Zewail City of Science and Technology مدينة زويل للعلوم والتكنولوجيا

CIE203

Software Engineering

EventeX

Software Design Document

TXLM

Month & Year

CIE 203: Phase 1 – TXLM
Project: <EventeX>

Zewail City for Science and Technology

Software Design Document

Contents

CIE 203: Phase 1 – TXLM
Project: <EventeX>

Zewail City for Science and Technology

Software Design Document

Team

ID	Name	Email	Mobile
201305277	Alhasan Abdellatif	s-al-hassan.abdel-latif@zewailcity.edu.eg	01099796548
201300944	Mahmoud Shaban	S-Mahmoud.Almaz@zewailcity.edu.eg	01121973212
201304826	Amr Elgendy	s-amr.elgendy@zewailcity.edu.eg	01060617836

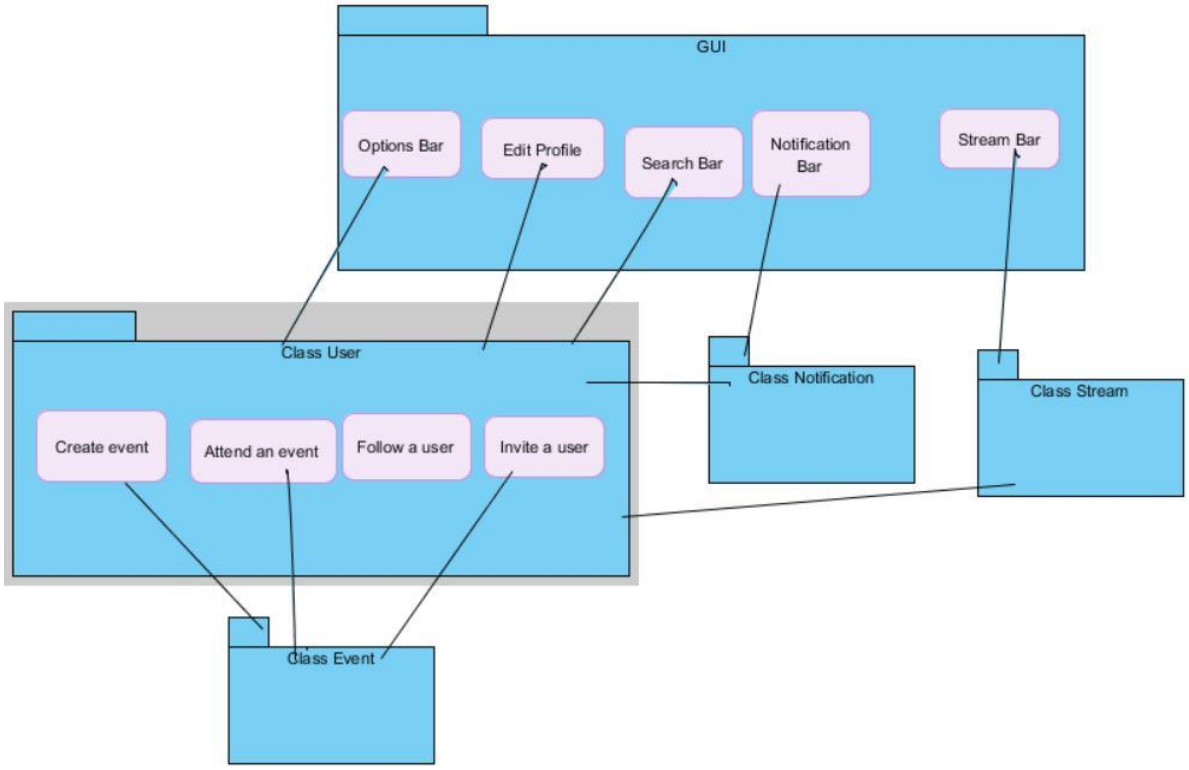
Document Purpose and Audience

This is an initial SDD document of software called EventeX which presents events to users in an interactive way. The document describes the design step of the software before the implementation. The audience to read this document is expected to be CEO's and project managers or any person having a software engineering background and interested in the product.

