Sportify
Software Architecture Document

Version 2.0

Revision History

Date	Version	Description	Author
<23/10/22>	<1.0>	First draft	Firangiz Ganbarli
<27/10/22>	<2.0>	Updated the design class diagram and descriptions	Firangiz Ganbarli

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. Is it intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This Software Architecture Document provides an architectural overview of Sportify. Sportify will allow users to register and create sport events and will also allow users to search for events that are around them and show the results of that query.

1.3 References

1. Sportify – Use Case Specification
2. Sportify – Supplementary Specification
3. Sportify – Use Case Realization

2. Architectural Representation

This document presents the architectural as a series of views: use case view, process view, deployment view, and implementation view. These views are presented as Rational Rose Models and use the Unified Modeling Language (UML).

3. Architectural Goals and Constraints

Sportify is to be developed as a stand-alone web app that will be accessible by any major Internet Browsers. It consists of four key components: a User Client Module, a Server Module, a Database, and an Administrator Client Module. All components must be able to execute on a personal computer. The User Client and Administrator Client modules must communicate with the server over a TCP/IP connection. The Server and Database components should be located on the same host.

4. Use-Case View

The Use Case View is crucial input to the selection of the set of scenarios and use cases that are the focus of an iteration. It describes the set of scenarios and use cases that represent some significant, central functionality. It also describes the set of scenarios and use cases that have many architectural elements.

For more information, refer to Use-Case Specification Requirements document.

4.1 Use-Case Realizations

Refer to Use Case Realization document.

5. Logical View

5.1 Overview

This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.

5.2 Architecturally Significant Design Packages

5.2.1 Design Model: Packages Diagram

The design model represents the structure and organizations of Sportify. Packages and classes are presented with a brief description.

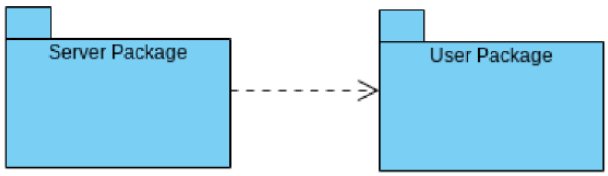

Figure 1: Design Model Package

5.2.2 Design Model: Packages Description

Server Package	
Description	This is the main system package where all client queries are managed
Corresponding Classes	Observable, Observer, SportifyServer, DeveloperObserver, AdministratorObserver
Relations	Main package, Dependent of: User
Sub-Packages	Users

User Package	
Description	All info regarding users are handled in this package
Corresponding Classes	User, Event, EventDatabase, SignedUpEvent, SearchResults
Relations	Sub package of Server
Sub-Packages	None

5.2.3 Design Model: Design Class Diagram

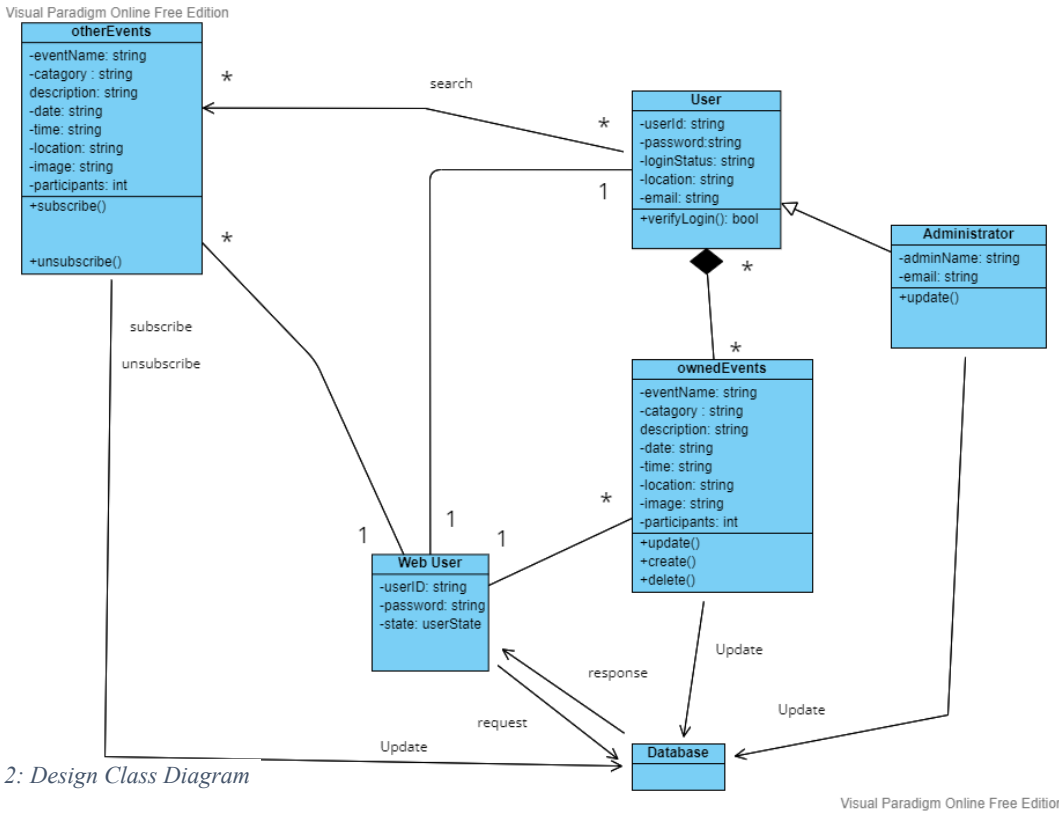

Figure 2: Design Class Diagram

5.2.4 Design Classes Description

Property	Description
Name	User
Description	Represents the User entity
Responsibilities	Maintains and updates all the private User information
Relations	Generalization with Administrator, Composition with ownedEvents of multiplicity *, one way association with otherEvents of multiplicity * and association with Web User.
Attributes	userID: the username of the user password: password the user sets up loginStatus: whether login was successful or not location: the city user is based in email: user's email
Methods	verifyLogin(): verifies the login information and authenticates the user

Property	Description
Name	Administrator
Description	It's a class that represents the SportifyServer, deals with the user and database.
Responsibilities	Communicates between the user and the database the server uses.
Relations	Generalization with User, One-way Association with Database
Attributes	adminName: the name of the administrator email: email of the admin acocunt
Methods	update(): updates the user information accordingly to the database

Property	Description
Name	ownedEvents
Description	It's a class that represents events that host has created.
Responsibilities	Allows the event host to manage the event information page.
Relations	Composition with User of multiplicity *, One-way association with database, and an association with a web user of 1.
Attributes	eventName: title of the event category: the sport category of the event description: short description of the event date: day of the event time: time of the event location: the address the event is located at image: optional image to put as a background image participants: list of users who signed up for the event
Methods	update(): lets the host edit event information create(): lets users create a new event delete(): lets hosts delete their event

Property	Description
Name	Database
Description	Class that represents where the event and user data are located at
Responsibilities	Contains all the event and user information in a secure database
Relations	Association with ownedEvents, Administrator and WebUser
Attributes	none
Methods	none

Property	Description
Name	Web User
Description	Represents an active user
Responsibilities	Allows a logged in user access various features of the app
Relations	Association with otherEvents with multiplicity of *, association with database, association with User with multiplicity of 1, and an association with ownedEvents with multiplicity of 1.
Attributes	userID: the username of the user password: password user has set up for login state: the activity state of the user
Methods	none

Property	Description
Name	otherEvents
Description	Class that represents events that users have access to see and sign up for
Responsibilities	Contains all the event info and lets users sign up or unregister for an event.
Relations	Association with User with multiplicity of *, and association with 1 web user, and an association with database
Attributes	eventName: title of the event category: the sport category of the event description: short description of the event date: day of the event time: time of the event location: the address the event is located at image: optional image to put as a background image participants: list of users who signed up for the event
Methods	subscribe(): lets users sign up for an event unsubscribe (): lets users unregister for an event they have already signed up for

6. Interface Description

Refer to the files within Sportify -> Prototype folder.

