Software Design Specification

➢ Purpose

The purpose of this document is to give to our team members and all developers around the world a visual perspective of the workflow and interaction among various components of our **Event Manager System**.

➢ Scope

This SDS (software design specification) document is centered on the design and structure aspect of boundaries, controls and back-end processes.

Here implementation and testing will not be emphasized, but however some illustration of implementation may be seen.

➢ Definitions, acronyms and abbreviation

Items	Types	Description
UI	Acronym	User Interface
Auth	Abbreviation	Authentication
DB_Manager	Abbreviation	Database Manager

GUI	Acronym	Graphical User Interface
Info	Abbreviation	Information

✓ Sequence diagrams

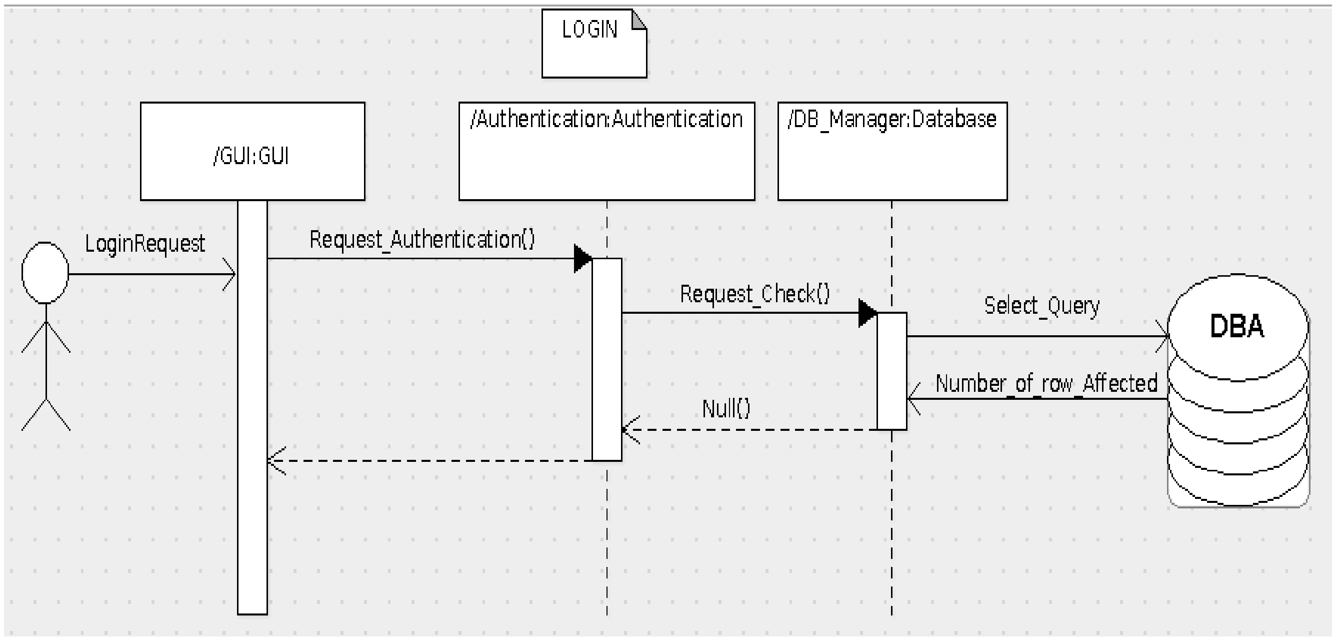
Login

The login process of Event Manager is describe as follow:

The user types his username and password through the GUI, which will create and **Authentication** object.

