**UNIVERSITI TEKNOLOGI MALAYSIA**

# OBJECT ORIENTED PROGRAMMING (SECJ2154-04)

# SEMESTER 2 2023/2024

**GROUP PROJECT**

# LOCAL EVENT FINDER

**AHMAD MOHAMMAD MAZEN ALYASIN (A21EC4030)**

**SECTION 04**

**Lecturer:**

**MADAM LIZAWATI MI YUSUF**

**Synopsis:**

The "Local Event Finder" technology assists users in locating and monitoring local events that align with their interests. Users have the option to register as either normal or premium members. The system enables users to do event searches based on parameters such as title, date, or location, and thereafter delivers alerts on events that align with their interests. The system is capable of supporting both online and offline events, making it a flexible tool for discovering events.

**Objective and Scope:**

The main goal of the "Local Event Finder" method is to make it easier for people to find events in their area.

• Make it easy to look for and sort events through an easy-to-use interface.
• Let people sign up and take care of their pages.
• Let people know about events that are coming up.
• Help with different kinds of events, both online and off

**Work Flow:**

1. Registration of Users:
   Users can become normal or paid members when they sign up.
2. Making an event:
   You can add events and give them a title, a description, a date, and a place.
3. Event Search:
   People can use options like title, date, or place to look for events.
4. Notifications:
   Users are told about events that interest them.

Viewing Events: Users can see a lot of information about events that are happening online or off.

**OO Concepts:**

- **Inheritance:** Differentiates between regular and premium users, and between online and offline events.
- **Encapsulation:** Protects user and event data within their respective classes.
- **Polymorphism:** Manages different event types (online and offline) with uniform interfaces.
- **Abstraction:** Hides complex event search and notification logic behind simple interfaces.

**Screens:**

Main Screen: Shows choices for logging in or signing up. Users can look for events, see information about events, and change how alerts work after logging in.

Search Screen: This screen lets people look for events by title, date, or place. Finding events is easy with this screen's simple layout.

The Event Details Screen displays in-depth details about a chosen event, such as its title, description, date, place, and, for online events, a link to the event itself.

**Section B: Class Diagrams**

**Class Diagram for Local Event Finder System**

**Class User:**

| Attributes | Description |
|---|---|
| userId | Unique identifier for the user. |
| name | User's name |
| email | User's email Address |
| password | User's Password |
| **Methods** | **Description** |
| getUserId | Retrieves the user ID. |
| getName | Retrieves the user's name. |
| getEmail | Retrieves the user's email. |
| checkPassword | Verifies the user's password. |
| updatePassword | Changes the user's password. |

**Class PremiumUser extends User:**

| Attributes | Description |
|---|---|
| premiumFeaturesEnabled | Indicates if premium features are active. |
| **Methods** | **Description** |
| hasPremiumFeatures | Checks if premium features are active. |

**Class Event:**

| Attributes | Description |
|---|---|
| eventId | Unique identifier for the event. |
| title | Event title. |
| description | Event description. |
| location | Event location. |
| date | Event date. |
| **Methods** | **Description** |
| getEventId | Retrieves the event ID. |

| | |
|---|---|
| getTitle | Retrieves the event title. |
| getDescription | Retrieves the event description. |
| getDate | Retrieves the event date. |
| getLocation | Retrieves the event location. |

Class OnlineEvent extends Event:

| Attributes | Description |
|---|---|
| link | URL for the online event. |
| **Methods** | **Description** |
| getLink | Retrieves the URL for the online event. |

Class OfflineEvent extends Event:

| Attributes | Description |
|---|---|
| venue | Venue for the offline event. |
| **Methods** | **Description** |
| getVenue | Retrieves the venue for the offline event. |

Class EventSearcher:

| Attributes | Description |
|---|---|
| events | List of events to search through. |
| **Methods** | **Description** |
| addEvent | Adds an event to the list. |
| searchByTitle | Finds events by title. |
| searchByDate | Finds events by date. |
| searchByLocation | Finds events by location. |

Class Notification:

| Attributes | Description |
|---|---|
| notificationId | Unique identifier for the notification. |
| userId | ID of the user receiving the notification. |
| message | Notification message. |
| date | Date of the notification. |
| **Methods** | **Description** |
| getNotificationId | Retrieves the notification ID. |
| getUserId | Retrieves the user ID receiving the notification. |
| getMessage | Retrieves the notification message. |