7. Size and Performance

The selected architecture supports the sizing and timing requirements. The website components have been designed to ensure that it takes minimal time to load and does not slow down the browser significantly.

8. Quality

The software architecture supports the quality requirements, as stipulated in the Software Requirements Specifications and Supplementary Specification.

Sportify
Use-Case-Realization Specification

Version 2.0

Revision History

Date	Version	Description	Author
<23/10/2022>	<1.0>	Added the initial use-case-realization specifications	Firangiz Ganbarli
<27/10/2022>	<2.0>	Formatted the document, changed the sequence diagram to match the others	Raven Duong

Table of Contents

Use-Case-Realization Specification

1. Introduction

1.1 Purpose

This document provides a comprehensive overview of the system, using a number of different diagrams for representing the system functions.

1.2 Scope

Sportify is a social media app that allows sport enthusiasts to see recreational events around them and create their own events to gather sign-ups. This Use Case Realization document provides an overview of the use cases developed in Sportify.

1.3 References

1.3.1 Sportify – Use Case Specification

1.3.2 Supplementary Specifications

1.4 Overview

The sections of the Use Case Realization document describe use cases in terms of their flow of events, participating objects and corresponding diagrams.

2. Use-Case <Login>

2.1 Brief Description

This Use-Case defines how users are logged into the system and get access to their respective functionalities.

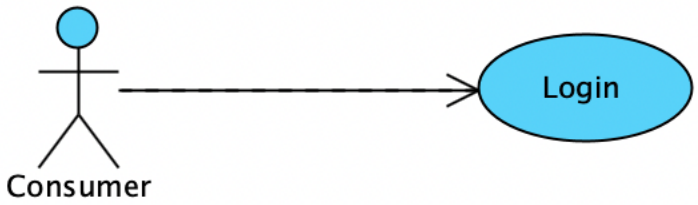

Figure 1: Login

2.2 Flow of Events

The user provides his username and password and submits the form. Data is validated and login process is activated.

2.3 Interaction Diagrams

- The user enters their username and password and submits the data
- Browser receives the query and loads the webpage

2.3.1 Sequence Diagrams

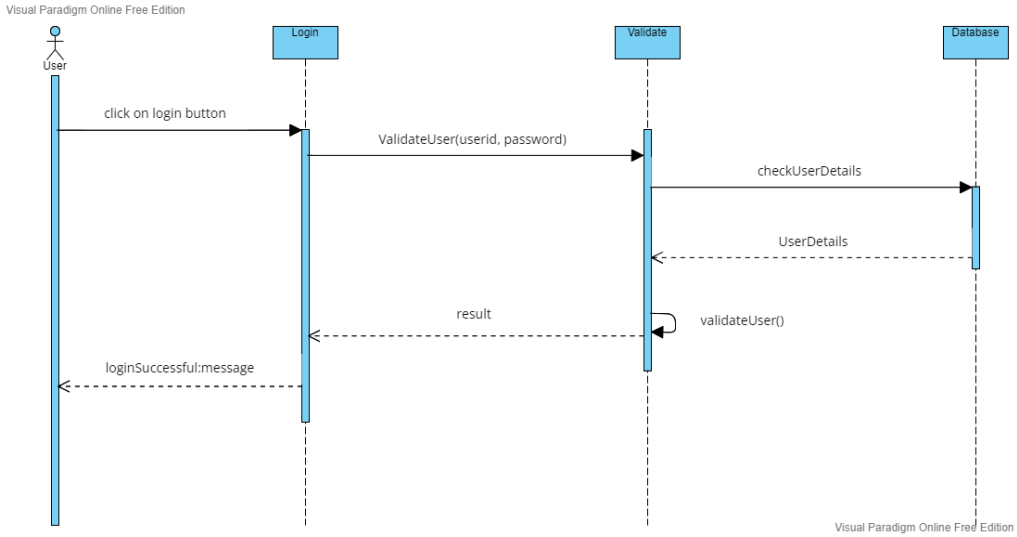

Figure 2: Sequence Diagram - Login

2.4 Participating Objects

The following objects collaborate and define the Use-Case <Login>:

- Observable: This object represents the visible interface of the application and allows the user to login to their account.
- Server: This object executes and validates the Login query by communicating with the database.

2.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

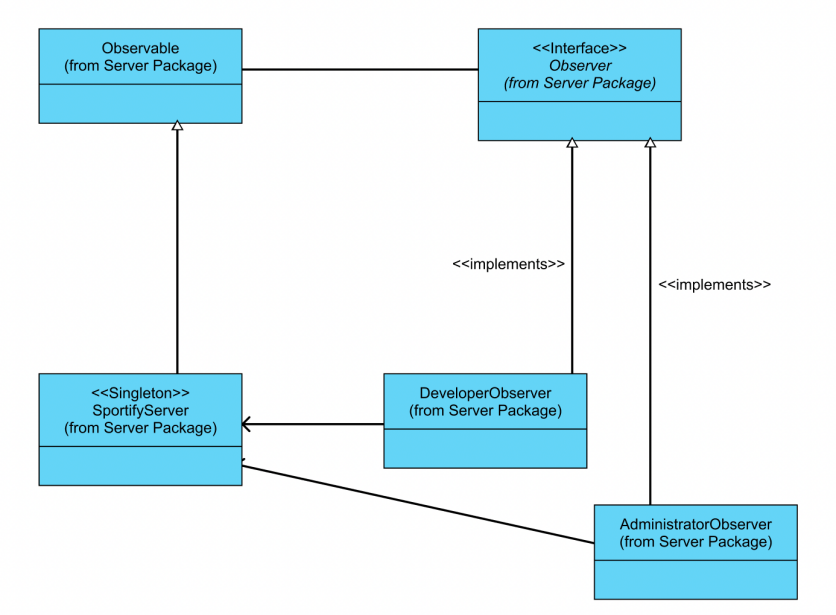

Figure 3: Object Diagram - Login

3. Use-Case <Logout>

3.1 Brief Description

This Use-Case defines how users are logged out from the system and exit the application normally.

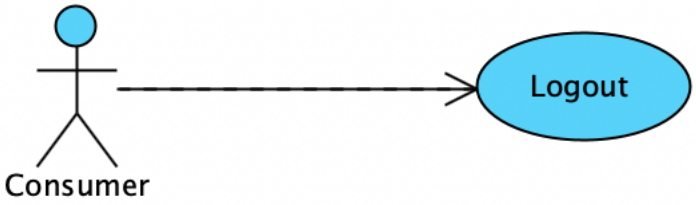

Figure 4: Logout

3.2 Flow of Events

The user exits the application by clicking the appropriate “Log out” button. Query is validated, and logout process is activated.