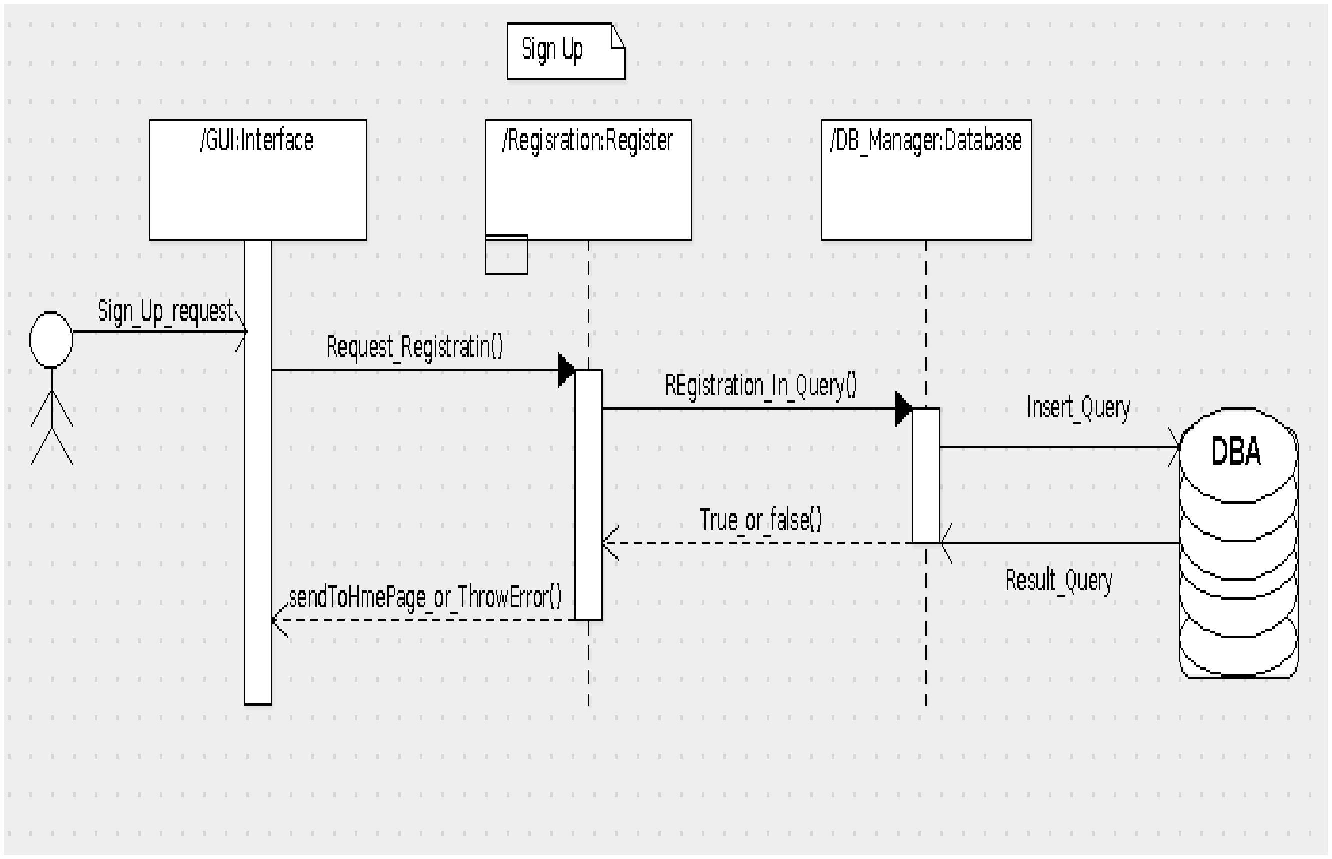
Authentication class will create a **DB_Manager object** by sending a requestCheck.

From there DB_Manager object will finally make selectQuery to the database and return a **User** object back to the **Authentication** which will check if the result is null then throw and exception otherwise relay the **User** info.