Software Design Document

System Models

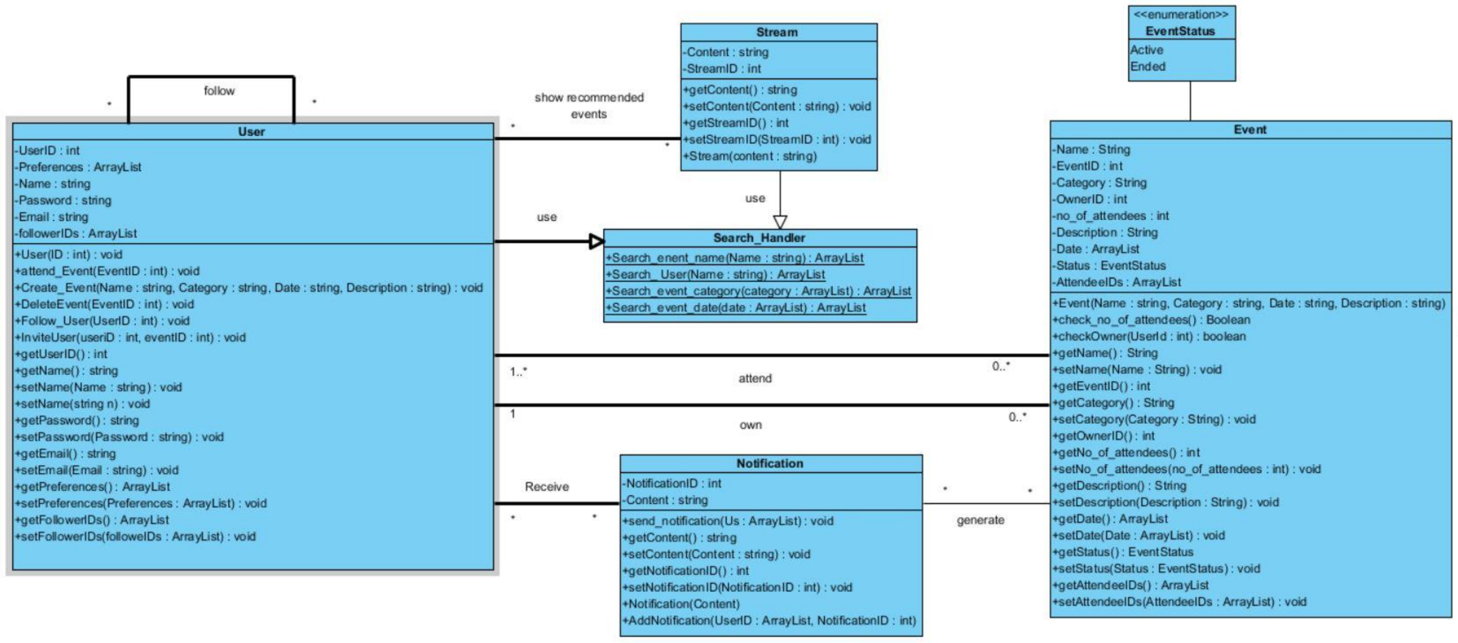
I. System Decomposition



Zewail City for Science and Technology

Software Design Document

II. Class diagrams


Stream
-Content : string
-StreamID : int
+getContent() : string
+setContent(Content : string) : void
+getStreamID() : int
+setStreamID(StreamID : int) : void
+Stream(content : string)
<<enumeration>>
EventStatus
Active
Ended
follow
show recommended events
User
-UserID : int
-Preferences : ArrayList
-Name : string
-Password : string
-Email : string
-followerIDs : ArrayList
+User(ID : int) : void
+attend_Event(EventID : int) : void
+Create_Event(Name : string, Category : string, Date : string, Description : string) : void
+DeleteEvent(EventID : int) : void
+Follow_User(UserID : int) : void
+InviteUser(userID : int, eventID : int) : void
+getUserID() : int
+getName() : string
+setName(Name : string) : void
+setName(string n) : void
+getPassword() : string
+setPassword(Password : string) : void
+getEmail() : string
+setEmail(Email : string) : void
+getPreferences() : ArrayList
+setPreferences(Preferences : ArrayList) : void
+getFollowerIDs() : ArrayList
+setFollowerIDs(followeIDs : ArrayList) : void
use
use
Search_Handler
+Search_enent_name(Name : string) : ArrayList
+Search_User(Name : string) : ArrayList
+Search_event_category(category : ArrayList) : ArrayList
+Search_event_date(date : ArrayList) : ArrayList
Event
-Name : String
-EventID : int
-Category : String
-OwnerID : int
-no_of_attendees : int
-Description : String
-Date : ArrayList
-Status : EventStatus
-AttendeeIDs : ArrayList
+Event(Name : string, Category : string, Date : string, Description : string)
+check_no_of_attendees() : Boolean
+checkOwner(UserId : int) : boolean
+getName() : String
+setName(Name : String) : void
+getEventID() : int
+getCategory() : String
+setCategory(Category : String) : void
+getOwnerID() : int
+getNo_of_attendees() : int
+setNo_of_attendees(no_of_attendees : int) : void
+getDescription() : String
+setDescription(Description : String) : void
+getDate() : ArrayList
+setDate(Date : ArrayList) : void
+getStatus() : EventStatus
+setStatus(Status : EventStatus) : void
+getAttendeeIDs() : ArrayList
+setAttendeeIDs(AttendeeIDs : ArrayList) : void
1..*
attend
0..*
1
own
0..*
Receive
Notification
-NotificationID : int
-Content : string
+send_notification(Us : ArrayList) : void
+getContent() : string
+setContent(Content : string) : void
+getNotificationID() : int
+setNotificationID(NotificationID : int) : void
+Notification(Content)
+AddNotification(UserID : ArrayList, NotificationID : int)
generate

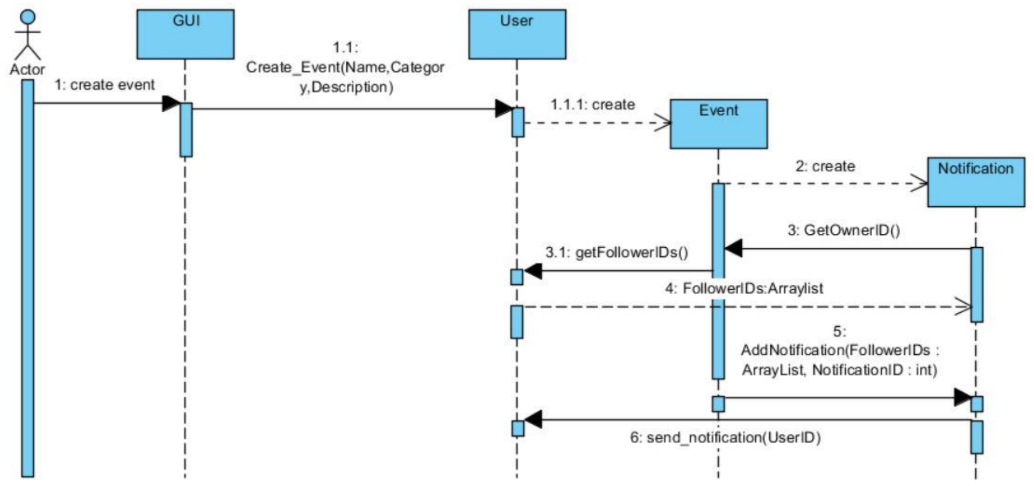
Class ID	Class Name	Subsystem ID	Description & Responsibility
1	User	2	• It allows the user to update his/her profile including changing the preferred categories. • It allows the user to follow another user to get notifications about the activities of the follower. • It allows the user to create, edit, and invite other members to attend his/her events.

