## GOT’M

# Contents

[GOT’M 1](#_Toc133236269)

[Contents 2](#_Toc133236270)

[General Introduction 3](#_Toc133236271)

[Chapter I Functional Requirements 4](#_Toc133236272)

[I.1- Use cases: 4](#_Toc133236273)

[I.2- Actors: 4](#_Toc133236274)

[Chapter II Non Functional Requirements 6](#_Toc133236275)

[Chapter III UML 7](#_Toc133236276)

[III.1- Use Case Diagram: 7](#_Toc133236277)

[III.2- Global Class Diagram: 8](#_Toc133236278)

[III.3- Sequence Diagrams: 8](#_Toc133236279)

[Chapter IV CONCLUSION 29](#_Toc133236280)

# General Introduction

Clubs, associations, and organizations are ubiquitous in institutional establishments, leading to fierce competition among them. Each of these groups strives to cultivate a positive image and expand its community by implementing innovative ideas and activities. In this domain, digital marketing plays a pivotal role in attracting new audiences and fostering solid communities.

Our club, “GOT'M,” was created by gamers to assist fellow gamers in pursuing their competitive aspirations and potentially venturing into the professional e-sports scene. However, during our investigation, we discovered that our club's activities were scattered across various platforms like Google Drive, resulting in the fragmentation of member data. Furthermore, the manual updating of information by the administrative committee prolonged multiple projects.

To address these issues, we developed a website that centralizes information about our club, manages members' activities, and tracks ongoing projects.

This document outlines the work completed and is structured into four distinct chapters with the goal of delivering a dependable application:

* Chapter 1, “Functional Requirements”
* Chapter 2, “Non Functional requirements”
* Chapter 3, “UML”
* Chapter 4, “Conclusion”

Our report concludes with general summary and key takeaways.

Chapter I  
Functional Requirements

### I.1- Use cases:

During the early stages of development, it is essential to determine the system's needs, objectives, and functionalities. This process involves understanding the purpose of the system, how it will function, and how it will meet the club's needs. The identified functionalities are then quantified based on their importance and expected performance.

The use case diagram illustrates that the system will allow users to perform the following actions:

* Browse Pages
* Sign up
* Sign in
* View progress
* View overview page
* Manage members
* Manage events
* Manage meetings
* Manage assignments
* Manage e-sport teams
* Manage e-sport player

These functionalities are not necessarily ordered, but will be prioritized and quantified in the product backlog. The functional requirements of the system have been developed to ensure that the system meets the needs of the club and its members.

### I.2- Actors:

Actors are external entities that interact with the system being developed. For our website, we have identified three actors: Internet user, Member of the club, and Administrator of the club.

Internet user:  
The Internet user represents any user who wants to browse the site for informational purposes or register for an open event.

Member of the club:   
The Member actor refers to individuals who belong to the club and are interested in tracking their progress within it.

Administrator of the club:   
The Administrator actor is a member of the administration committee and is responsible for managing different aspects and activities of the club and its members.

|  |  |
| --- | --- |
| **Internet user** | * Browse website * Sign up |
| **Member** | * Browse website * Sign up * Sign in * View progress |
| **Admin** | * Browse website * Sign up * Sign in * View Overview page * Manage members * Manage Events * Manage e-sport teams * Manage Assignments * Manage Meetings * Manage e-sport Teams |

Chapter II  
Non Functional Requirements

As part of our software development process, non-functional requirements are essential in determining the general constraints of the system. Non-functional requirements define how the system should behave in terms of factors such as development costs, operational costs, performance, reliability, maintainability, portability, and robustness. Non-functional requirements are often expressed using adverbs or modifying clauses, such as "the system allows for a good and accessible browsing experience.