| getDate | Retrieves the date of the notification. |
| --- | --- |

# Section C: Source Code and User Manual

UML Diagram:
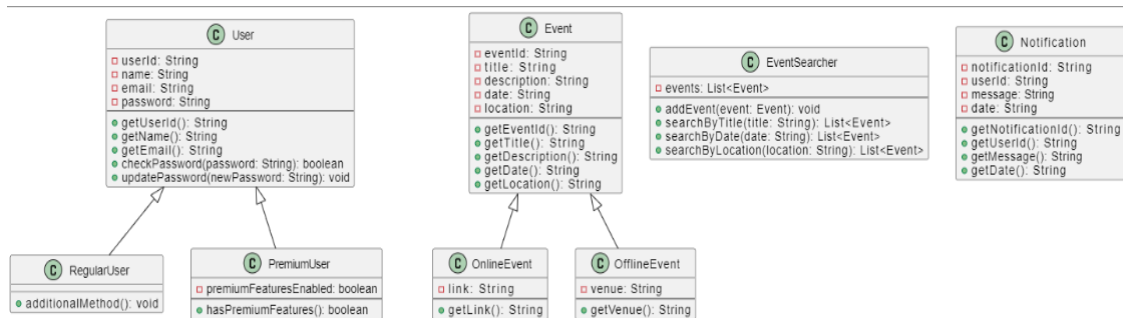


Flowchart:
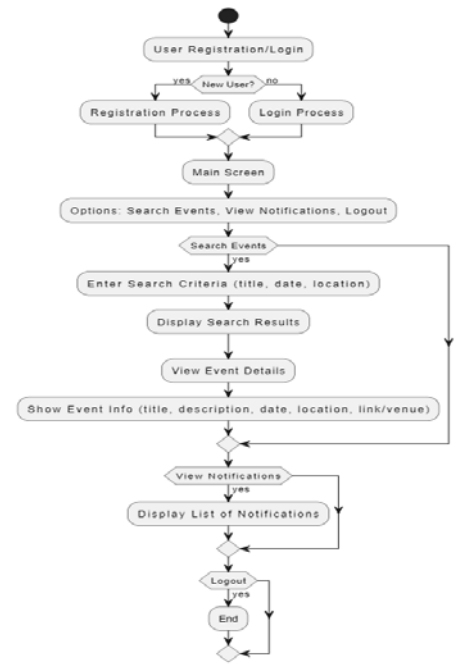
**Main Screen**: Users can sign in or join on the main screen. Users can look for events, see information about events, and change how alerts work after logging in. After the person successfully gets in, this screen shows up.

**Search Screen:** The search screen lets people look for events by topic, date, and place, among other things. The goal of this screen is to make it easy for users to find events that interest them.

**Event Details Screen:** This screen shows detailed information about an event that has been chosen. It has the event's title, summary, date, place, and, if it's an online event, a link to the event. When a person chooses an event from the search results, this screen shows up.

Source Code:

```java
package user_management;

public class User {
    private String userId;
    private String name;
    private String email;
    private String password;

    public User(String userId, String name, String email, String password) {
        this.userId = userId;
        this.name = name;
        this.email = email;
        this.password = password;
    }

    public String getUserId() {
        return userId;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public boolean checkPassword(String password) {
        return this.password.equals(password);
    }
}
```

```java
public class User {
    }

    public void updatePassword(String newPassword) {
        this.password = newPassword;
    }
}

// RegularUser.java
package user_management;

public class RegularUser extends User {
    public RegularUser(String userId, String name, String email, String password) {
        super(userId, name, email, password);
    }
}

// PremiumUser.java
package user_management;

public class PremiumUser extends User {
    private boolean premiumFeaturesEnabled;

    public PremiumUser(String userId, String name, String email, String password) {
        super(userId, name, email, password);
        this.premiumFeaturesEnabled = true;
    }

    public boolean hasPremiumFeatures() {
        return premiumFeaturesEnabled;
    }
}
```

```java
package event_management;

public class Event {
    private String eventId;
    private String title;
    private String description;
    private String date;
    private String location;

    public Event(String eventId, String title, String description, String date, String location) {
        this.eventId = eventId;
        this.title = title;
        this.description = description;
        this.date = date;
        this.location = location;
    }

    public String getEventId() {
        return eventId;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getDate() {
        return date;
```

```java
    public String getDate() {
        return date;
    }

    public String getLocation() {
        return location;
    }
}

// OnlineEvent.java
package event_management;

public class OnlineEvent extends Event {
    private String link;

    public OnlineEvent(String eventId, String title, String description, String date, String location, String link) {
        super(eventId, title, description, date, location);
        this.link = link;
    }

    public String getLink() {
        return link;
    }
}

// OfflineEvent.java
package event_management;

public class OfflineEvent extends Event {
    private String venue;
```

```java
package event_management;

public class OfflineEvent extends Event {
    private String venue;

    public OfflineEvent(String eventId, String title, String description, String date, String location, String venue) {
        super(eventId, title, description, date, location);
        this.venue = venue;
    }

    public String getVenue() {
        return venue;
    }
}

// EventSearcher.java
package search_and_filter;

import event_management.Event;
import java.util.ArrayList;
import java.util.List;

public class EventSearcher {
    private List<Event> events;

    public EventSearcher() {
        events = new ArrayList<>();
    }

    public void addEvent(Event event) {
        events.add(event);
```

```java
public class EventSearcher {

    public List<Event> searchByTitle(String title) {
        List<Event> result = new ArrayList<>();
        for (Event event : events) {
            if (event.getTitle().equalsIgnoreCase(title)) {
                result.add(event);
            }
        }
        return result;
    }

    public List<Event> searchByDate(String date) {
        List<Event> result = new ArrayList<>();
        for (Event event : events) {
            if (event.getDate().equalsIgnoreCase(date)) {
                result.add(event);
            }
        }
        return result;
    }

    public List<Event> searchByLocation(String location) {
        List<Event> result = new ArrayList<>();
        for (Event event : events) {
            if (event.getLocation().equalsIgnoreCase(location)) {
                result.add(event);
            }
        }
        return result;
    }
}
```

```java
public class Notification {
    private String notificationId;
    private String userId;
    private String message;
    private String date;

    public Notification(String notificationId, String userId, String message, String date) {
        this.notificationId = notificationId;
        this.userId = userId;
        this.message = message;
        this.date = date;
    }

    public String getNotificationId() {
        return notificationId;
    }

    public String getUserId() {
        return userId;
    }

    public String getMessage() {
        return message;
    }

    public String getDate() {
        return date;
    }
}
```