3.3 Interaction Diagrams

- The user clicks on the Logout button
- The Internet Browser receives the query and logs the user out of the system, redirecting to the appropriate webpage.

3.3.1 Sequence Diagrams

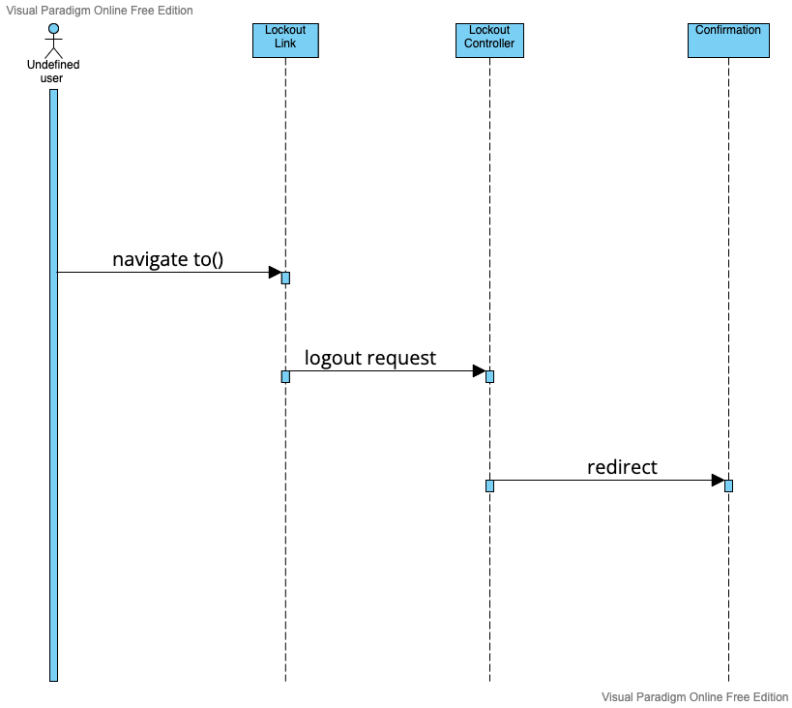

Figure 5: Sequence Diagram - Logout

3.4 Participating Objects

The following objects collaborate and define the Use-Case <Logout>:

- Observable: This object represents the visible interface of the application and allows the user to log out of their account.
- Server: This object executes and validates the Logout query by communicating with the database.

3.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

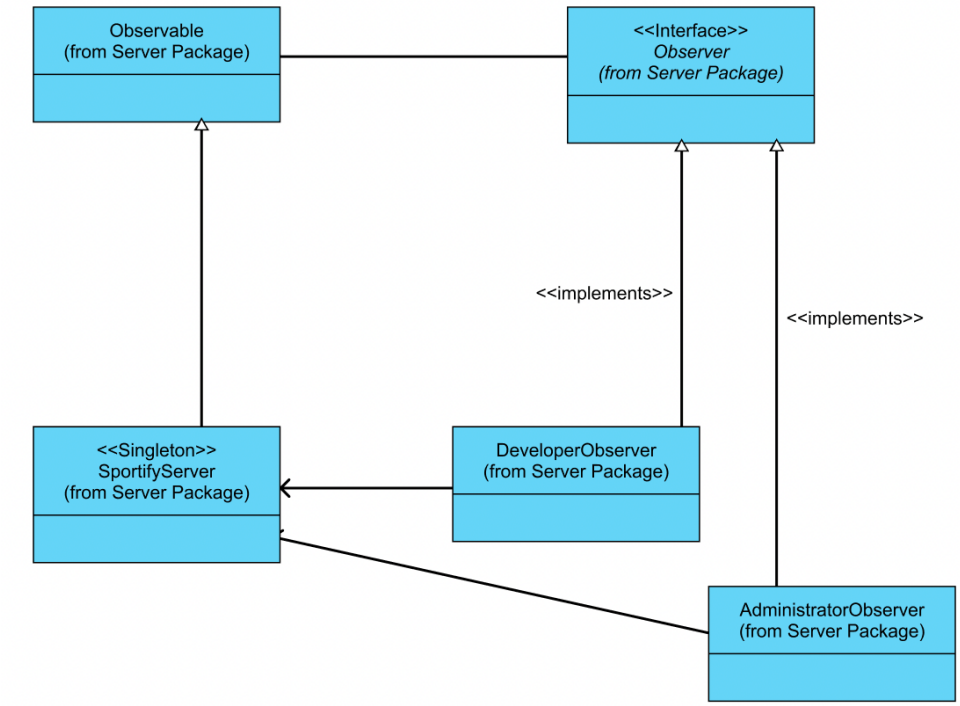

Figure 6: Object Diagram: Logout

4. Use-Case <Create an account>

4.1 Brief Description

This Use-Case defines how users create an account on the website.

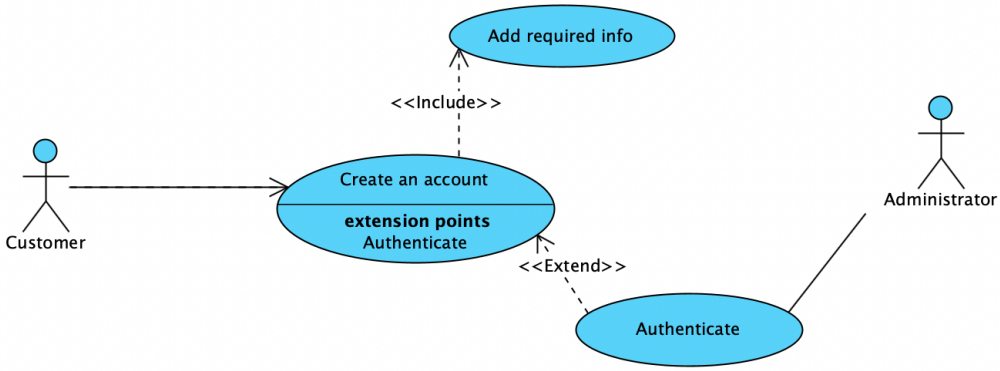

Figure 7: Create an account

4.2 Flow of Events

The user clicks on Create an account button and fills out the required information. The administrator authenticates their given info.

4.3 Interaction Diagrams

- The user clicks on Create an account button

- The query is received the Internet Browser, which communicates with the Authentication services to verify user and then redirects them to log in page.