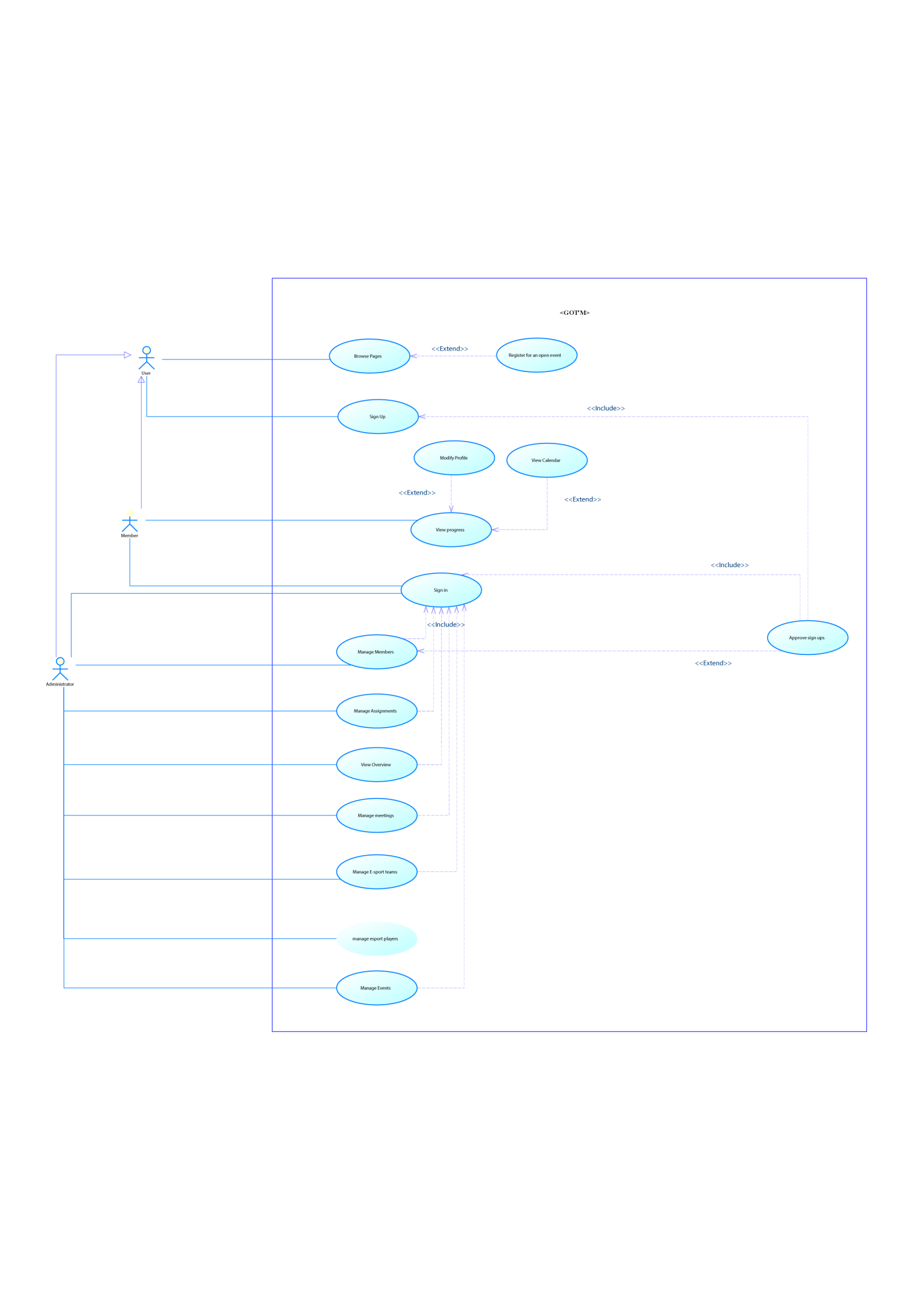
In our project, we are committed to delivering a user-friendly experience. Therefore, the non-functional requirements for our system include:

* Aesthetically pleasing interface:   
  The application's interface must be visually appealing, with a design that is eye-catching and professional. The navigation should be smooth and polished, creating an enjoyable user experience.
* Accessibility:   
  The system must be responsive and user-friendly across multiple devices and platforms, ensuring that all users can easily access and use the application.
* Efficient Performance:   
  The system must perform consistently and with minimal to no errors, ensuring fast and reliable performance. This will contribute to a better user experience and increased productivity.

By meeting these non-functional requirements, we aim to create an optimal user experience that satisfies the needs of the club's members and visitors to the website.

Chapter III  
UML

### III.1- Use Case Diagram:

 A use case diagram is a visual representation that depicts the various interactions between users and a system. Its primary objective is to identify the different tasks and actions that each user expects the system to perform. This diagram serves as a powerful tool for us, software developers to design and implement an efficient system that meets the users' needs and requirements.

