registration input fields. Throws IllegalArgumentException if any field is empty or email format is invalid (Unchecked Exception - propagation).

• createMainMenuPanel()

Creates and returns the main menu panel with navigation buttons for event management operations including create event, add attendee, display attendees, remove attendee, cancel event, and exit.

• createNewEvent()

Handles the creation of a new event with input validation. Checks for existing events, validates event fields, creates event object, and saves to file. Catches user-defined exceptions.

• checkEventExists()

Checks if the current organizer already has an active event. Throws EventAlreadyExistsException if an event already exists (User-Defined Exception).

• validateEventFields(String name, String type, String date, String location)

Validates that all event fields are filled. Throws EmptyFieldException if any field is empty (User-Defined Exception).

• saveEventToFile(Event event)
Saves event details to "event_data.txt" file. Handles IOException internally using try-catch block (Checked Exception handling).

addAttendee()

Handles the process of adding a new attendee to the current event. Checks for active event existence, displays available participants, and adds selected person to the event.

• findParticipant(String name)

Searches for a participant by name in the people list. Throws AttendeeNotFoundException if participant not found or not a Participant instance.

• displayAttendeesByType()

Displays event attendees filtered by type (VIP, Regular, or All). Shows different information based on participant category with formatted output.

• removeAttendee()

Removes an attendee from the current event. Displays current attendees list and removes the specified person after confirmation.

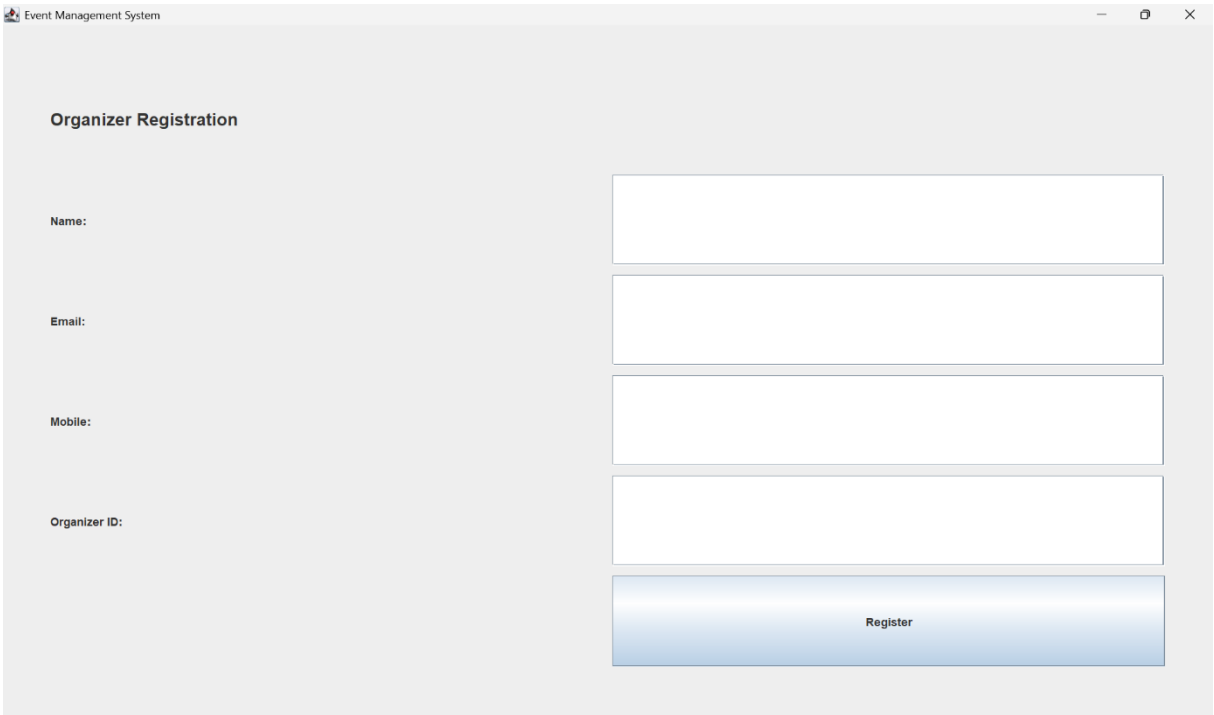
• cancelEvent()

Cancels the currently organized event after user confirmation. Handles event cancellation process and updates organizer status.

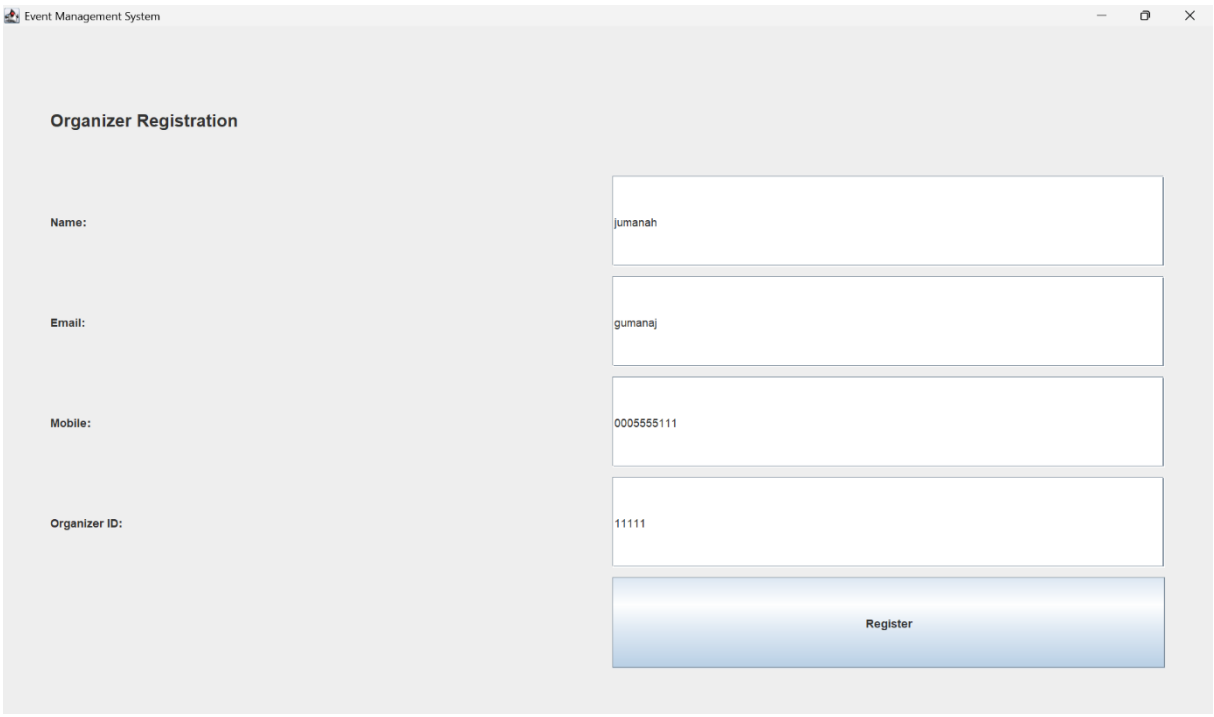
• main(String[] args)