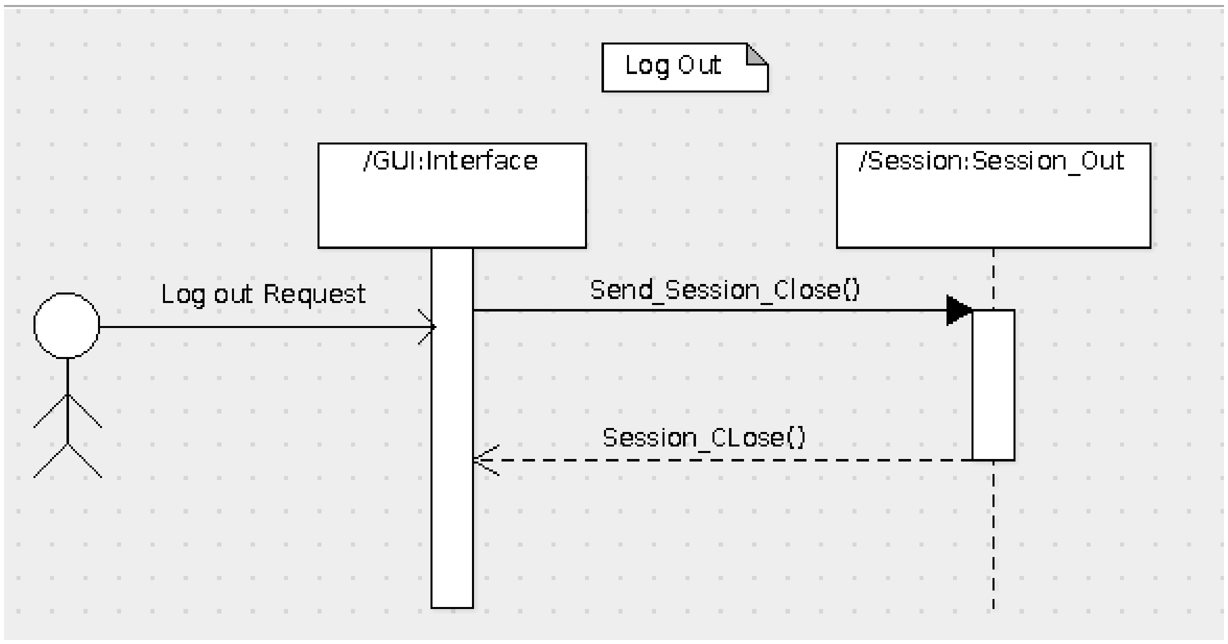
Sign up

As for the Signup process. The user will enter his personal details into the appropriate fields provided by the GUI (First Name, Last Name, email, e.t.c). The GUI then creates a Registration object via the **Request_Registration()** function. When this is done the **Registration** class further creates a **DB_Manager object by** making a **Registration_Inquiry()** and the **DB_Manager** finally sends an **Insert_Query()** to the **Database** which responds with a Boolean *true/false* and the number of rows affected. The **DB_Manager class** handles this **Database response** accordingly by creating a **User object** and sending it back to the **Registration** and finally to the **GUI** in case of a Boolean *true*, or by sending a null value back to the **GUI** through the **Registration** in the case of a Boolean *false*.



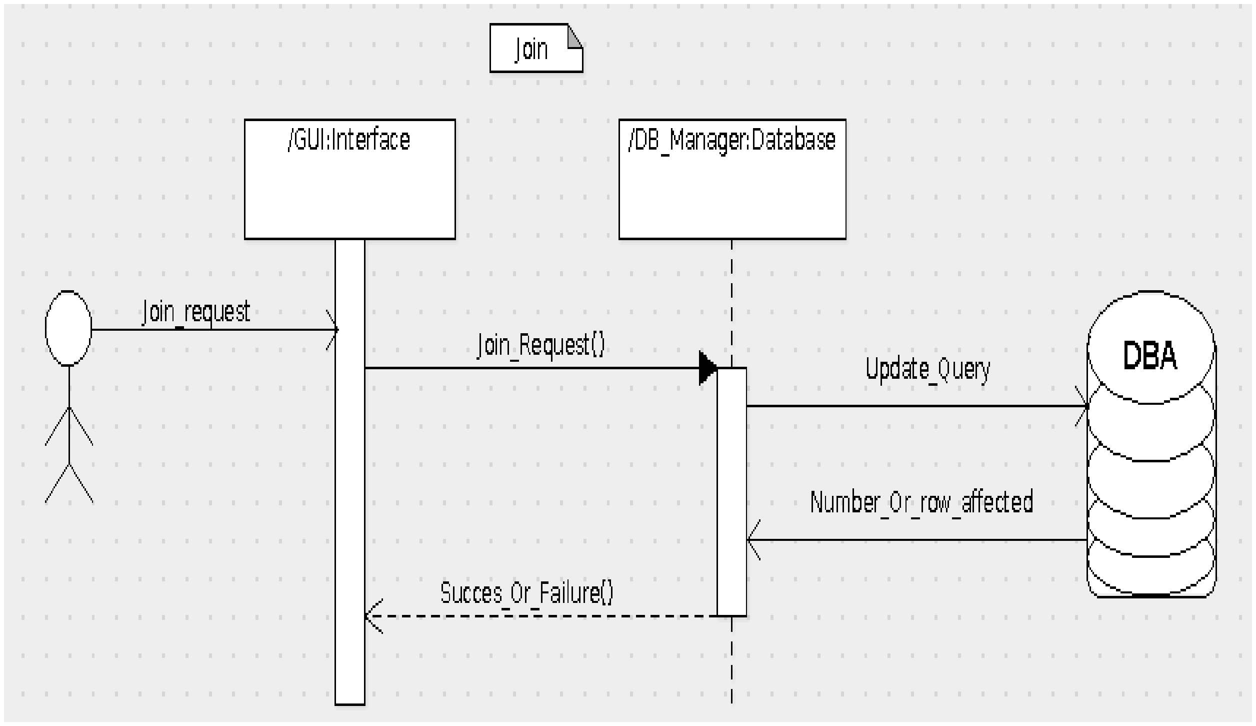
Logout

In the Logout sequence, the user simply interacts with a *Log out* button provided by the **GUI**. From here the **GUI** creates a **Session object** via a **Send_Session_Close()** function which returns a *Session close* back to the **GUI**.