4.3.1 Sequence Diagrams

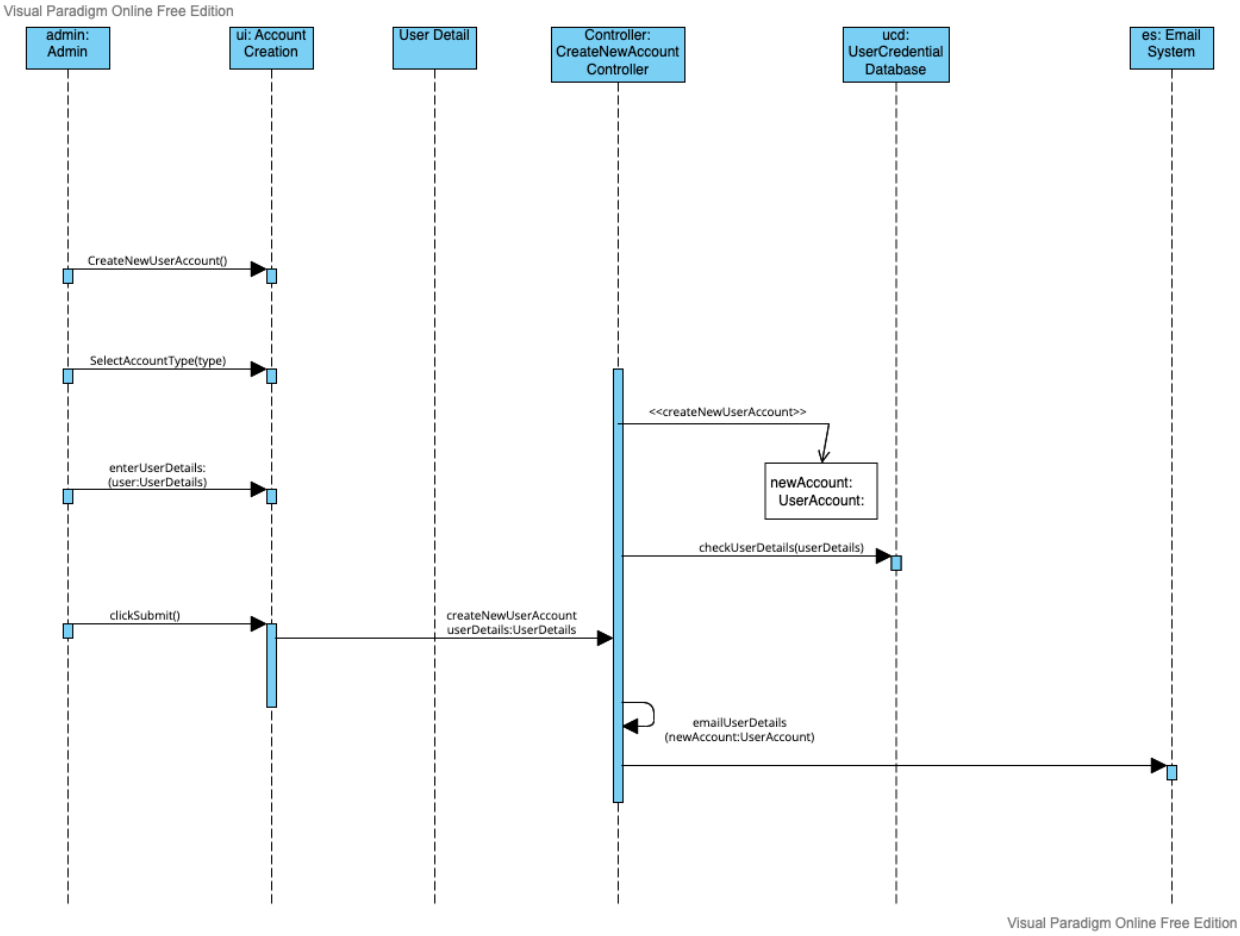

Figure 8: Sequence Diagram - Create an account

4.4 Participating Objects

The following objects collaborate and define the Use-Case <Create an account>:

- Observable: This object represents the visible interface of the application and allows the user to create an account
- Server: This object interacts with the database to create the user's account

4.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

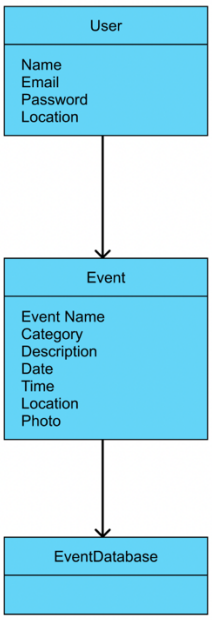

Figure 9: Object Diagram - Create an account

5. Use-Case <Create an event>

5.1 Brief Description

This Use-Case defines how users create their own event on the website.

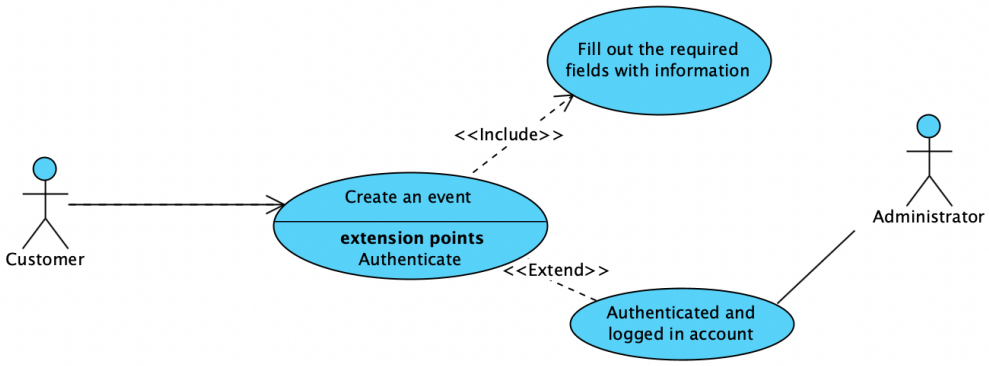

Figure 10: Create an event

5.2 Flow of Events

The user creates a new event on the webpage by clicking the appropriate “Create an event” button. Query is validated, and they are directed to the “Create an event” page where they fill out the appropriate fields of the form and submit.

5.3 Interaction Diagrams

- The user clicks on Create an event button
- Query is received by the Internet Browser, and the user is redirected to the “Create an event” page
- Once the required information is filled out, the query is received and the event details are added to the database.

5.3.1 Sequence Diagrams

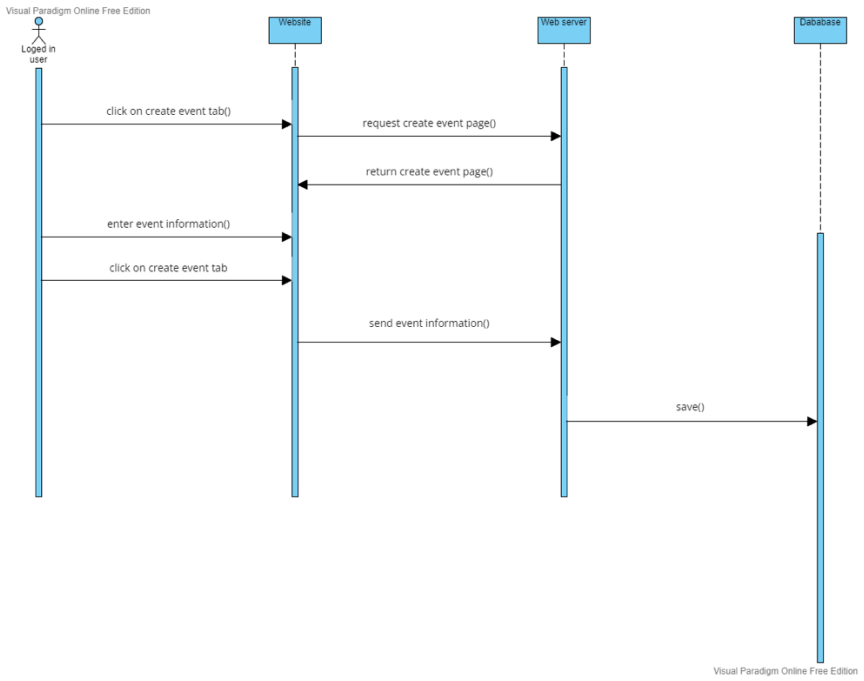

Figure 11: Sequence Diagram - Create an event

5.4 Participating Objects

The following objects collaborate and define the Use-Case <Create an event>:

- Observable: This object represents the visible interface of the application and allows the user to create an event
- Server: This object interacts with the database to create the user's event.