Class ID	Class Name	Subsystem ID	Description & Responsibility
2	Search	1	• This class is responsible for allowing the user to search for other users or search for events by name, category or date.
3	Event	3	• This class is responsible for managing only the allowed members (creators of events) to edit the name, category, maximum number of attendees, date and description of the events. • It is also responsible for providing the GUI with the information needed to be presented about each event.
4	Stream	2	• It creates the content that each user can see in his/her stream bar. • It provides this content to the GUI to put it in the stream.
5	Notification	2	• It creates the content that each user can see in his/her notification bar. • It provides this content to the GUI to put it in the notification bar • It also sends these notifications by mail.
6	EventStatus	3	• It specifies the status of of the event either active or ended.

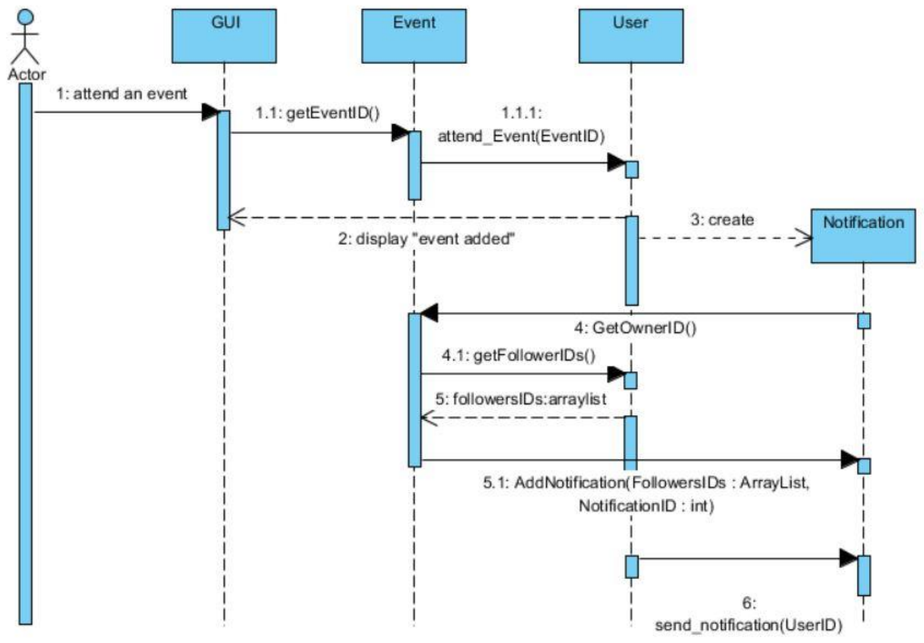
III. Sequence diagrams

We developed 6 main sequence diagrams representing the main use cases in the program.

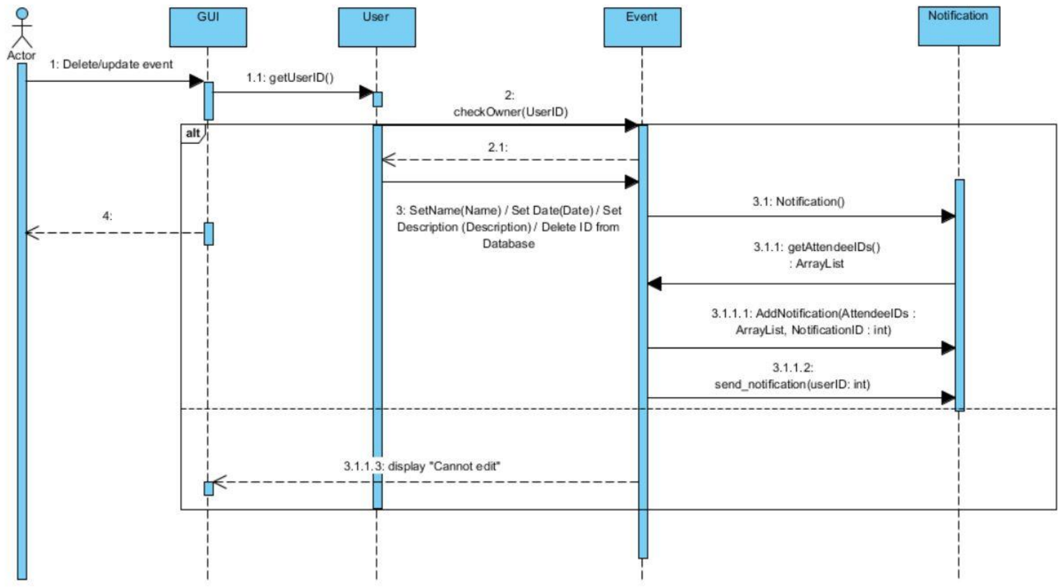
1- Create an Event


Actor
GUI
User
Event
Notification
1: create event
1.1: Create_Event(Name,Category,Description)
1.1.1: create
2: create
3: GetOwnerID()
3.1: getFollowerIDs()
4: FollowerIDs:Arraylist
5: AddNotification(FollowerIDs : ArrayList, NotificationID : int)
6: send_notification(UserID)