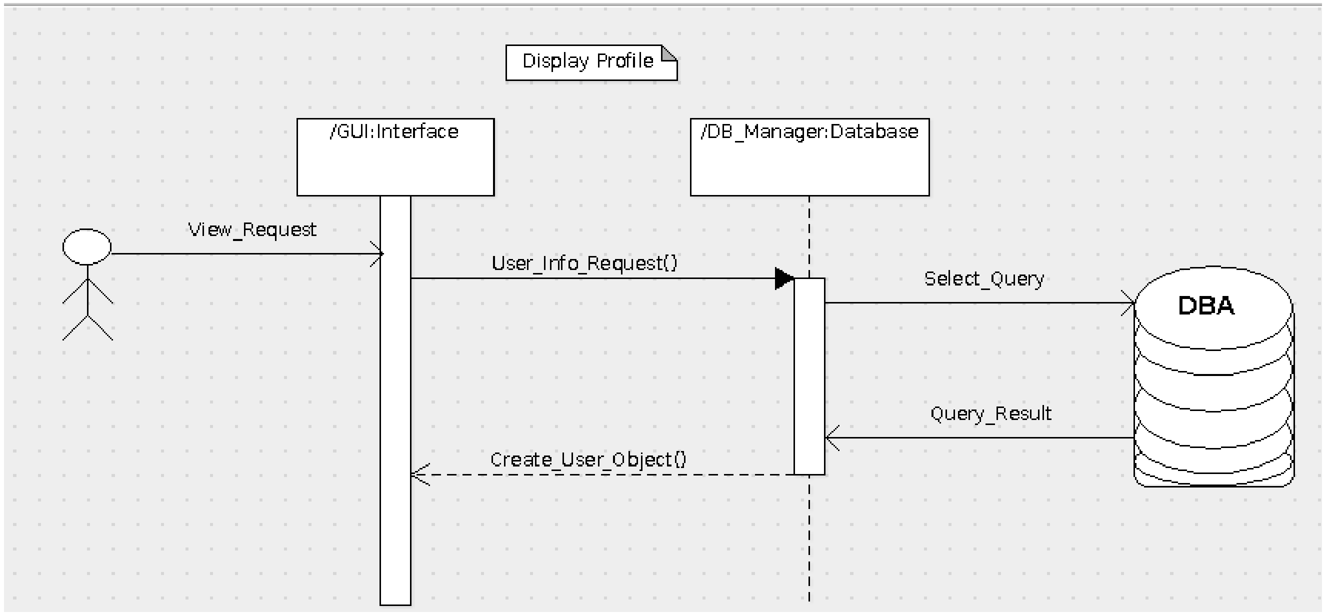
Join Even

To complete the join event process, the user begins by clicking a *Join Event* button provided by the GUI which then relays the users **Join_Request()** to the **DB_Manager**. From here an Update_Quary is sent to the Database which responds with a *number of rows affected* response and a Boolean *true/false* which is finally sent back to the **GUI**.