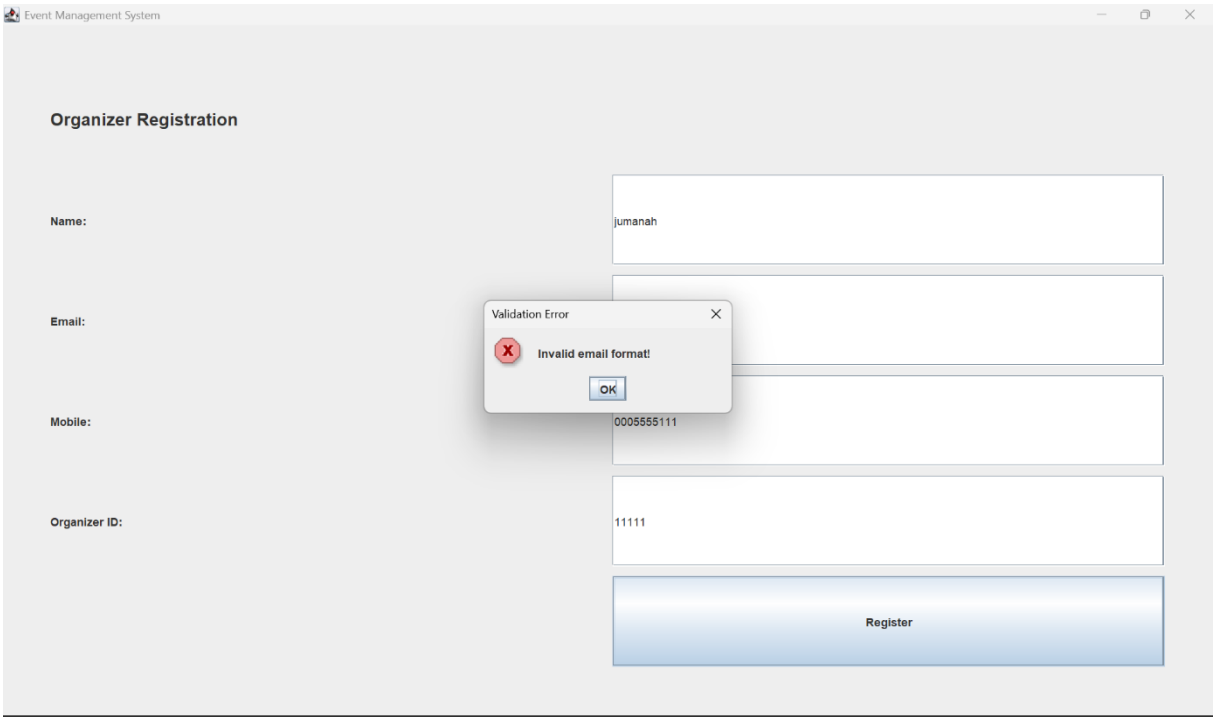
Entry point of the application. Initializes and displays the Event Management GUI using SwingUtilities.