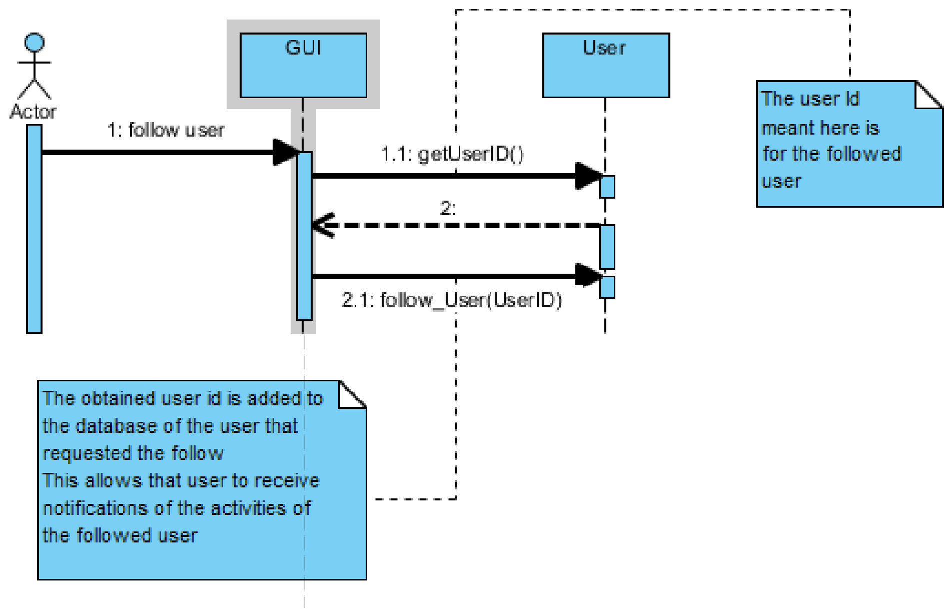
2- Attend an Event


Actor
GUI
Event
User
1: attend an event
1.1: getEventID()
1.1.1: attend_Event(EventID)
3: create
Notification
2: display "event added"
4: GetOwnerID()
4.1: getFollowerIDs()
5: followersIDs:arraylist
5.1: AddNotification(FollowersIDs : ArrayList, NotificationID : int)
6: send_notification(UserID)

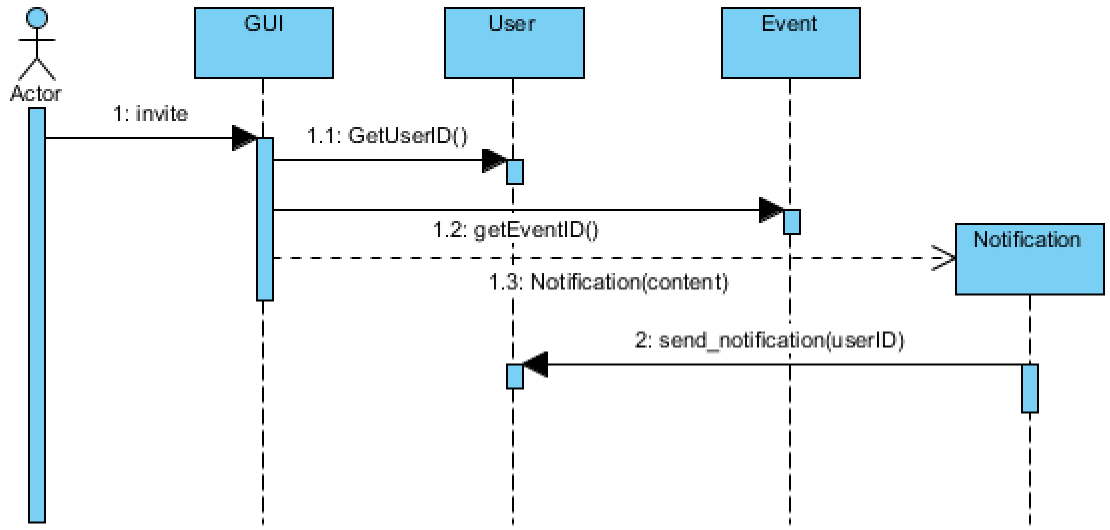
3- Update or delete an event:


Actor
GUI
User
Event
Notification
1: Delete/update event
1.1: getUserID()
2: checkOwner(UserID)
alt
2.1:
4:
3: SetName(Name) / Set Date(Date) / Set Description (Description) / Delete ID from Database
3.1: Notification()
3.1.1: getAttendeeIDs() : ArrayList
3.1.1.1: AddNotification(AttendeeIDs : ArrayList, NotificationID : int)
3.1.1.2: send_notification(userID: int)
3.1.1.3: display "Cannot edit"

4- Follow a user:

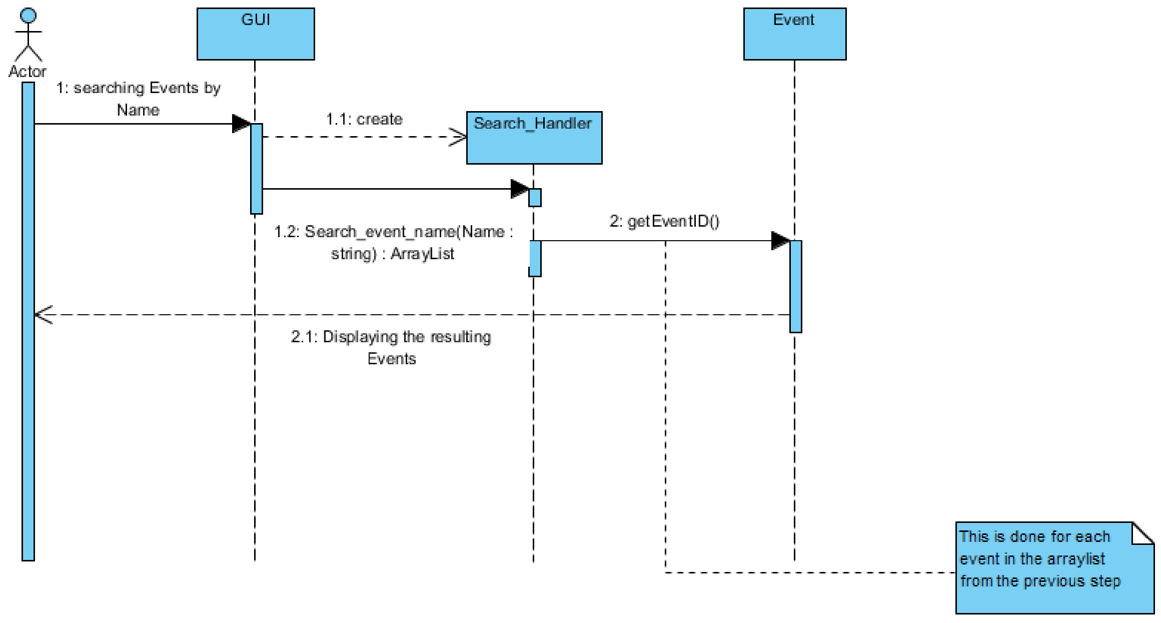

Actor
GUI
User
1: follow user
1.1: getUserID()
2:
2.1: follow_User(UserID)
The user Id meant here is for the followed user
The obtained user id is added to the database of the user that requested the follow
This allows that user to receive notifications of the activities of the followed user