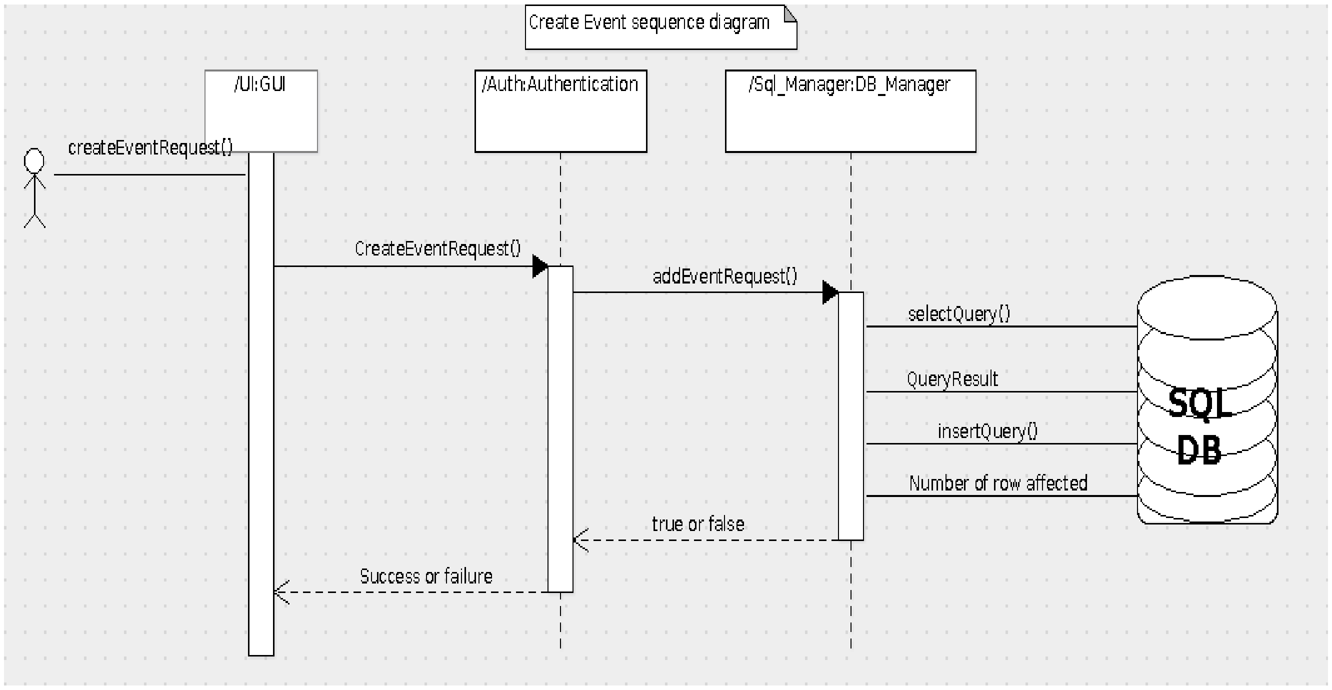
Display Profile Info

In order for the user to view his/her profile info, the user clicks on a *View Profile* button provided by the GUI. A **DB_Manager object** will be created by the GUI via a **View_User_Info()** function. The **DB_Manager class** will the send a **Select_Quary()** to the Database which will then respond with a **quaryResult**. The **DB_Manager** will then respond to the *Query Result* by returning a **User object** to the **User class** which will then relay it back to the **GUI**.