GUI demo

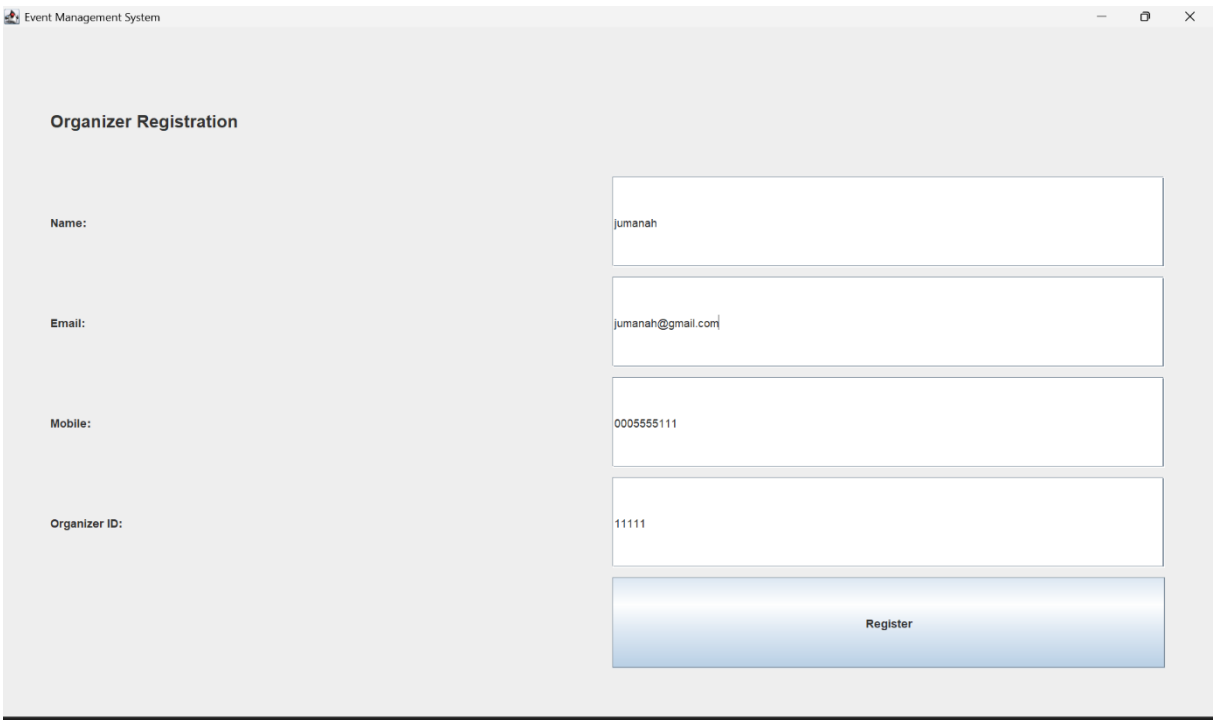


When writing invalid email

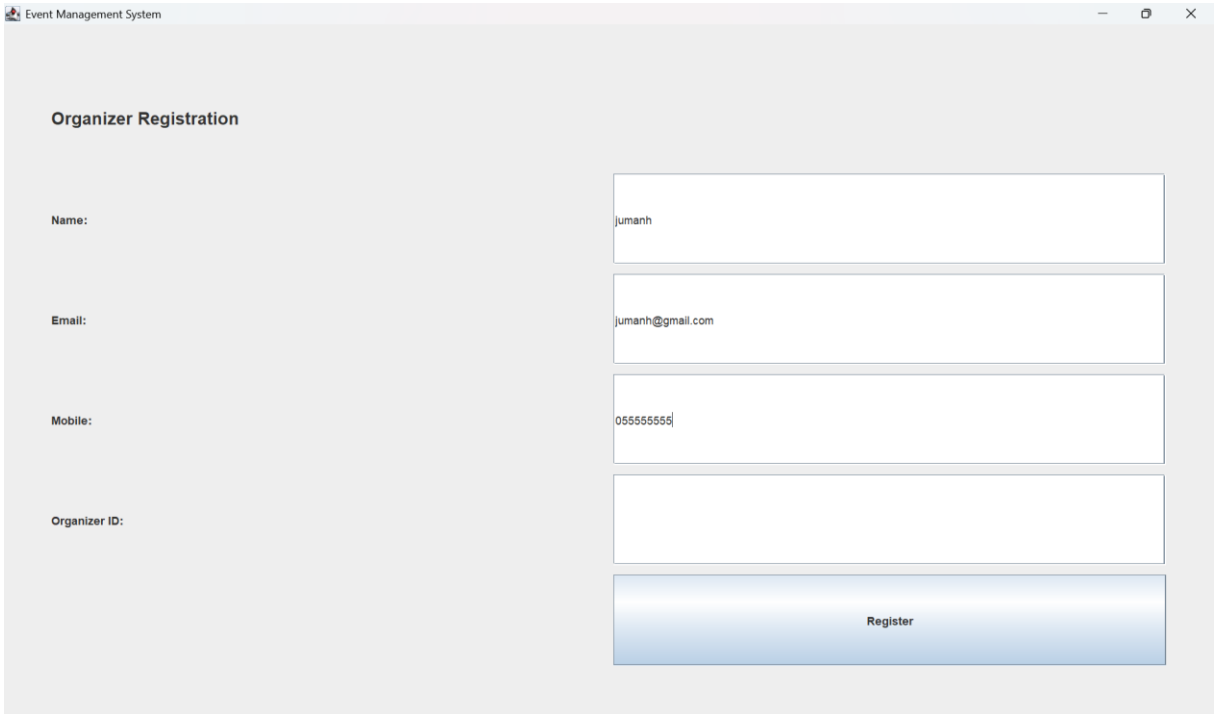


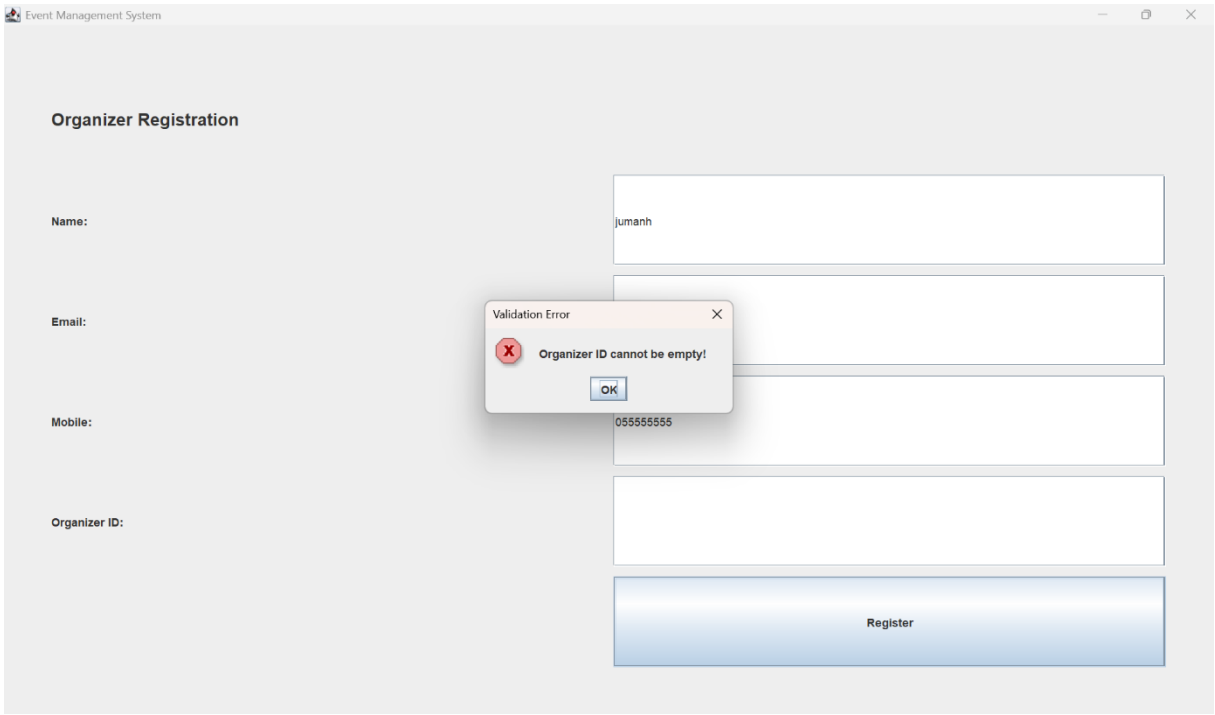


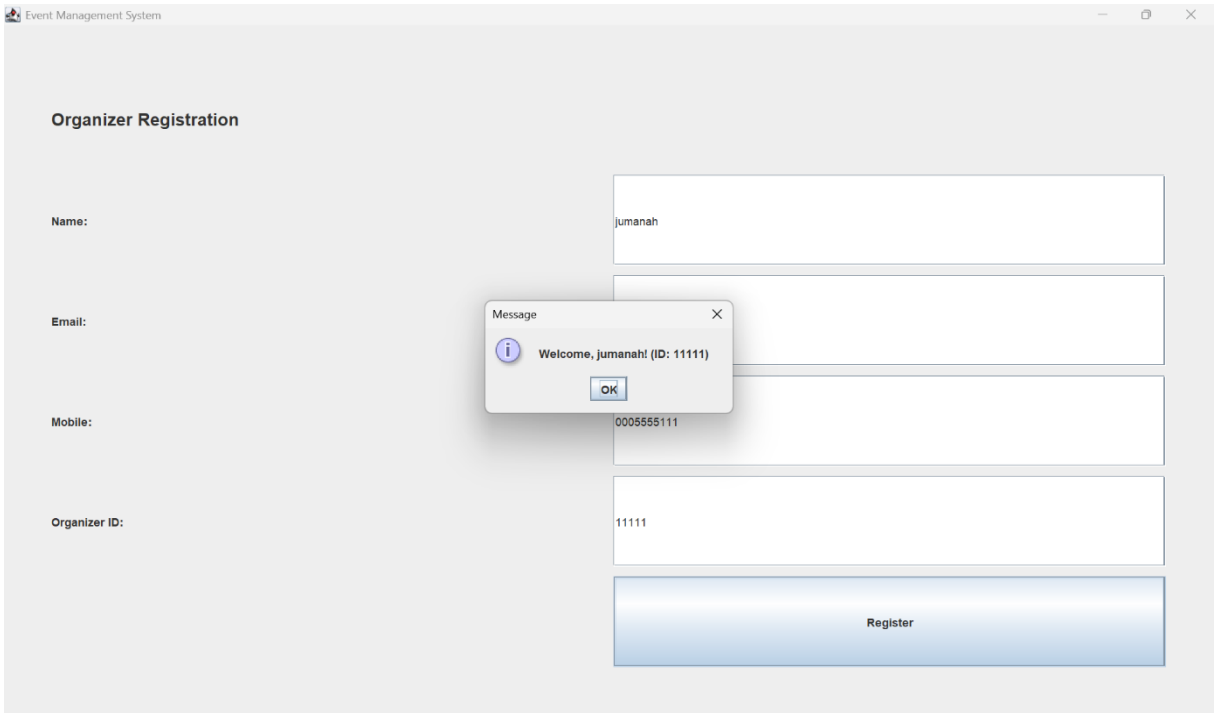
Correct email



When I don’t complete organizer information



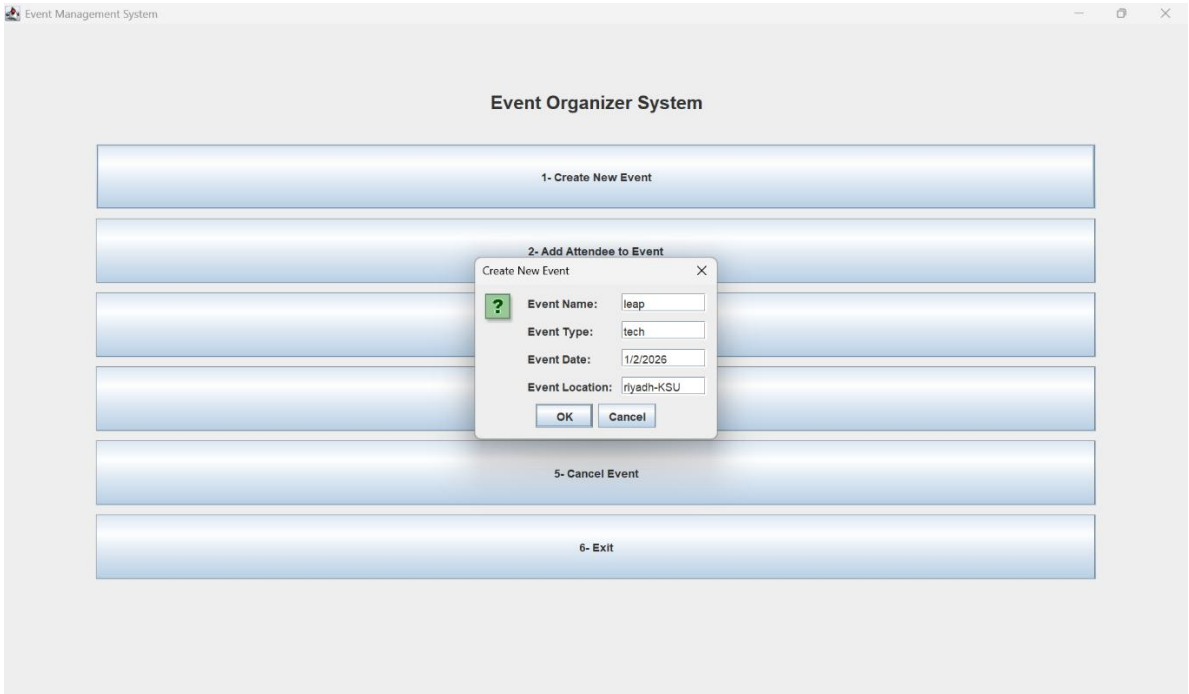


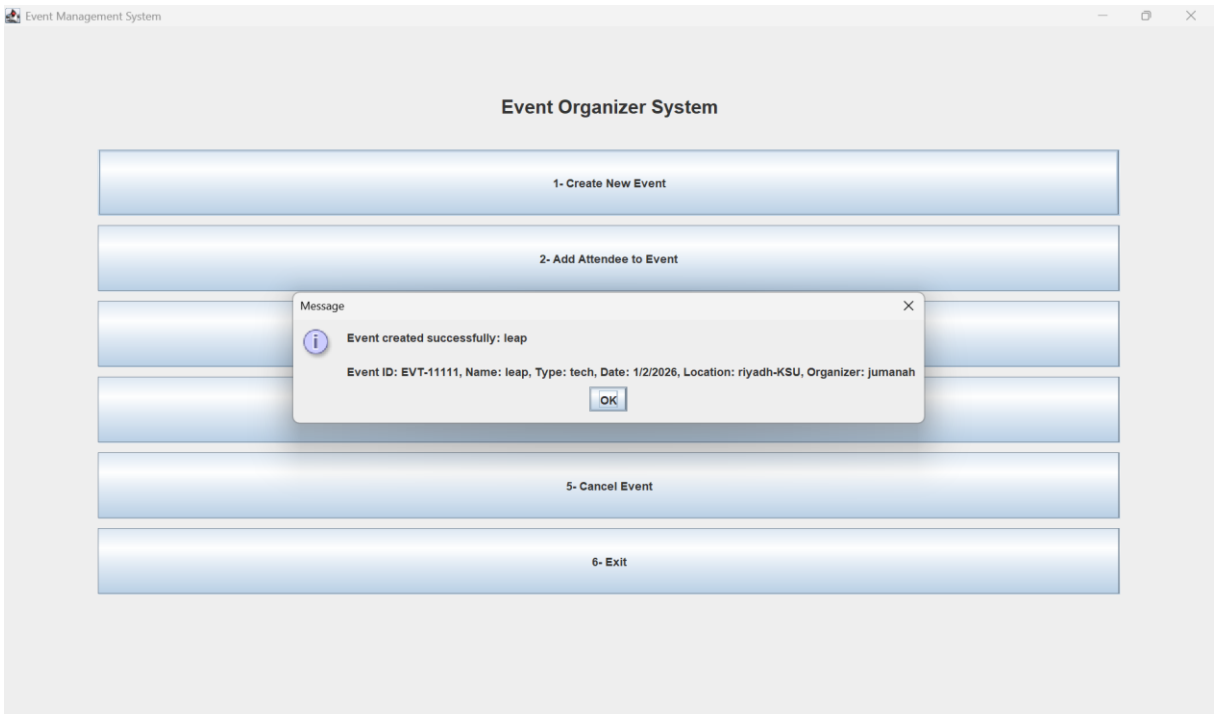

Event Management System