### III.2- Global Class Diagram:

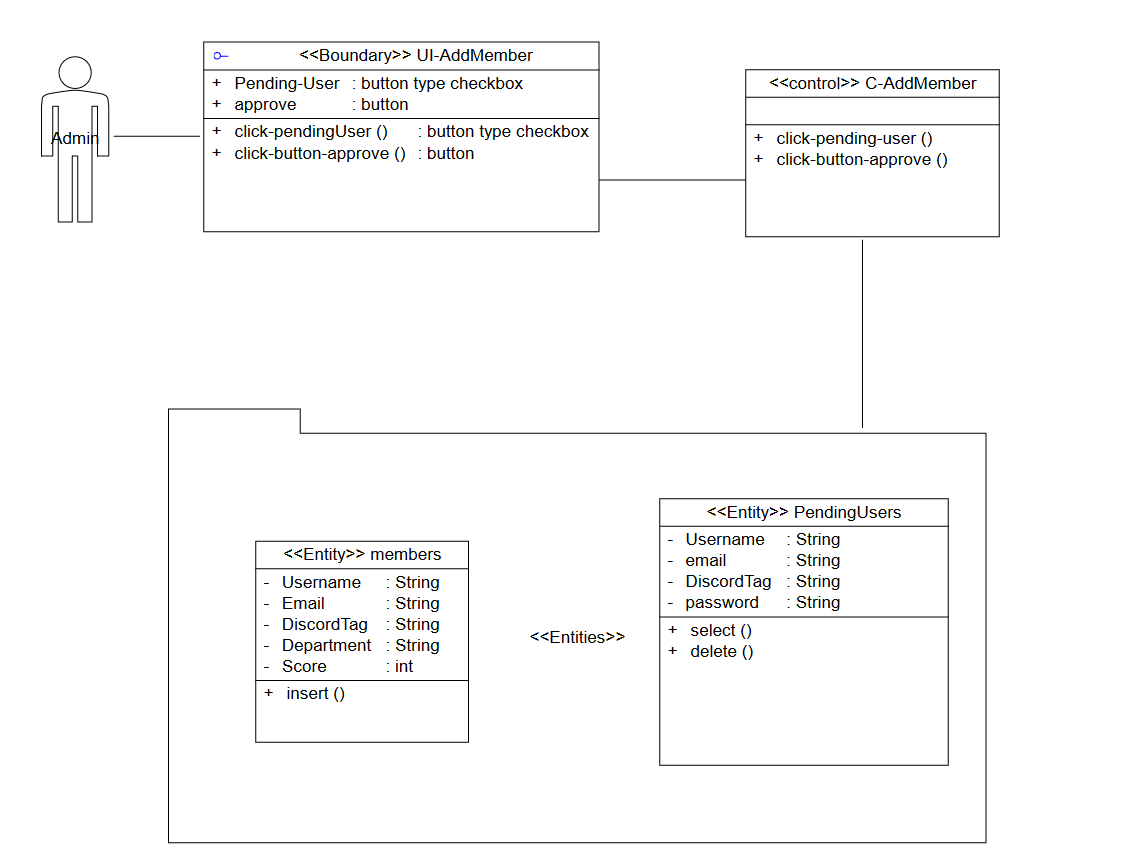
A global class diagram is a type of visual representation that helps us understand the relationships and interactions between the different classes in a system. It shows the various classes, their attributes and methods, and how they are related to one another. This diagram is a helpful tool for us as developers to get an overview of the entire system and to identify potential issues or areas for improvement. By analyzing the relationships and dependencies between classes, we can better design and implement the system to meet our needs and the needs of our users.