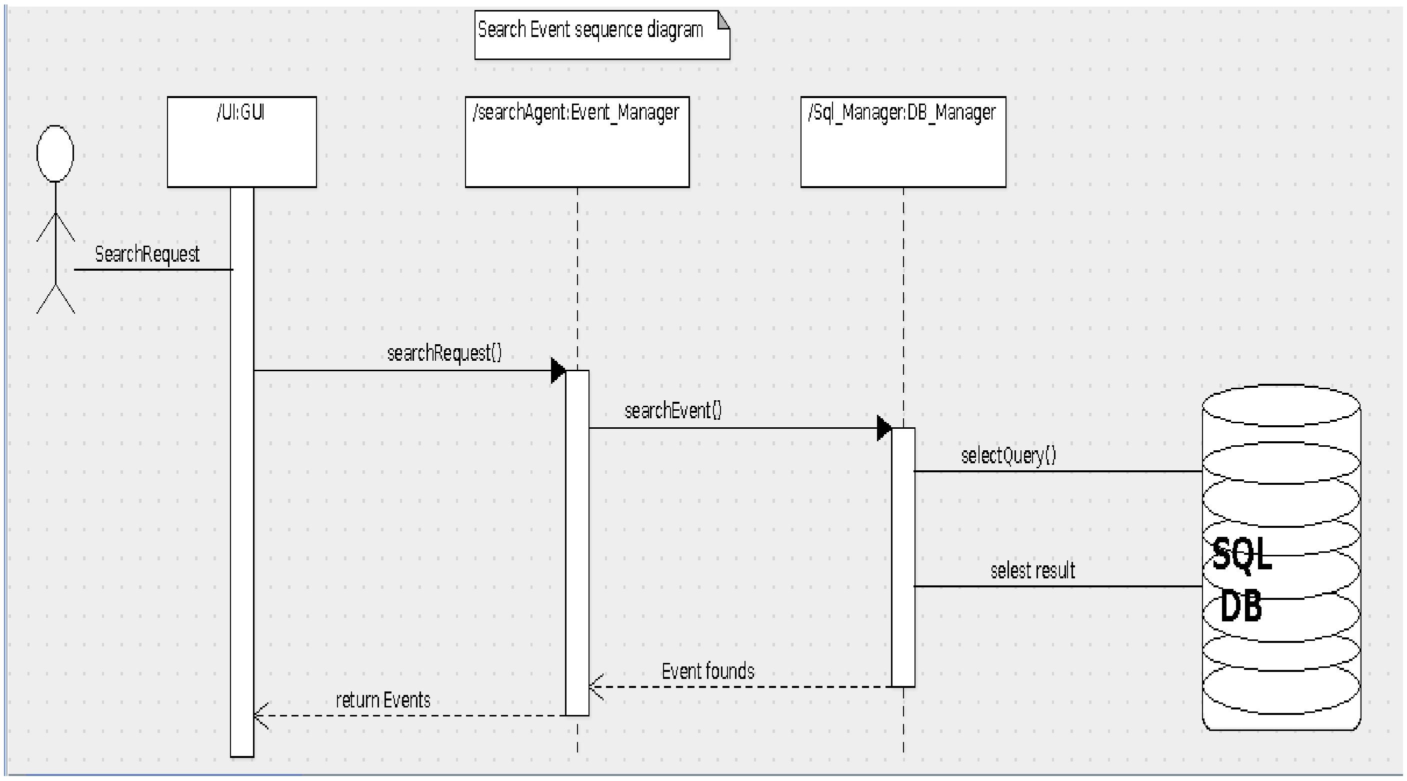
Create Event

To create an event, the user interacts with a *Create Event* button provided by the **GUI** by simply clicking the button. The GUI then creates Authentication object via the **Create_Event_Request()** function. The Authentication class further creates a DB_Manager object through the **Add_Event()** function and the **DB_Manager** finally sends a **Select_Query()** to the **Database** and begins to handle the *Select Result* returned by the **Database** which of course is of Boolean form. In the case of a false, the **DB_Manager** will send a *null/false* value to the **Authentication** which will relay an Event creation fail to the **GUI**. However in the case of a true, the **DB_Manager** will interact with the Database once more by sending an **Insertion_Quary()** to the Database and handles the response which will more likely be a number of rows affected and a Boolean true accordingly. The result is then sent to the Authentication and finally relayed back to the **GUI** as an Event Creation success or Boolean true. No **user object** is required here as no User info needs to be displayed.



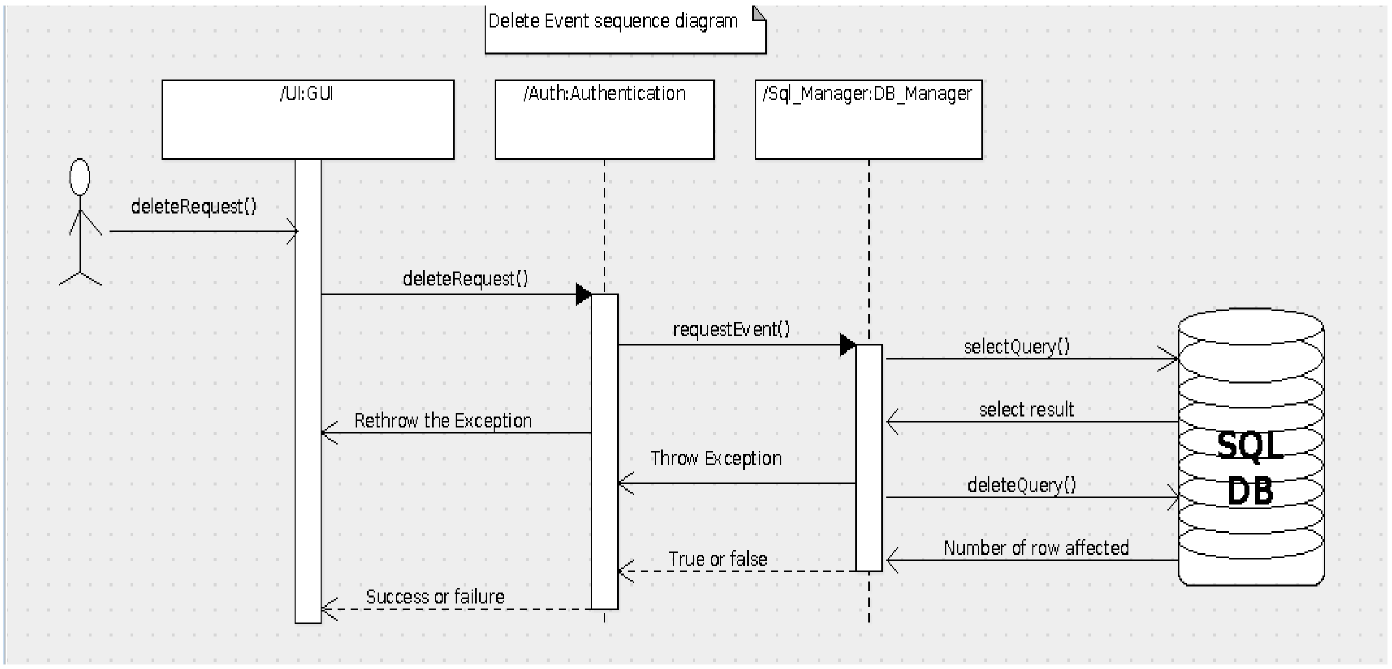
Search Even

In the search for Event process, the will enter Event info of the Event he/she is looking for in a search field provided by the **GUI**. Preferably in most cases this will be the Event Name. A **SearchAgent object** is then created by the **GUI** via the **Search_Request()** function. The **SearchAgent** then creates a **DB_Manager object** through the **Search_Event()** and finally the **DB_Manager** sends **Select_Quary()** to the **Database** which further responds with a *Select Result*(Boolean *true*/*false*). The Select Result is handled appropriately by the **DB_Manager** and an **Event object** is created by the **DB_Manager class** and sent back to the **GUI** as an Event Search *success* through the **SearchAgent class** for the Boolean *true* case. In the case of a false *Select Result* from the Database, the **DB_Manager** will send a null value to the **SearchAgent class** which will relay it to the **GUI** as an Event Search failure.