Organizer Registration
Name:
jumanah
Email:
Mobile:
0005555111
Organizer ID:
11111
Register
Message
Welcome, jumanah! (ID: 11111)
OK

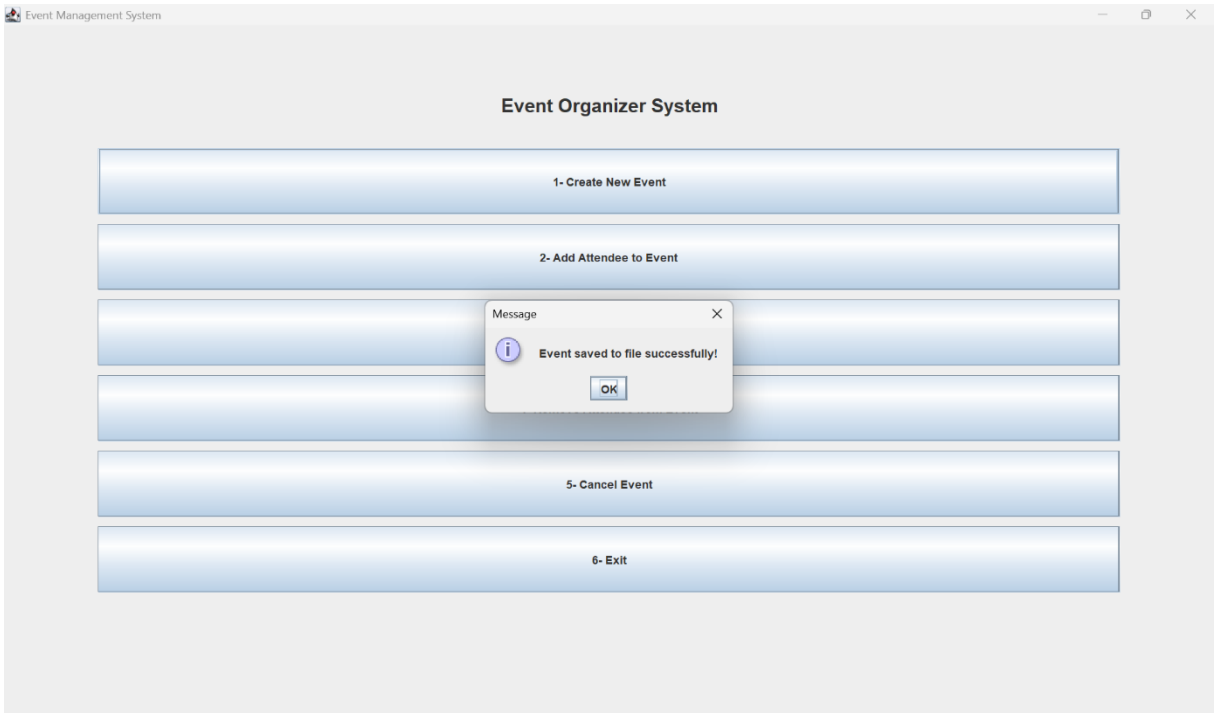
Menu


Event Management System
Event Organizer System
1- Create New Event
2- Add Attendee to Event
3- Display Attendees by Type
4- Remove Attendee from Event
5- Cancel Event
6- Exit