5- Invite a user to an event:



6- Search for event:

Please note that we made sequence diagram for search by name and other options for search follow the same analogy.

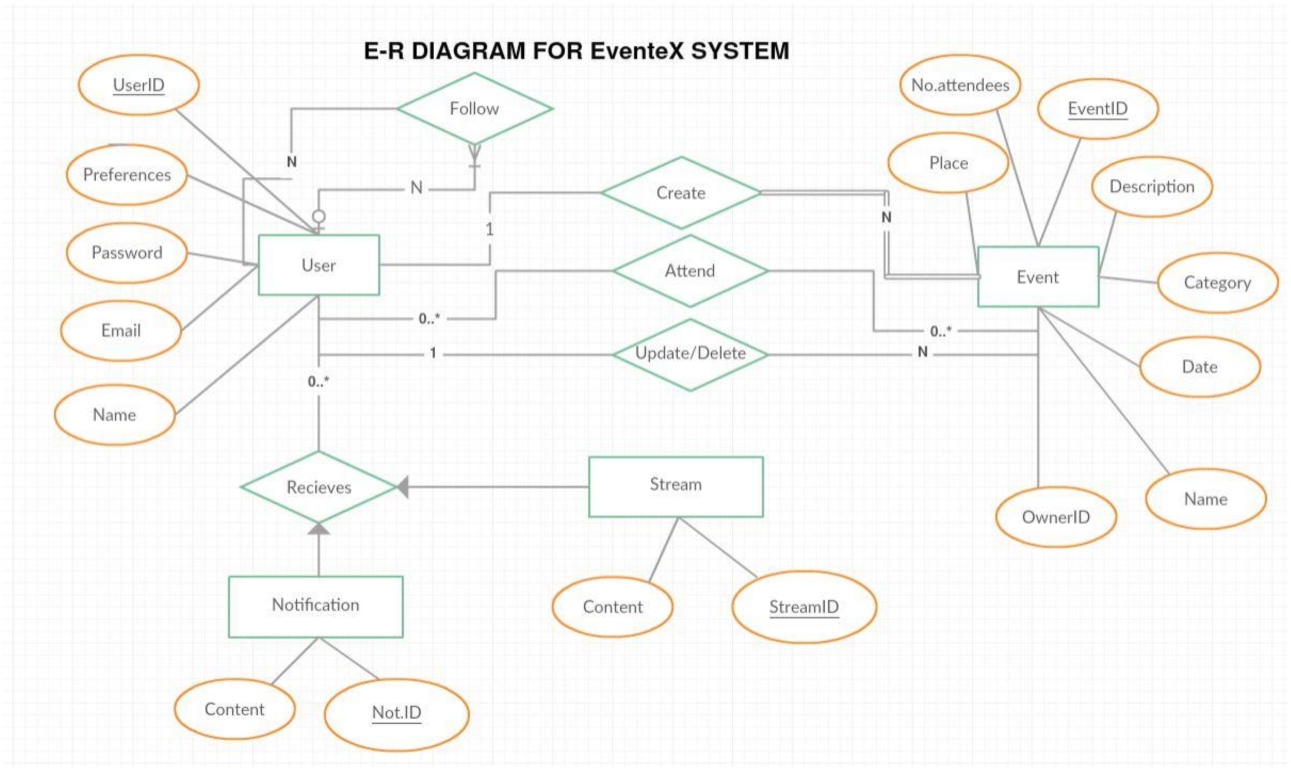

Actor
GUI
Event
Search_Handler
1: searching Events by Name
1.1: create
1.2: Search_event_name(Name : string) : ArrayList
2: getEventID()
2.1: Displaying the resulting Events
This is done for each event in the arraylist from the previous step

Class - Sequence Usage Table

- **In this table, we will list EVERY class in class diagram and which sequences used this class diagram. This helps in avoiding either unused classes or extra classes appears in sequence diagrams. In "Overall used methods" section, put all functions appeared in all sequences.**