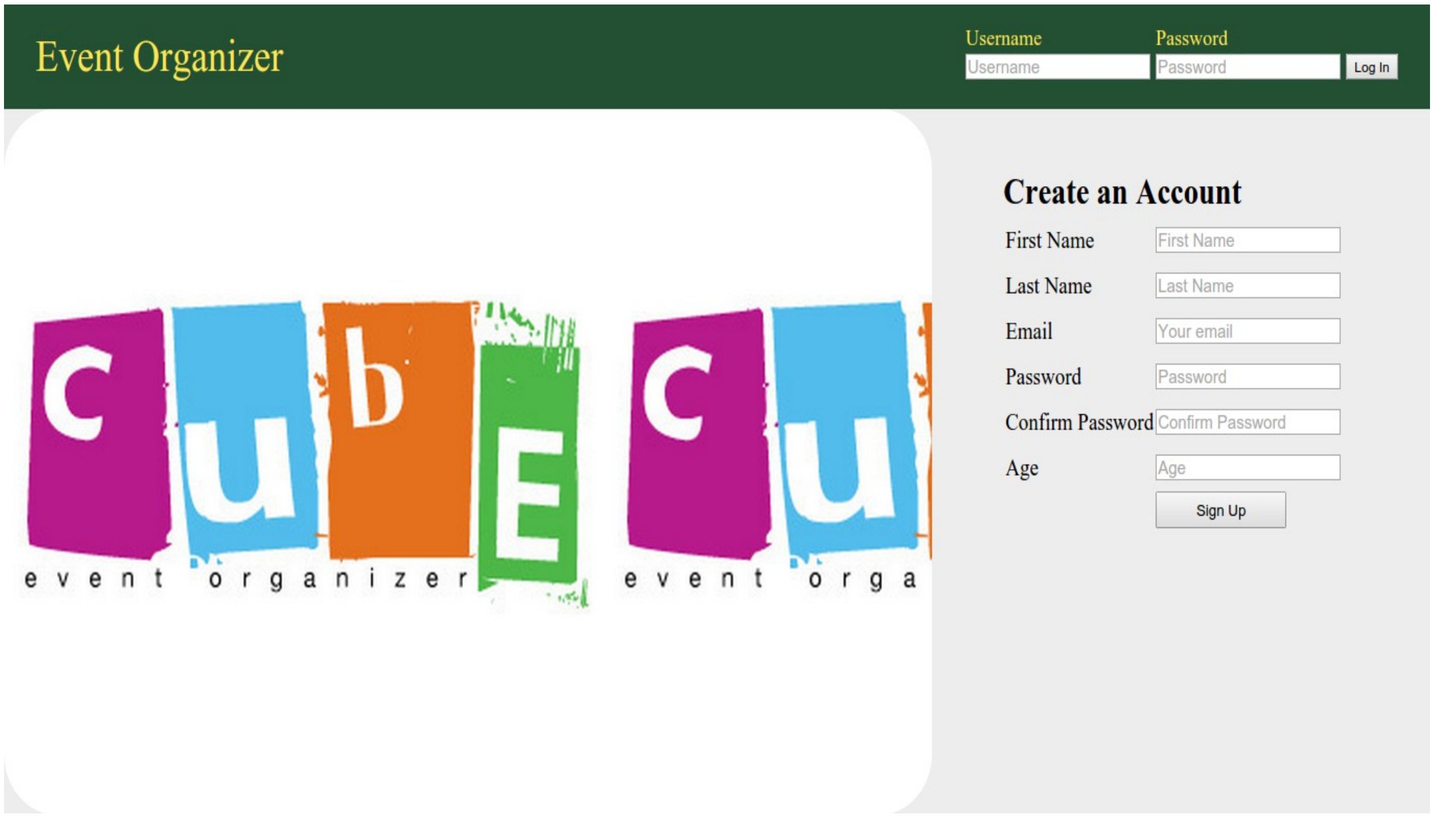


Delete Event

In the delete event process, the user enters the Event info of the Event he/she intends to delete and clicks on a *Delete* button. All these interfaces will be provided by the **GUI**. When this is done, an **Authentication object** will be created by the **GUI** via the **Delete_Request()** function. From here the **Authentication** class creates a **DB_Manager object** through a **Request_Event()** function and the **DB_Manager** finally sends a *Select Query* to the Database which will return a *Select Result*(Boolean *true*/*false*). In the case of a *true* Boolean value, the **DB_Manager** will return the Boolean *true* to the **Authentication** class which will then create another **DB_Manager object** but this time through the **Delete_Event()** function and the **DB_Manager** will interact with the Database once again, but this time with a *Delete Query*. A Boolean *true* and *number of rows affected response* is handled by the **DB_Manager** once again assuming no network or other errors occurred and is sent to the **Authentication** class which further relays the response(Boolean *true*) back to the **GUI** as a *Deletion success*. However in the case of a Boolean *false Select Result* response from the Database, the **DB_Manager** sends the result which is a *null* value to the **Authentication**, which further relays it to the **GUI** as a *Deletion fail*.