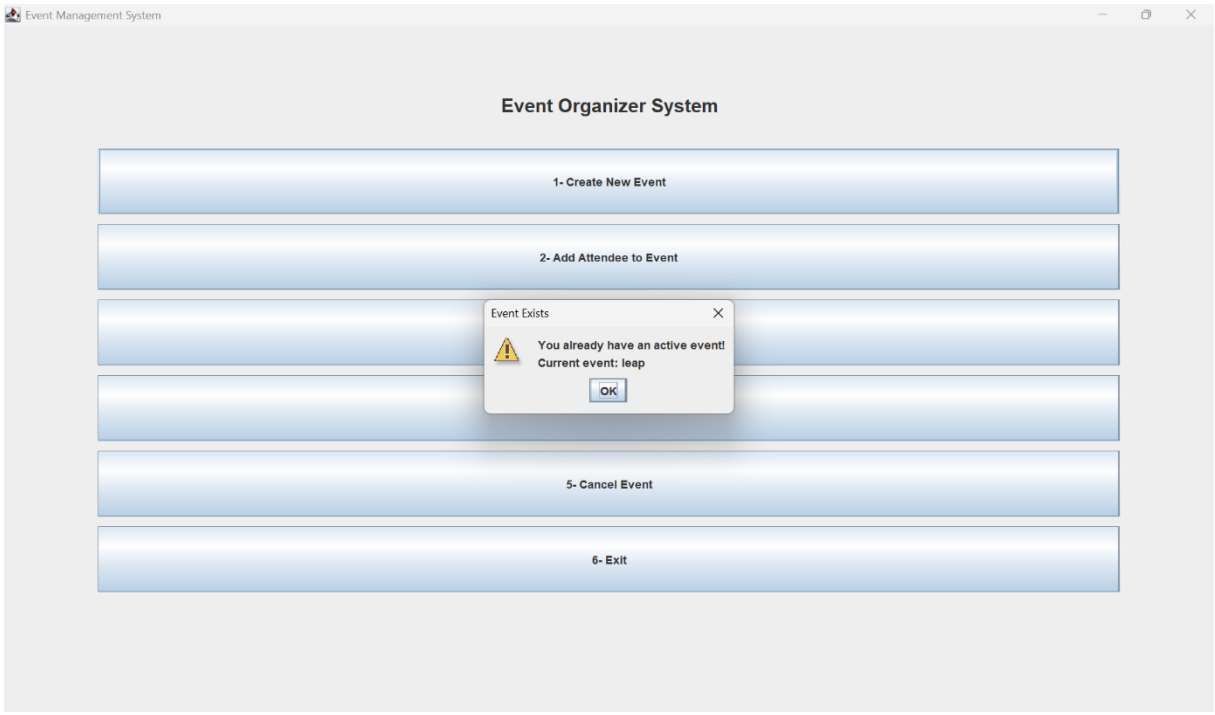
Create event



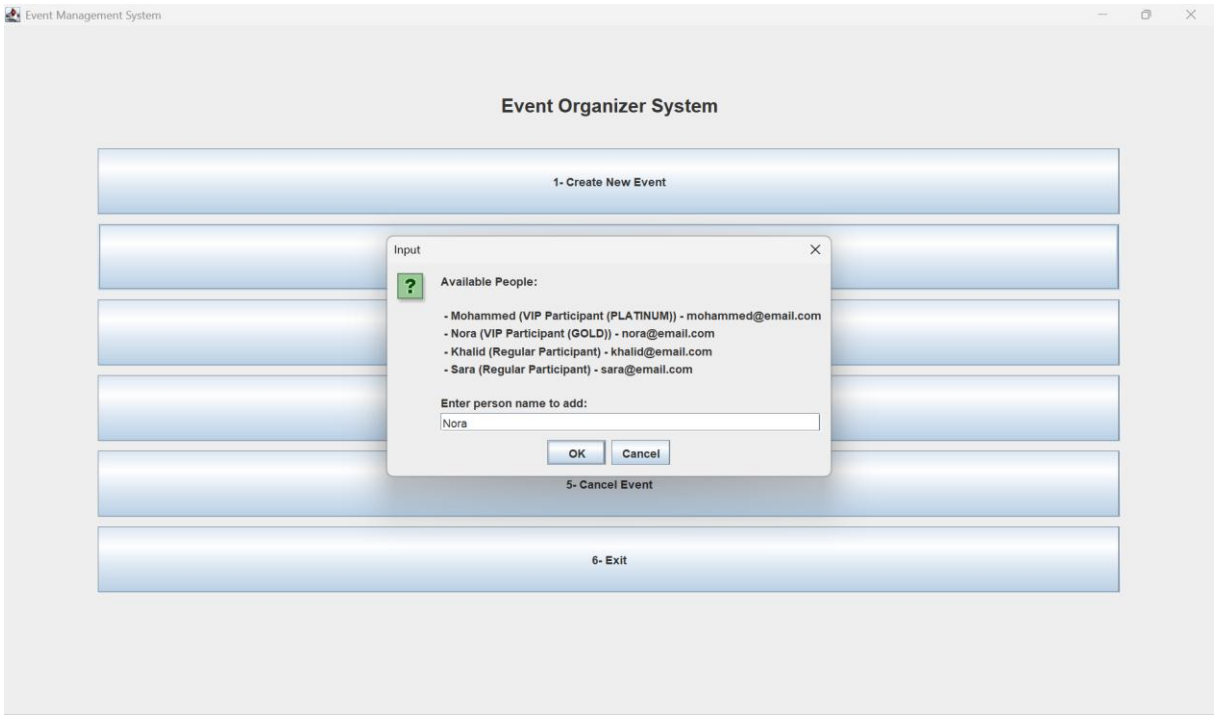


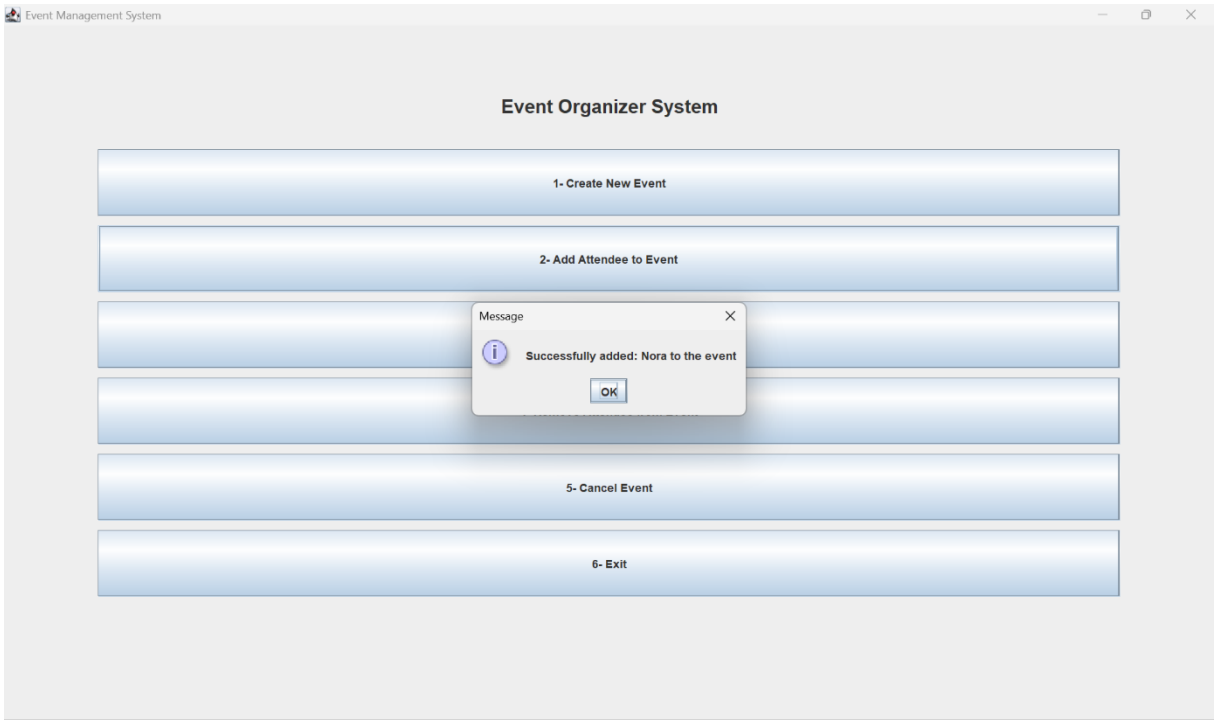


Create another event with same name

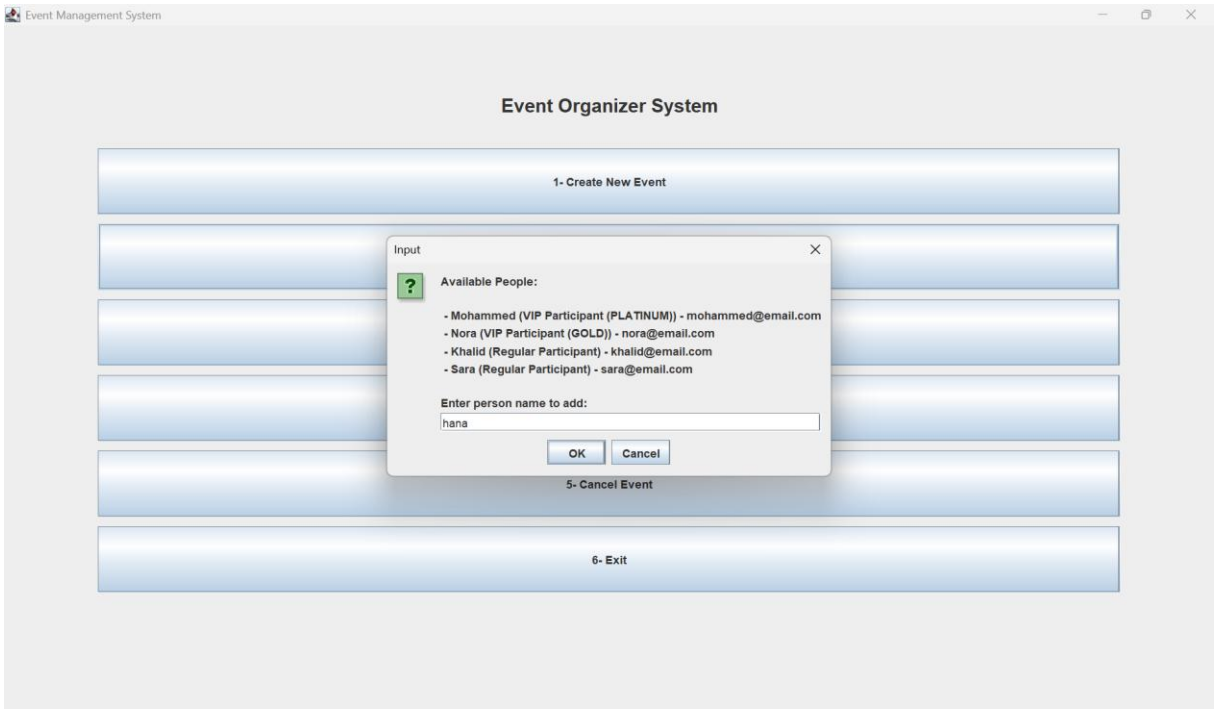