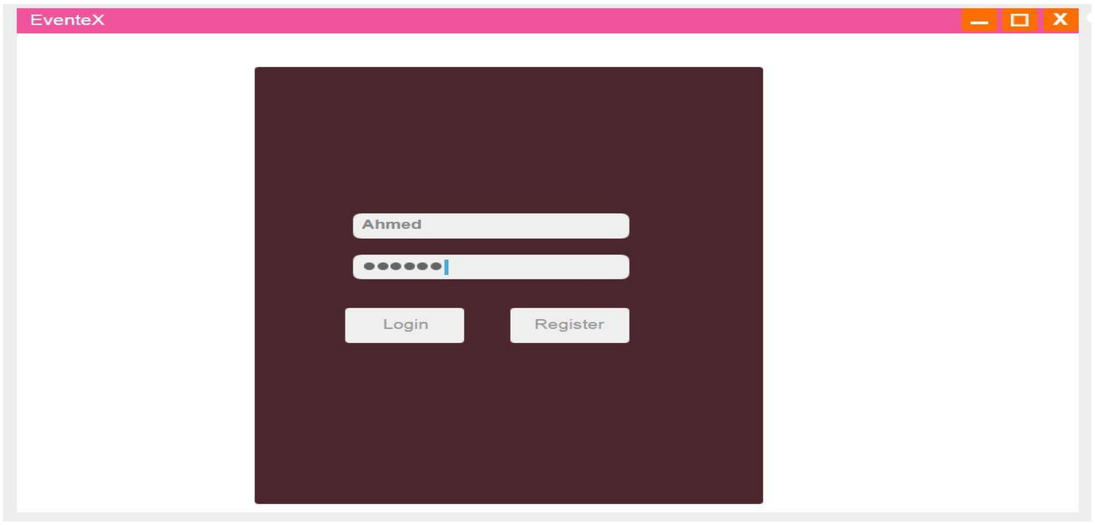
Class Name	Sequence Diagrams	Overall used methods
User	1,2,3,4,5	Create, delete and attend events Follow and invite users
Event	1,2,3,5,6	getownerID , getEventID , set name,description,… check owner
Notification	1,2,3,5	Sendnotification, Addnotification
Search_handler	6	Search_event_name,category,…
Stream	No sequence diagram. However it can be added to new sequence diagram for displaying recommended events, thus it is not trivial	

IV. Physical Entity-Relationship Diagram

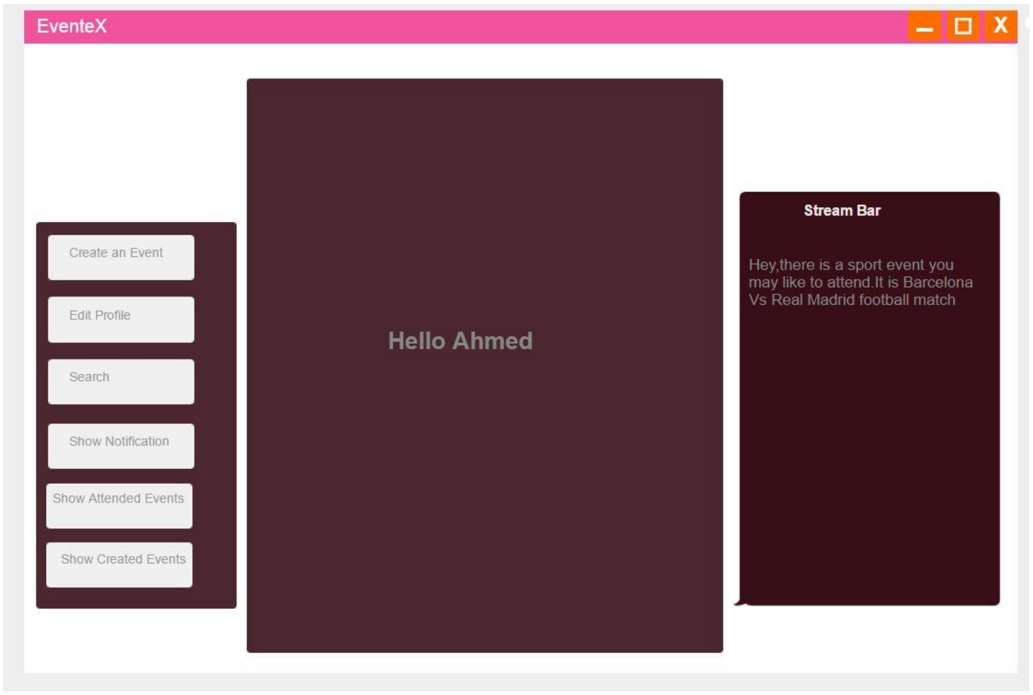


V. User Interface Design

The program begins with login or register window :

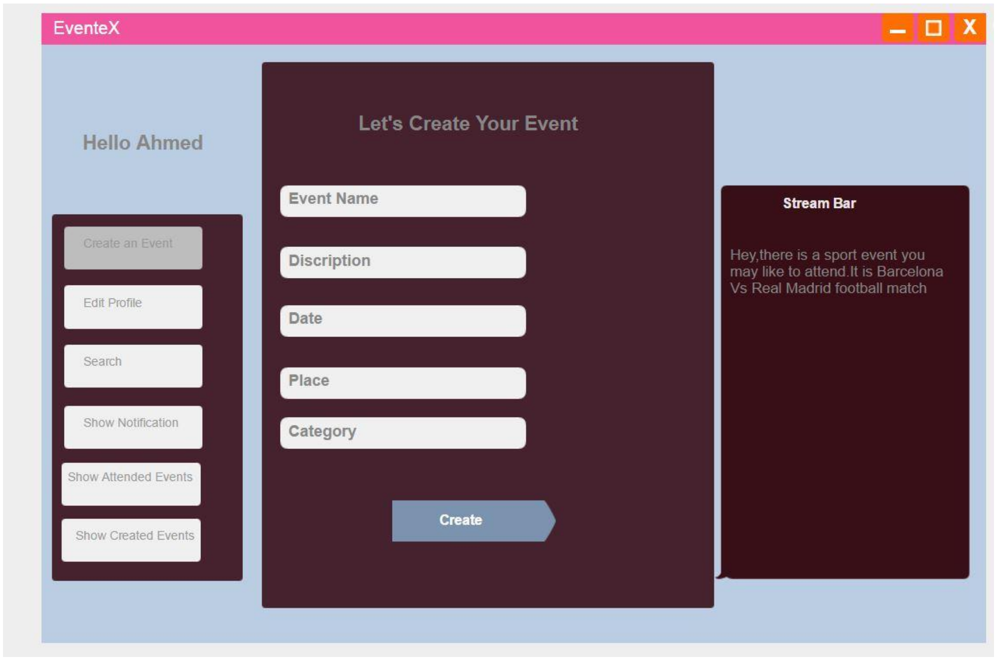


It then opens the main window:



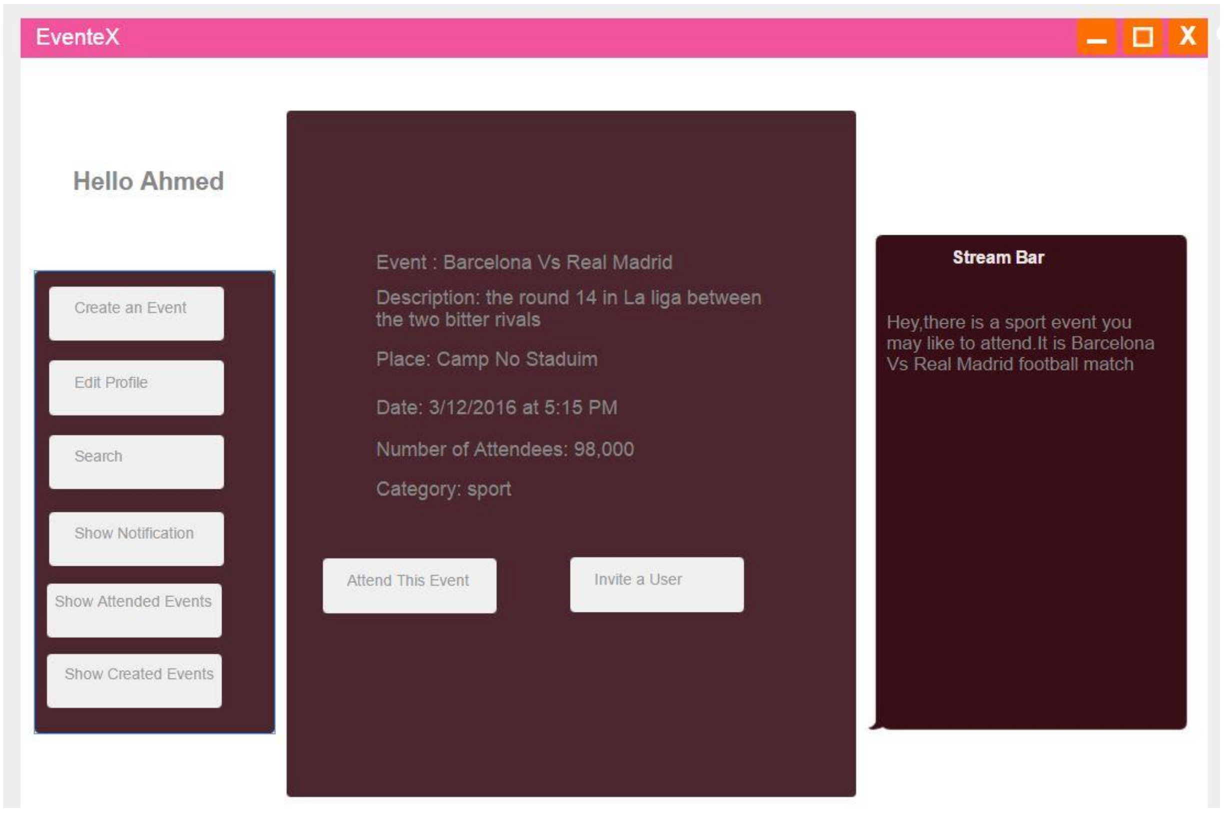
Hey,there is a sport event you may like to attend.It is Barcelona Vs Real Madrid football match

The user can choose any option if he wants to create an event :



Hey,there is a sport event you may like to attend.It is Barcelona Vs Real Madrid football match

If he clicked on the stream appeared on the stream bar:



Event : Barcelona Vs Real Madrid

Description: the round 14 in La liga between the two bitter rivals

Place: Camp No Staduim

Date: 3/12/2016 at 5:15 PM

Number of Attendees: 98,000

Category: sport

Hey,there is a sport event you may like to attend.It is Barcelona Vs Real Madrid football match

VI. Algorithms and Data Structures

Algorithms:

We described algorithms in terms of pseudo code explained below.

1- Main Algorithm:

```
While (true) {
Display a message to ask the user to log in or register if he is not a member
If (he clicked Log In)
```

```
        Call the method Log_In().
        Break the while loop.
If (he clicked Register)
        Call the method Register().
        Break the while loop.
}
While (true) {
If (the user clicked create an event)
        Call the method Create_Event() from the class User.
        Store the description of this action in the notification database with the related UserIDs.
If (the user clicked Delete an event)
        Call the method Delete_Event() from the class User.
        Store the description of this action in the notification database with the related UserIDs.
If (the user clicked attend an event)
        Call the method attend_Event() from the class User.
        Store the description of this action in the notification database with the related UserIDs.
If (the user clicked follow a user)
        Call the method Follow_User () from the class User.
        Store the description of this action in the notification database with the related UserIDs.
If (the user clicked edit my name)
        Call the method setName () from the class User.
        Store the description of this action in the notification database with the related UserIDs.
```

```
If (the user clicked edit my preferences)
    Call the method setPreferences() from the class User.
If (the user clicked Search for an event)
    Ask the user to choose how he wants to search for the event
    If (he chooses by name)
        Call the method Search_event_name() from the class Search_Handler.
    If (he chooses by category)
        Call the method Search_event_name() from the class Search_Handler.
    If (he chooses by date)
        Call the method Search_event_name() from the class Search_Handler.
If (the user clicked Search for a user)
    Call the method Search_User() from the class Search_Handler.
If (the user clicked the stream tab)
    Call the method getStreamID() from the Class Stream.
    Put the ID in the method getContent() from the Class Stream.
If (the user clicked the notification tab)
    Call the method getNotificationID() from the Class Notification.
    Put the ID in the method getContent() from the Class Notification.
    Call the method Set_notification() from the Class Notification.
    Call the method Send_notification() from the Class Notification.
```

2- Specific Algorithms:

```
Register()

Show fields to enter the Username, the Password, the Name and the preferences

If (the Username is unique)

    Say congratulation for registration.

    Break.

Else

    Ask the user to use another Username.

    Call the method Register() from the class Visitor.


Log_in()

Show fields to enter the Username, the Password.

If (the information entered is in the database)

    Break.

Else

Ask the user to enter the correct Username and Password.

Call the method Log_In() from the class User.

Display the Registration button in case the user wants to register.
```

Class User:

```
Set the Attributes: UserID, Password, Name, Email and preferences.

 Create_Event()

    create an object in the class Event with the parameters: name, category, and date.

    Store it in the related data base.
```

Delete_Event()

```
show the user`s events and ask him/her to check the event he/she wants to delete.
Send a confirmation message for the deletion process.
If (the user says confirm)
    Delete the event from the database and break.
else
    break
```

attend_Event()

Save the user in the database of the event`s attendees.