5.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

Figure 12: Object Diagram - Create an event

6. Use-Case <Edit an event>

6.1 Brief Description

This Use-Case defines how users can edit the event information after they have already created an event.

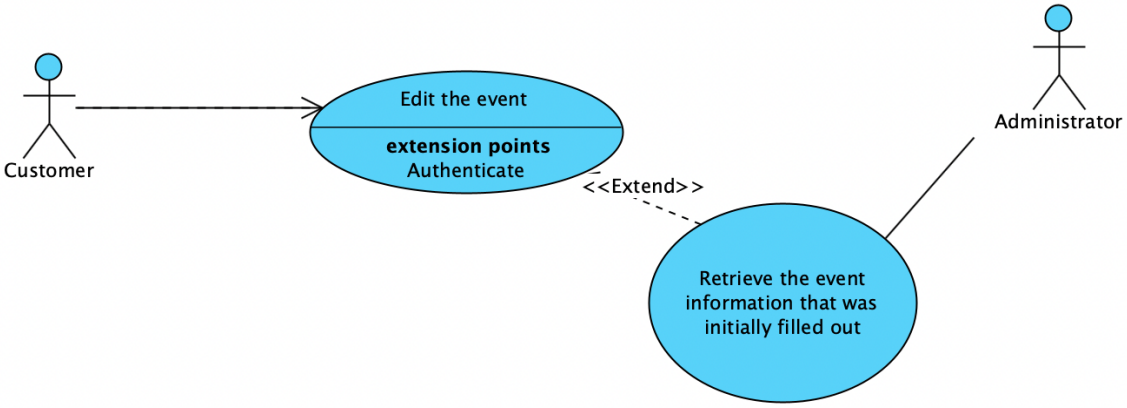

Figure 13: Edit an event

6.2 Flow of Events

The user can edit the event they already created by clicking on the “Edit event details” button and then change the appropriate field and submit the form again.

6.3 Interaction Diagrams

- The user clicks on “Edit event details” button
- The browser receives the query, and redirects to the event details page where user can change info
- This updated info is added to the database

6.3.1 Sequence Diagrams

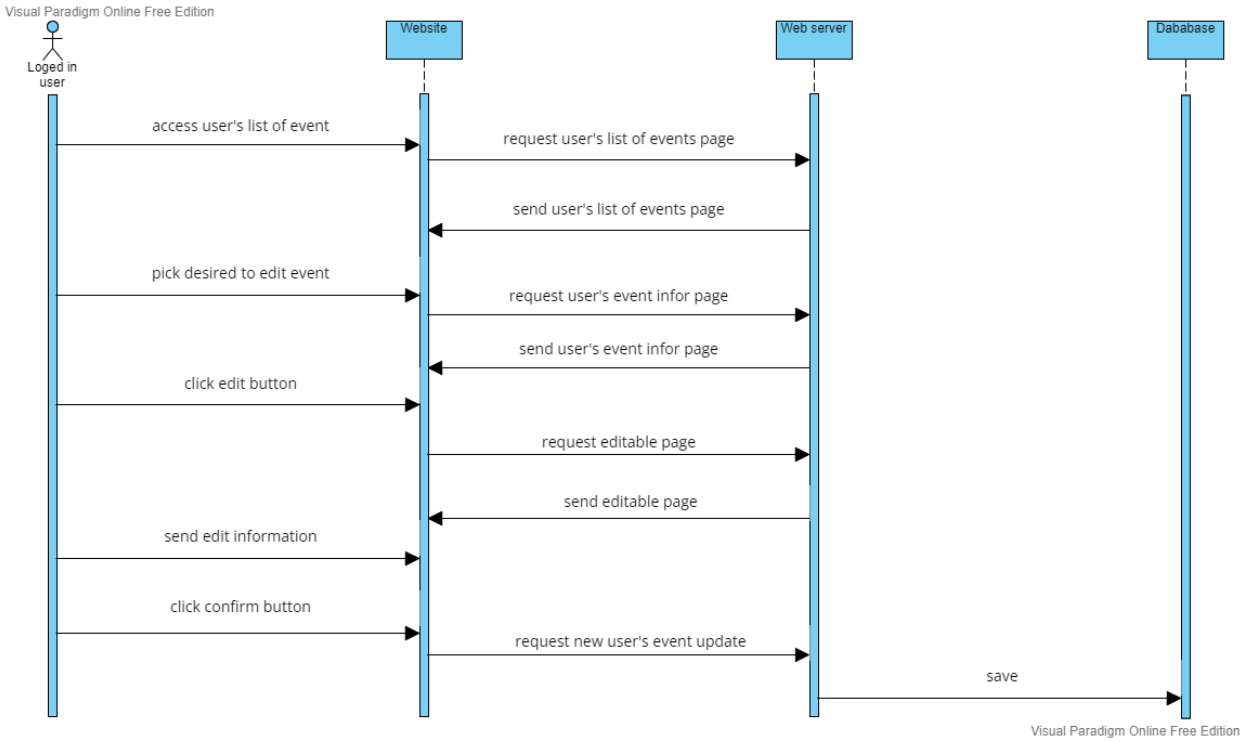

Figure 14: Sequence Diagram - Edit event

6.4 Participating Objects

The following objects collaborate and define the Use-Case <Edit an event>:

- Observable: This object represents the visible interface of the application and allows the user to edit their event.
- Server: This object interacts with the database to edit the user’s event.

6.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

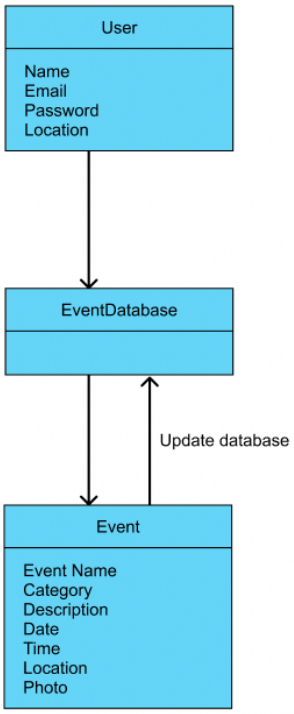

Figure 15: Object diagram - Edit an event

7. Use-Case <Delete an event>

7.1 Brief Description

This Use-Case defines how users can delete the event they created.