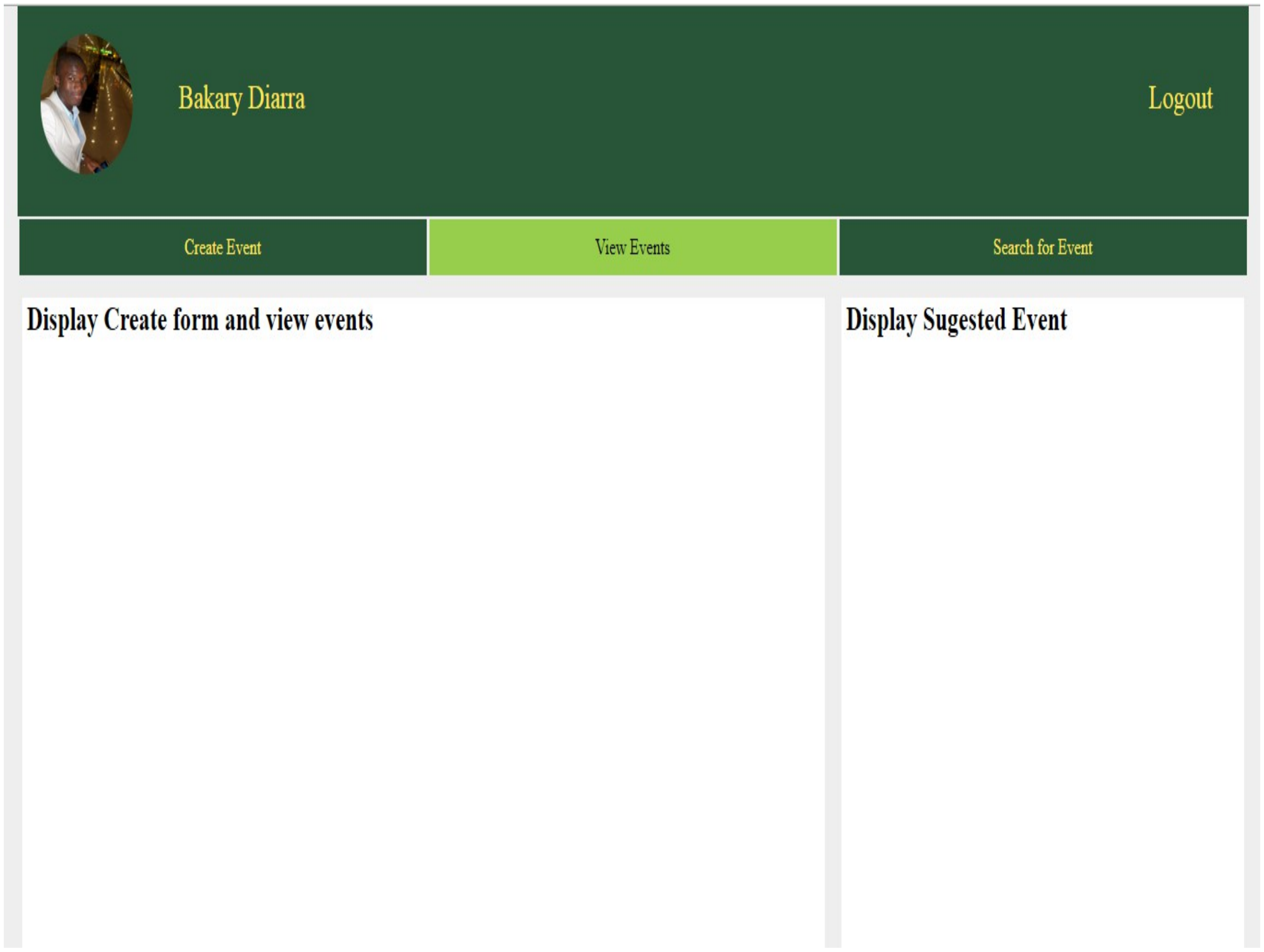


User Interface Preview



Create an Account

First Name	First Name
Last Name	Last Name
Email	Your email
Password	Password
Confirm Password	Confirm Password
Age	Age



Bakary Diarra
Logout
Create Event
View Events
Search for Event
Display Create form and view events
Display Sugested Event