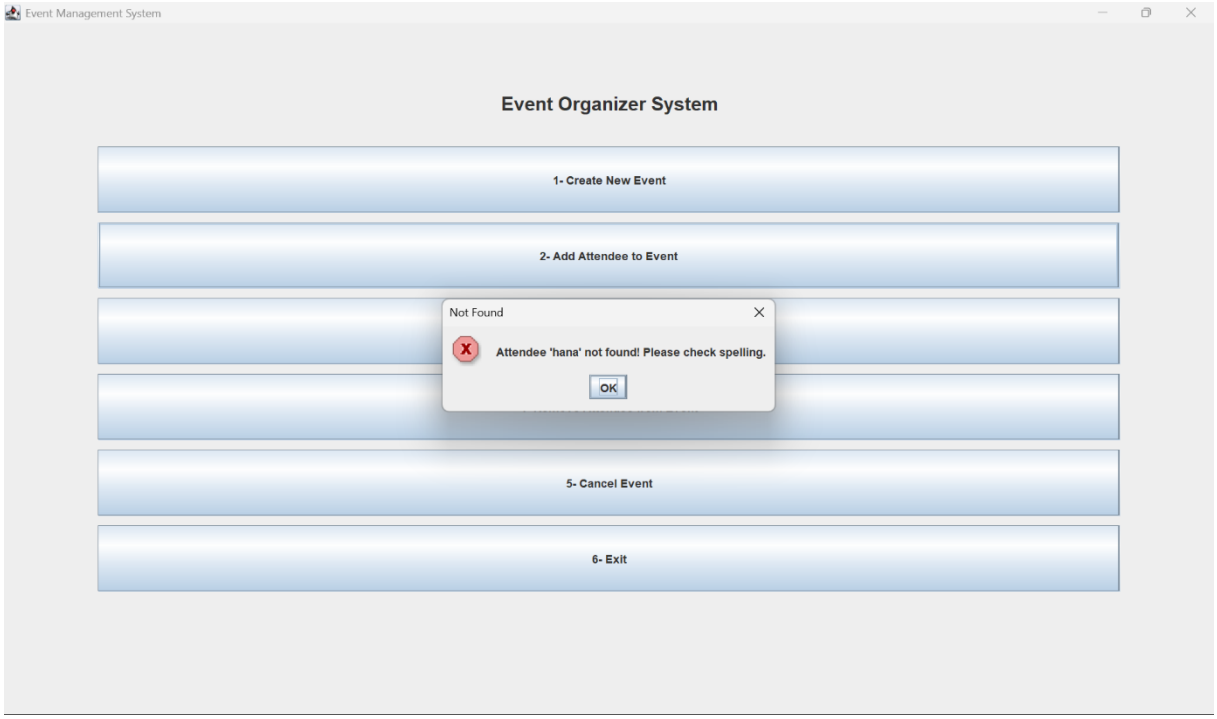
Add attendee



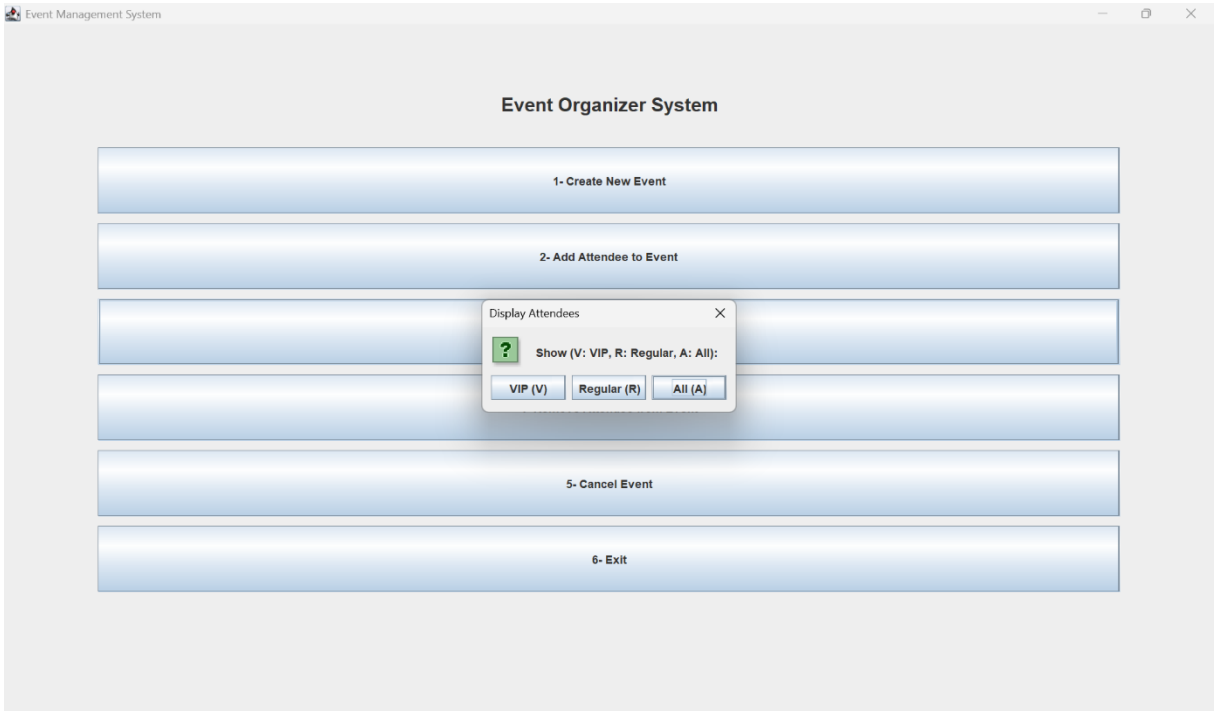


Adding attendee not in the system

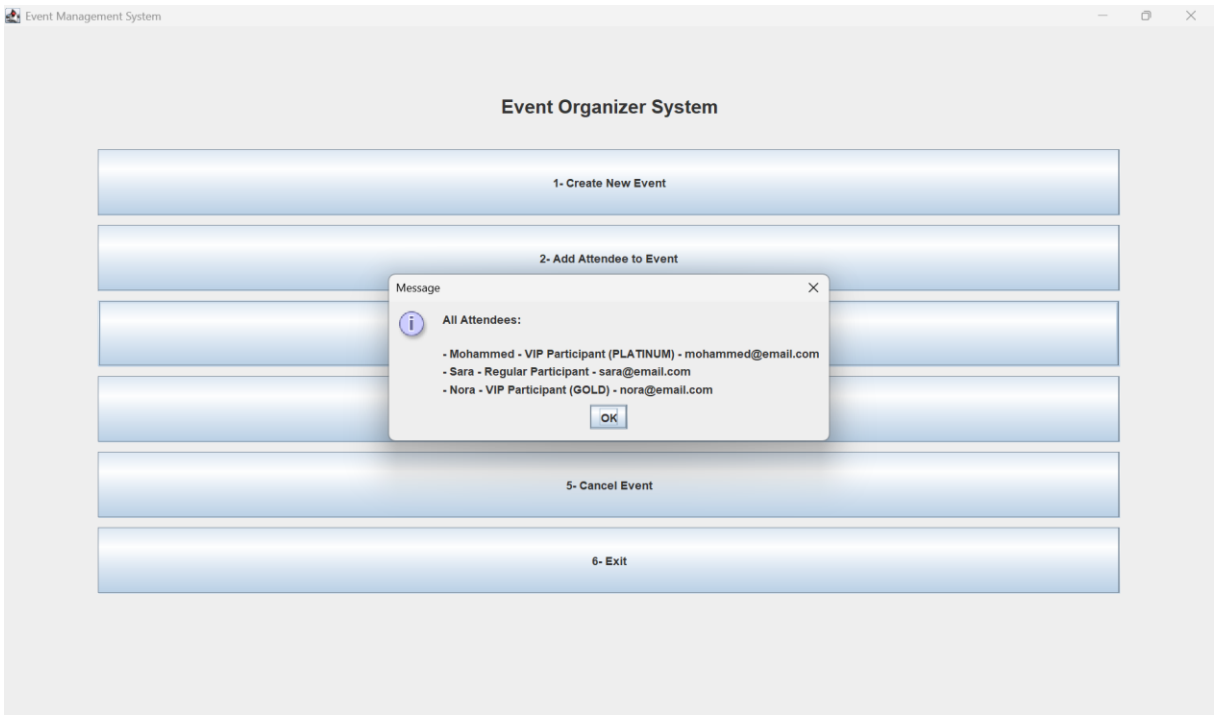