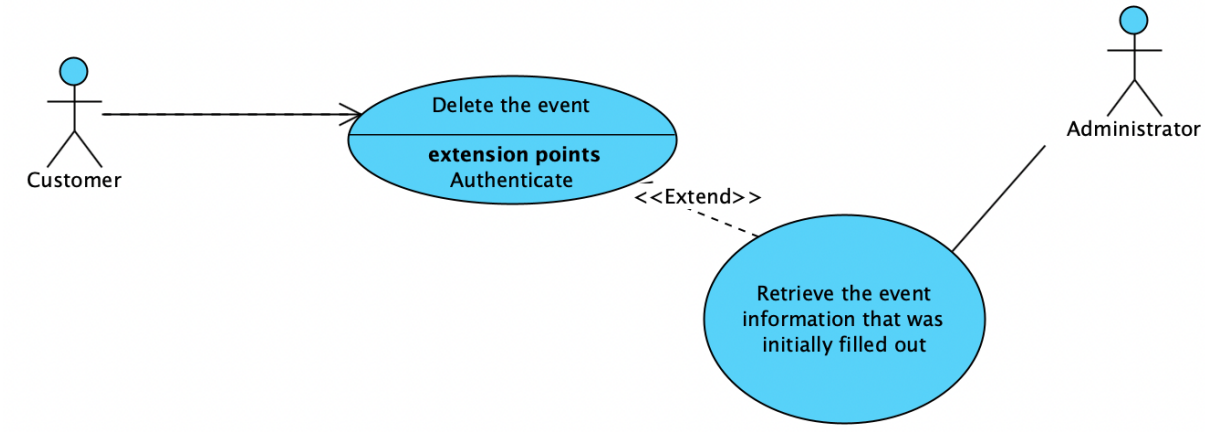

Figure 16: Delete an event

7.2 Flow of Events

The user can simply click on edit event button to be redirected to that page, where they can click on “Delete event” button to delete the event all together.

7.3 Interaction Diagrams

- The user launches “Edit event” page where they click on “Delete the event” button
- The Internet Browser receives this query, and communicates with database to delete the appropriate information.

7.3.1 Sequence Diagrams

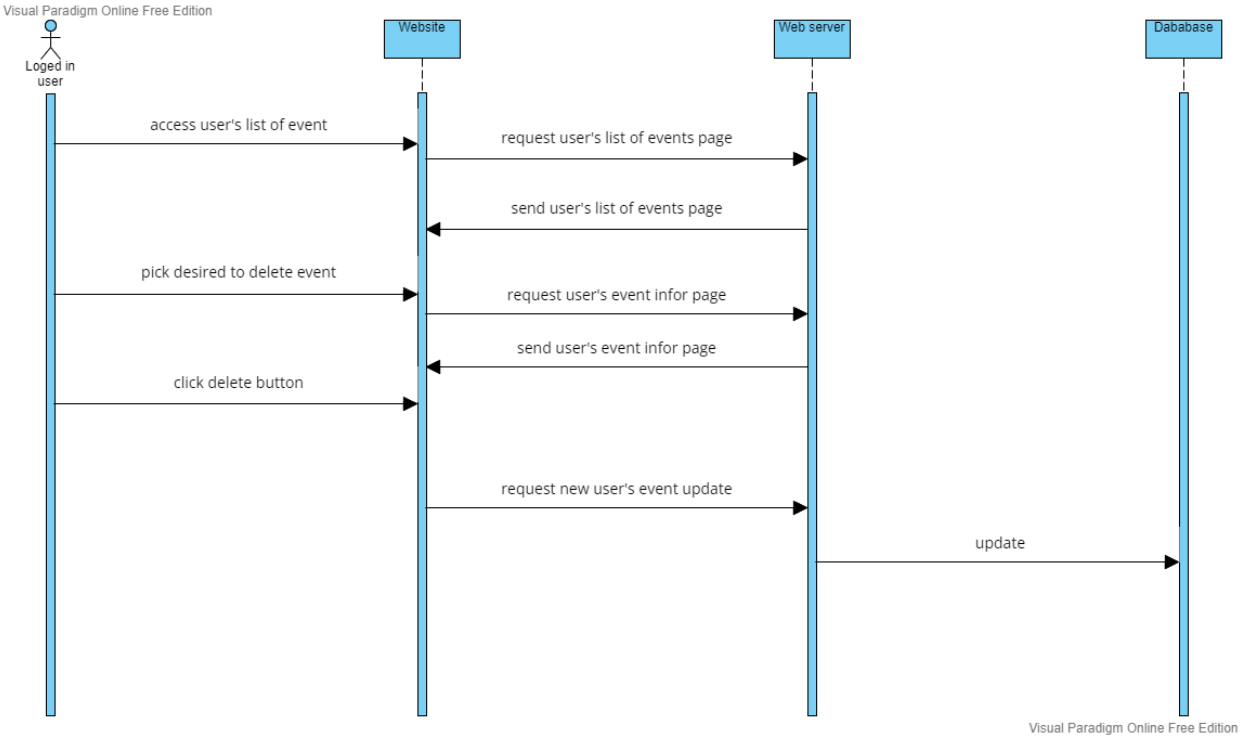

Figure 17: Sequence Diagram - Delete an event

7.4 Participating Objects

The following objects collaborate and define the Use-Case <Delete event>:

- Observable: This object represents the visible interface of the application and allows the user to delete their event.
- Server: This object executes and validates the Delete event query by communicating with the database.

7.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

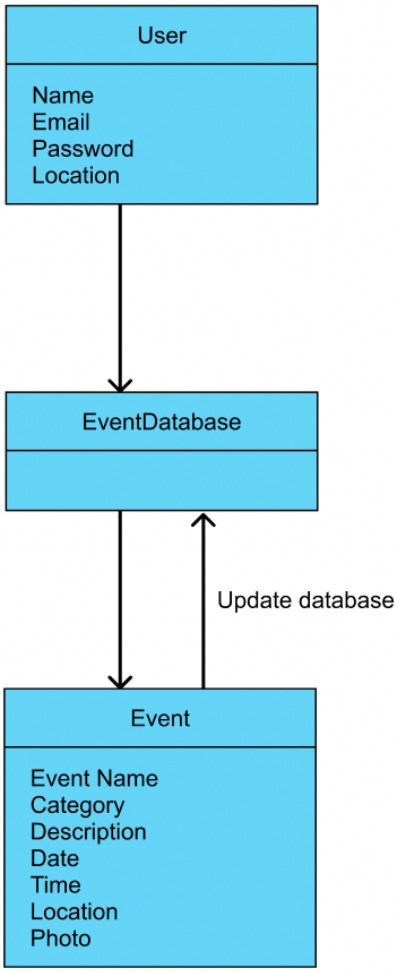

Figure 18: Object Diagram - Delete an event

8. Use-Case <View an event>

8.1 Brief Description

This Use-Case defines how users can view an event.

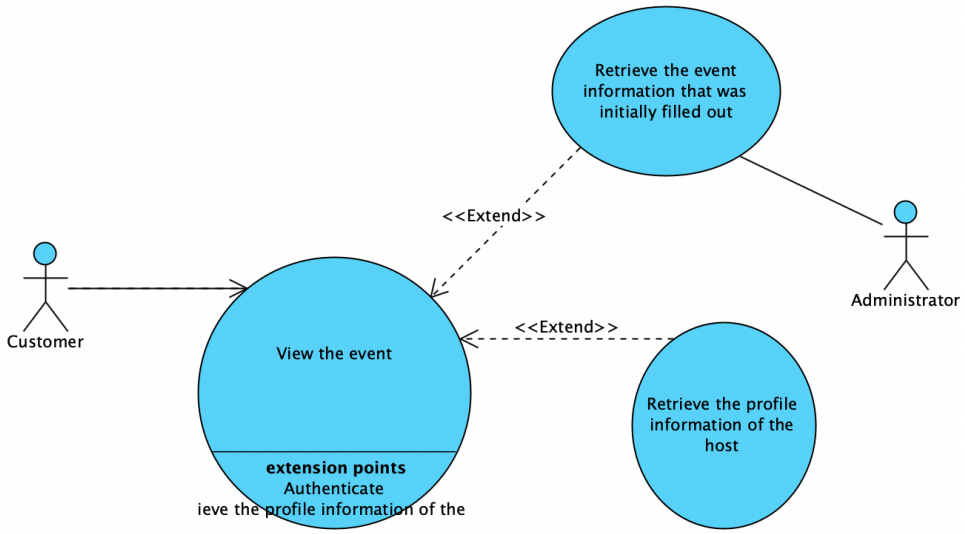

Figure 19: View an event

8.2 Flow of Events

The user can simply click on the event itself from the search or home page, when the query is recognized, the process of retrieving the event and host information will be initiated, and then the info will be displayed.