Follow_User()

Save the follower in the database of the followed list.

Save the followed user in the database of the user`s following list.

InviteUser()

Record a notification in the notifications database with the ID of the invited user.

getUserID()

return the ID of a selected user.

getName()

return the name of a selected user.

setName()

show a field to change the name of the user.

getPassword()

return the password of a selected user.

setPassword()

```
show two fields as a confirmation to change the password of the user.
If (the password in the two fields are the same)
    End the method
Else Ask the user to write the password again
```

getEmail()

return the email of the user.

setEmail()

show a field to change the email of the user.
Check the database if this email is used.
If (used) : ask the user to use another email.

getPreferences()

show all the categories with the preferred ones by the user checked.

setPrefernces()

show check boxes beside each category to check or uncheck.

Class Event:

set the attributes: name, EventID, Category, ownerID, description, status, no_of_attendees, and date.

check_no_of_attendees ()

```
Count the number.
If (the number of attendees is changed)
    Return true

Else: return false
```

checkOwner()

check the database of the user if this event exists among his events.

If (exists): return true

If (not): return false

getName()

return the name of a selected event.

setName()

show a field to change the name of the event.

getEventID()

return the ID of a selected event.

getCategory()

return the category of a selected event.

setCategory()

show all the categories and allow the user to set the category of the event.

getOwnerID()

return the ID of the event`s owner.

getNo_of_attendees ()

return the number of attendees.

setNo_of_attendees ()

show a field to set the maximum number of allowed attendees.

setDescription()

show a text field to write a description of the event.

getDescription()

show a text with description of the event.

getDate()

return the date of the event.

setDate()

show three fields to set the day, month and year of the event.

Class Search_Handler

Search_event_name()

A field is displayed and the user is asked to search the event by the name.
The results are shown after searching the similarities in the database.

Search_event_category()

A field is displayed and the user is asked to search the event by the category.
The results are shown after searching the similarities in the database.

Search_event_date()

three fields (year, month and day) are displayed and the user is asked to search the event by the date.
The results are shown after searching the similarities in the database.

Search_User()

A field is displayed and the user is asked to search the user by the name.
The results are shown after searching the similarities in the database.

Class Stream:

getContent()
return the database of the events to be put it in the stream of the user.

setContent()
get the preferences of the user from the method **getPreferences()** from the class **User**.
Use the method **Search_event_category()** from the class **Search** and get the a list of event IDs for each category.
Get the first event in each category then the second and so on, and store them in a content list.

getStreamID()
return the ID of the user.

setStreamID()
set the ID of the user.

Class Notification

Set the attributes: NotificationID and Content

getContent()
return the database of the events to be put it in the notification of the user.

setContent()
search the notification database for the id of the user.
Retrieve the notifications for that user and store it in a content list.

getNotificationID()
return the ID of the user.

setNotificationID()
set the ID of the user.

Set_notification()

Show the newest events first in the notification bar.

Send_notification()

Retrieve the notifications from the **getContent()** method and send them by email to the user.

Data Structures:

1- Stacks: we will use stack to store events because of its nature of retrieving items benefits our application. For Example, In Search handling, when we search for events, the first event to be showed is the newest one that was added. Hence, providing the user with latest events first when doing search and in the stream.

2- Queues: we use queues in association with sending notifications by email, thus we send notifications by mail in order (oldest first).

In both cases we will implement them using arraylist.

Ownership Report

Item	Owners
System Decomposition	*All members of the Team*
Class Diagram	*All members of the Team*
Sequence Diagram	*Amr Elgendy*
ER diagram	*Alhasan Abdellatif*
User Interface	*Alhasan Abdellatif*
Class responsibilities	*Mahmoud Shaban*
Algorithms and data structure	*Mahmoud Shaban*

Policy Regarding Plagiarism:

Students have collective ownership and responsibility of their project. Any violation of academic honesty will have severe consequences and punishment for ALL team members.

1. تشجع الكلية على مناقشة الأفكار و تبادل المعلومات و مناقشات الطلاب حيث يعتبر هذا جوهريا لعملية تعليمية سليمة
2. ساعد زملاءك على قدر ما تستطيع و حل لهم مشاكلهم فى الكود و لكن تبادل الحلول غير مقبول و يعتبر غشا.
3. أى حل يتشابه مع أى حل آخر بدرجة تقطع بأنهما منقولان من نفس المصدر سيعتبر أن صاحبيهما قد قاما بالغش.
4. قد توجد على النت برامج مشابهة لما نكتبه هنا أى نسخ من على النت يعتبر غشا يحاسب عليه صاحبه.
5. إذا لم تكن متأكدا أن فعلا ما يعد غشا فلتسأل المعيد أو أستاذ المادة.
6. فى حالة ثبوت الغش سيأخذ الطالب سالب درجة المسألة ، و فى حالة تكرار الغش سيرسب الطالب فى المقرر.

References

- http://www.mhhe.com/engcs/compsci/pressman/graphics/Pressman5sepa/common/cs1/design.pdf

Authors

- Mostafa Saad and Mohammad El-Ramly