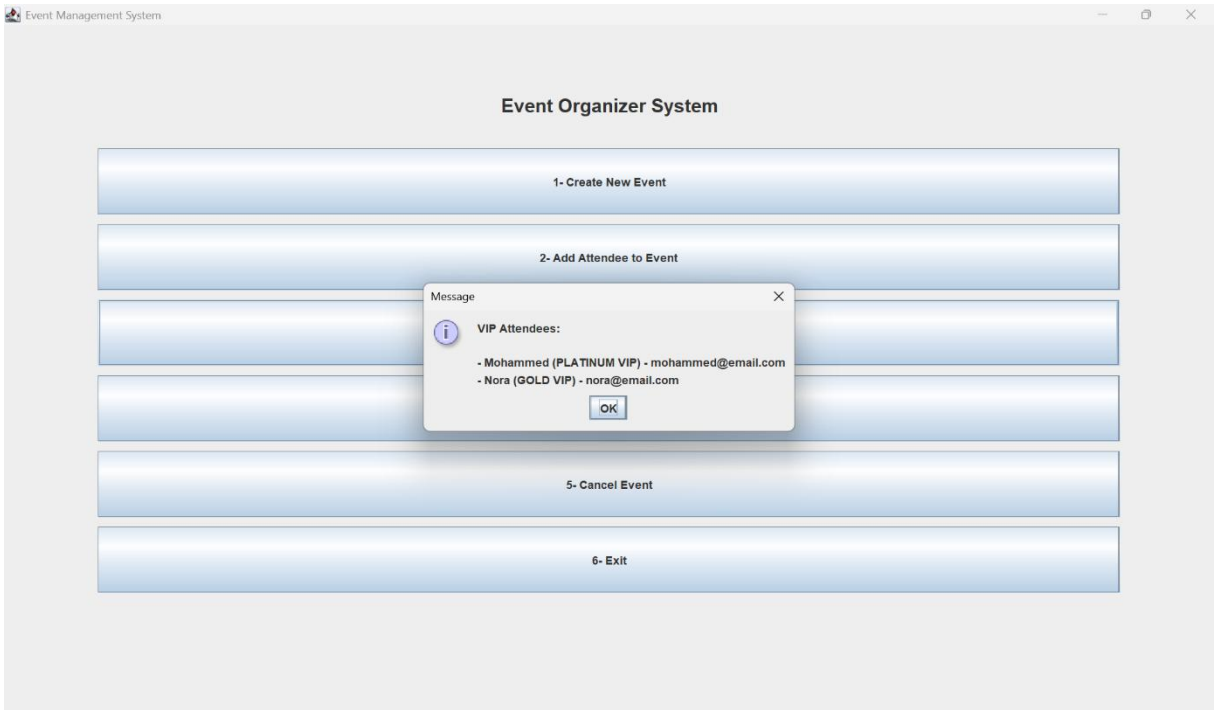
Display attendees



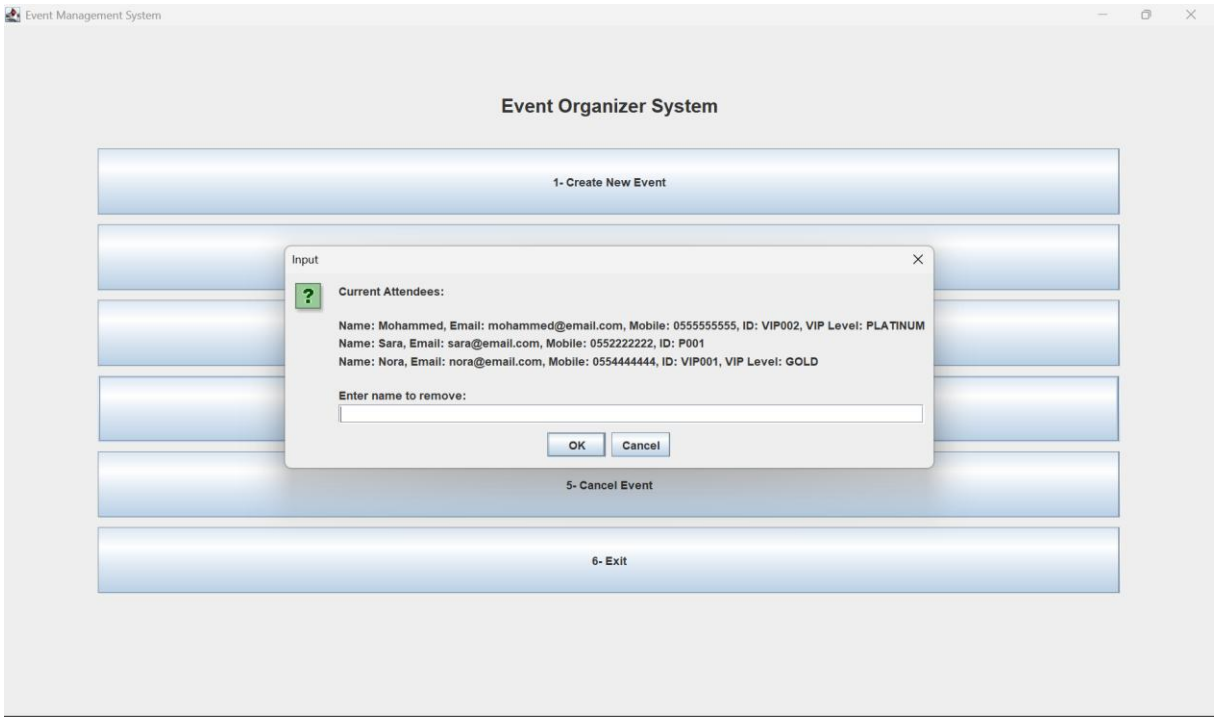
All

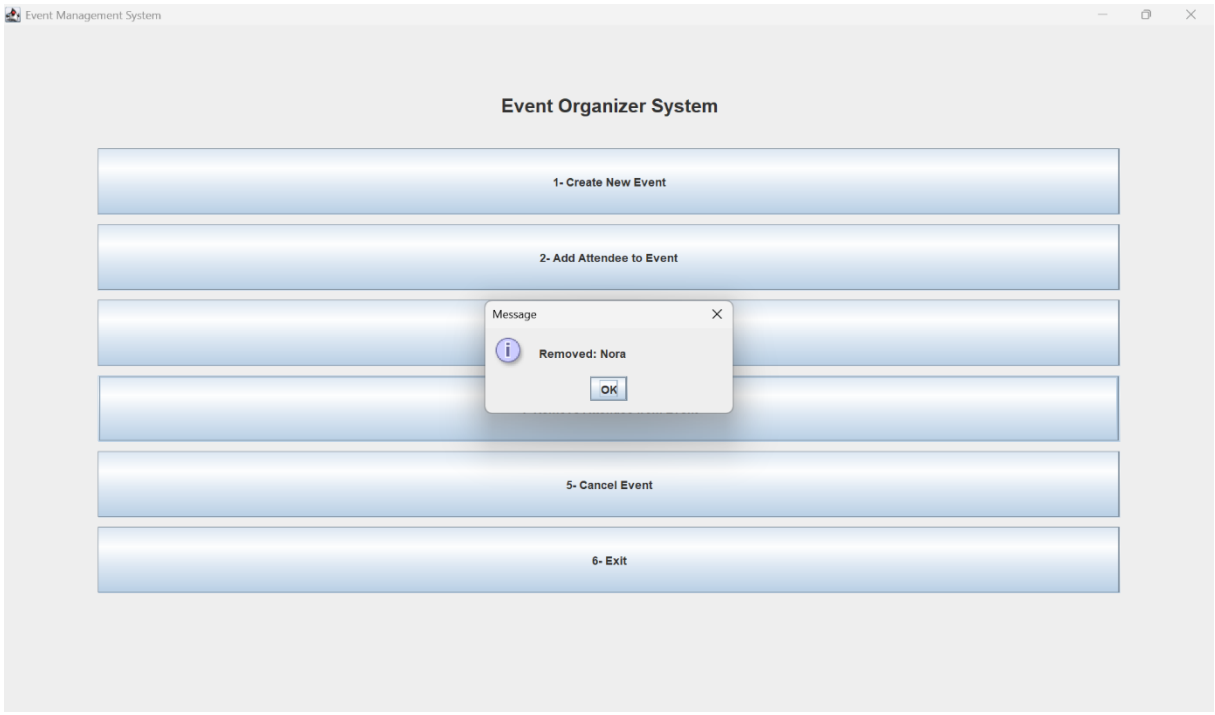


By type VIP

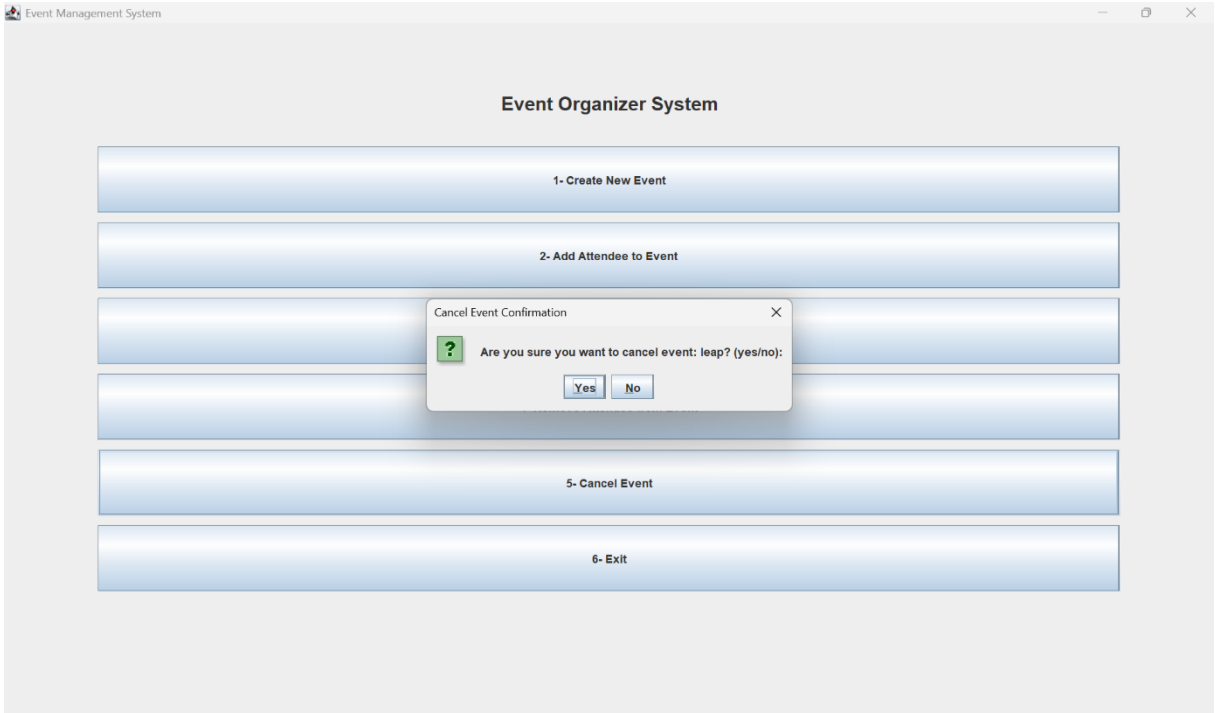


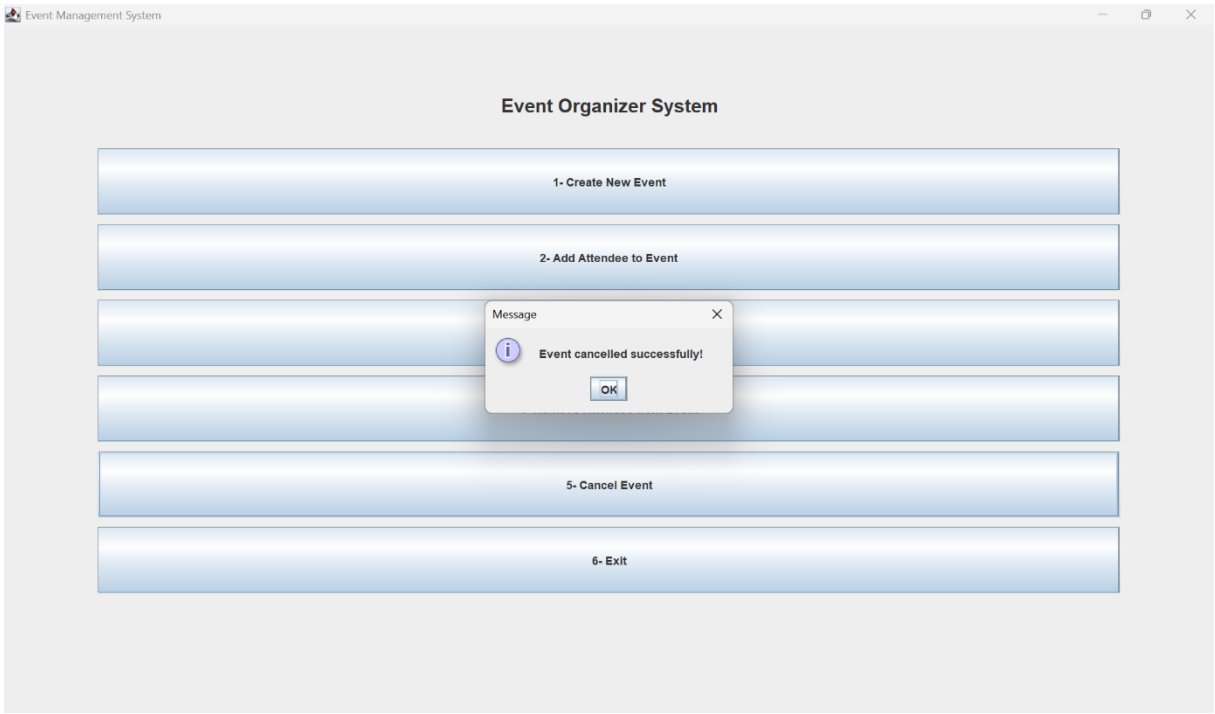
Remove attendee





Cancel event




Event Management System
Event Organizer System
1- Create New Event
2- Add Attendee to Event
Message
Event cancelled successfully!
OK
5- Cancel Event
6- Exit