8.3 Interaction Diagrams

- The user clicks on the event
- The Internet Browser receives the query and communicates with the database to display the necessary information.

8.3.1 Sequence Diagrams

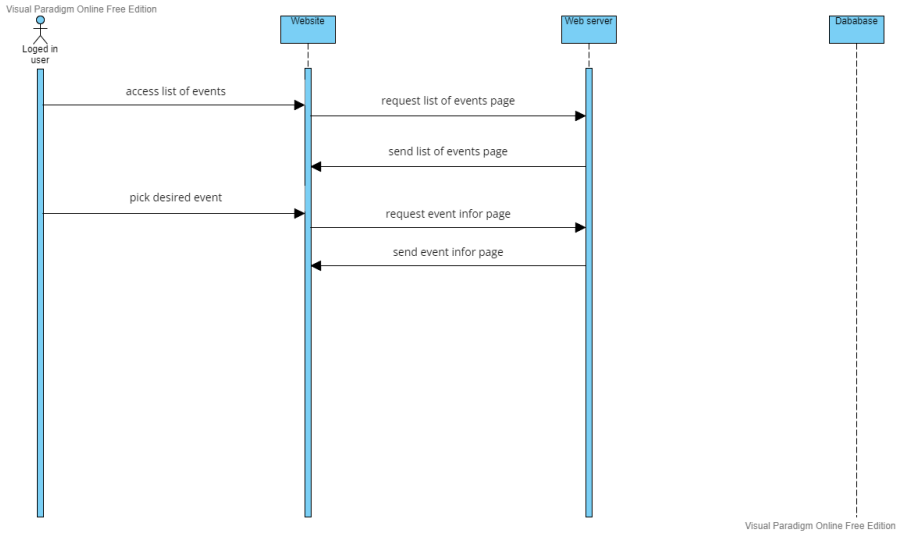

Figure 20: Sequence Diagram - View an event

8.4 Participating Objects

The following objects collaborate and define the Use-Case <View an event>:

- Observable: This object represents the visible interface of the application and allows the user to view an event.
- Server: This object interacts with the database to display an event.

8.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

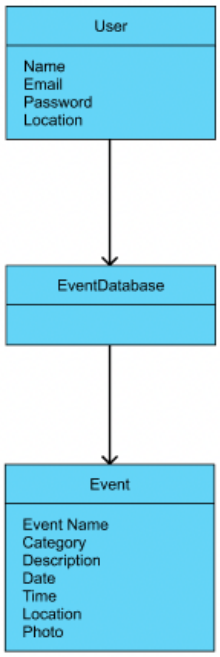

Figure 21: Object diagram - View an event

9. Use-Case <Sign up for an event>

9.1 Brief Description

This Use-Case defines how users can sign up for an event.

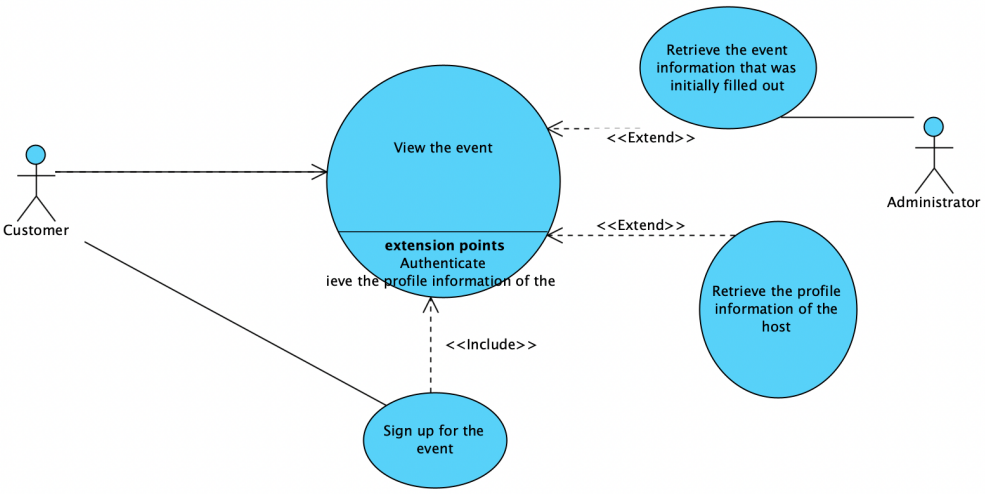

Figure 22: Sign up for an event

9.2 Flow of Events

On the event information page, the user can simply click on sign up for the event to be officially registered for it.

9.3 Interaction Diagrams

- The user clicks on “Sign up for event” button
- The Internet Browser receives the query, and updates the webpage accordingly
- Database adds the user to the list of registered people for the event

9.3.1 Sequence Diagrams

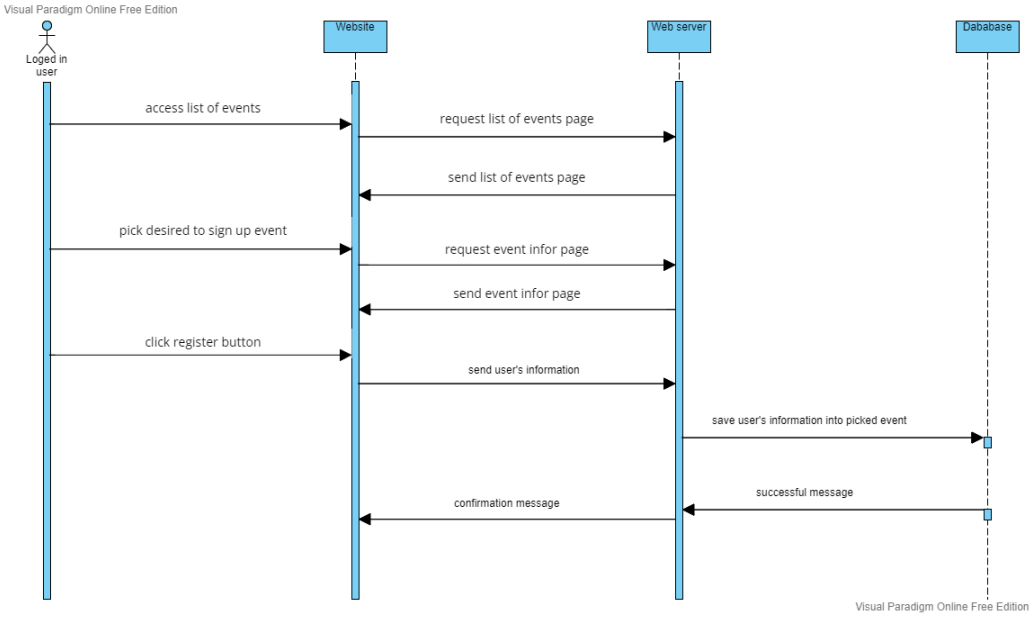

Figure 23: Sequence diagram - Sign up for an event

9.4 Participating Objects

The following objects collaborate and define the Use-Case <Sign up for an event>:

- Observable: This object represents the visible interface of the application and allows the user to sign up for an event.
- Server: This object executes and validates the Signup query by communicating with the database.

9.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

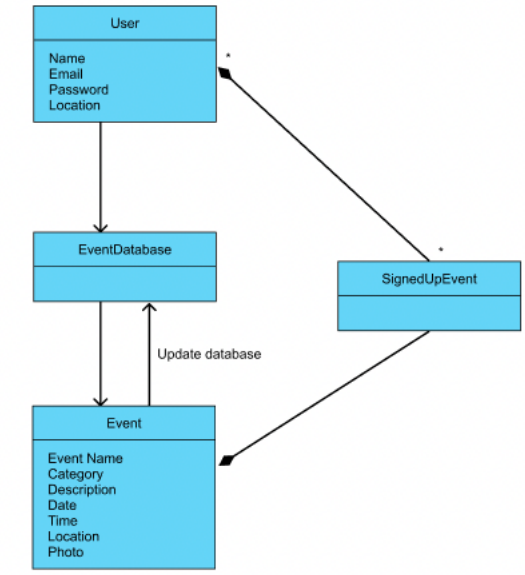

Figure 24: Object diagram - Sign up for an event

10. Use-Case <Search for an event>

10.1 Brief Description

This Use-Case defines how users can search for an event using the search bar on the navbar.

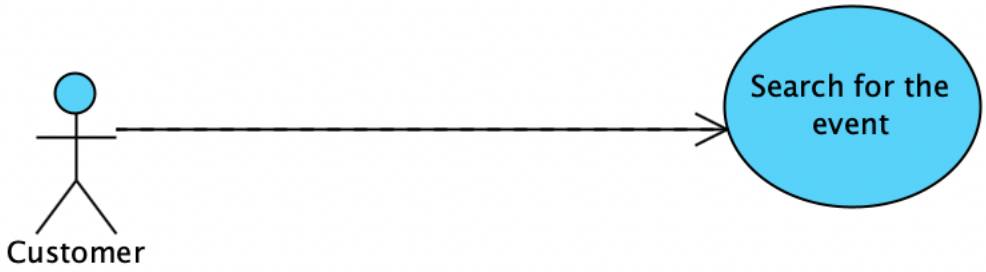

Figure 25: Search for an event

10.2 Flow of Events

Anywhere on the website, the users have access to the navigation bar, so they can use the search field to search for the event they want and once the query is received and processed, the page will display the list of events that match that query.

10.3 Interaction Diagrams

- The user types their search query
- The Internet Browser receives this query and searches the database for the keywords and displays the events that match that query.

10.3.1 Sequence Diagrams

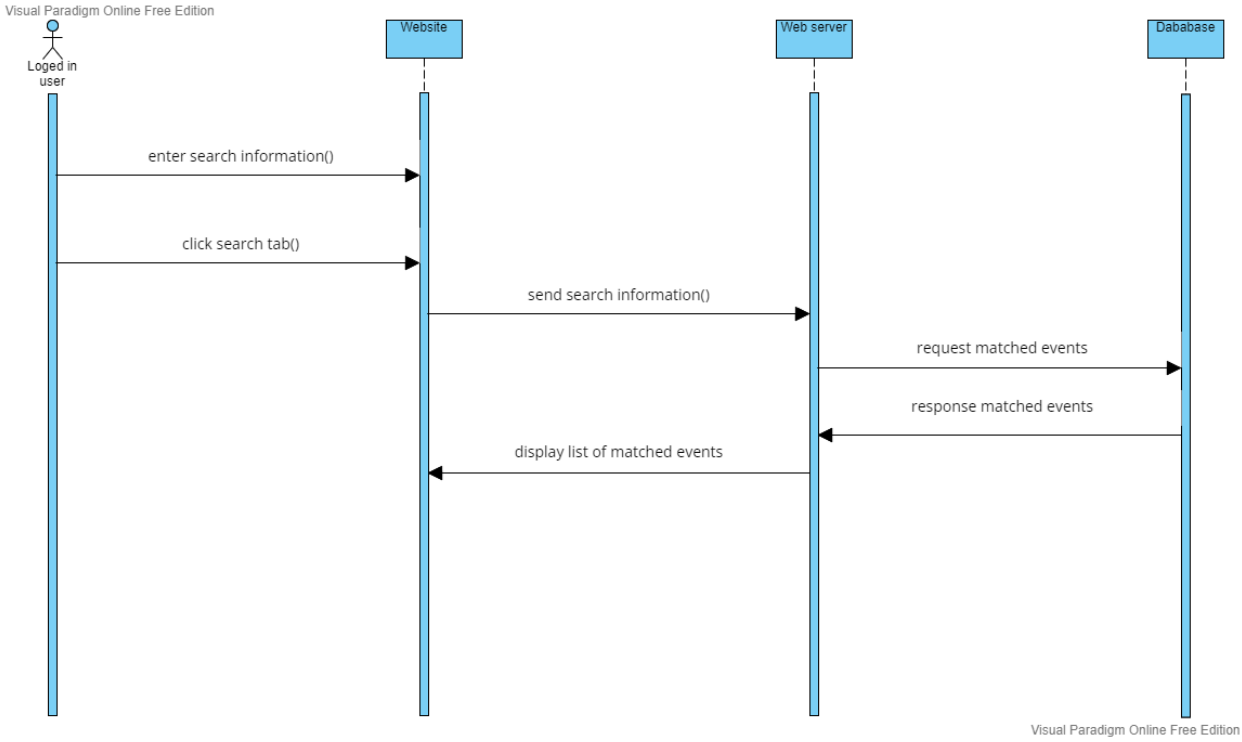

Figure 26: Sequence Diagram - Search for an event

10.4 Participating Objects

The following objects collaborate and define the Use-Case <Search for an event>:

- Observable: This object represents the visible interface of the application and allows the user to search for an event.
- Server: This object interacts with the database to display the search results of events.

10.5 Object Diagram

The following Object Diagram shows the relations and constraints between Classes and Objects that are involved in this use-case.

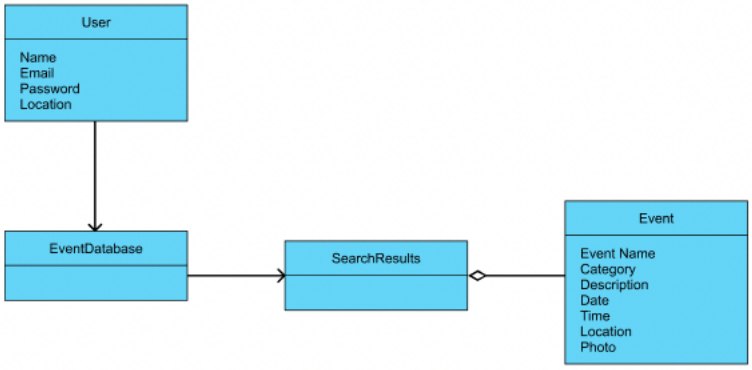

Figure 27: Object Diagram - Search for an event