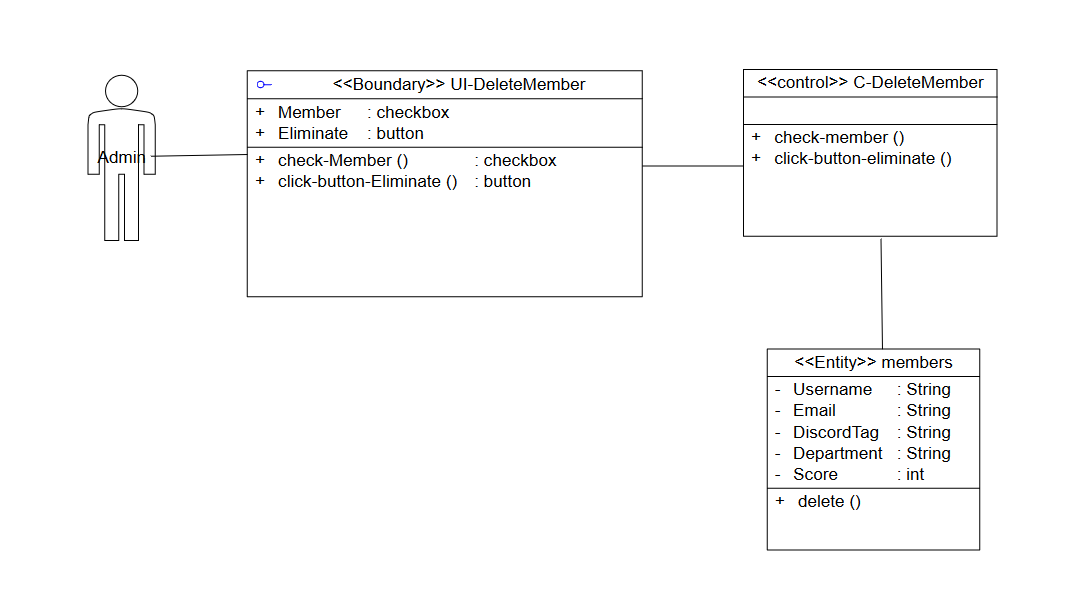
### III.3- Sequence Diagrams:

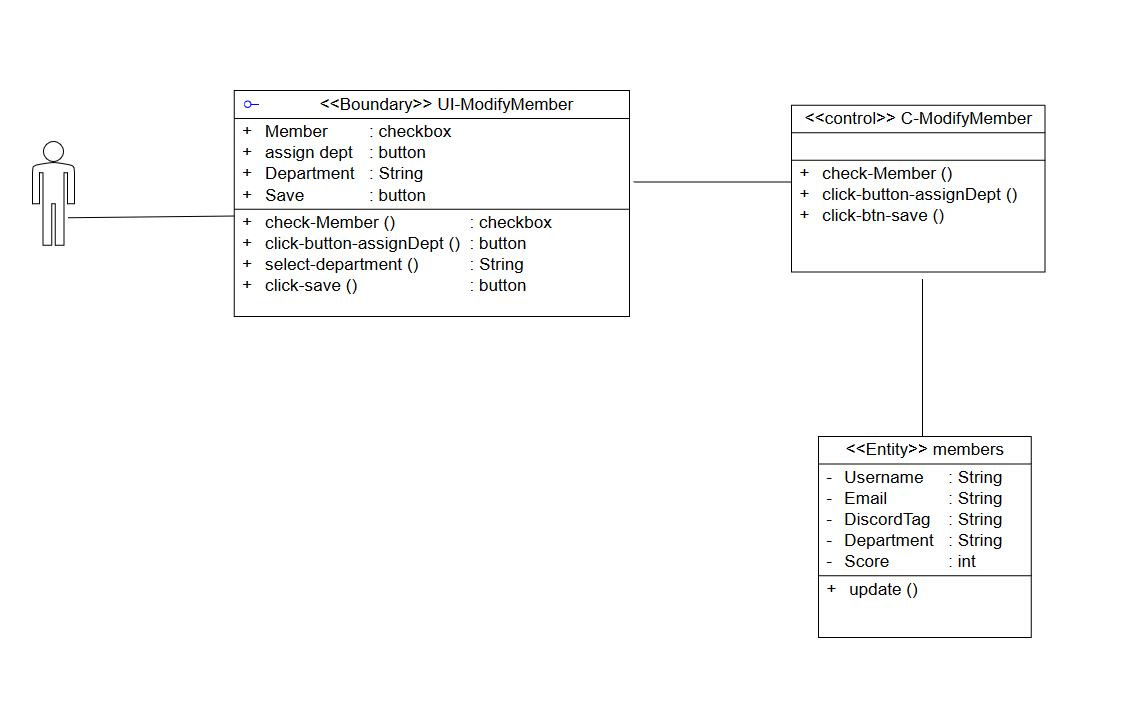
A sequence diagram is a type of interaction diagram in UML that represents the exchange of messages between objects or components within a system. Essentially, it shows the flow of communication between the different parts of the system as they carry out a specific task or process.

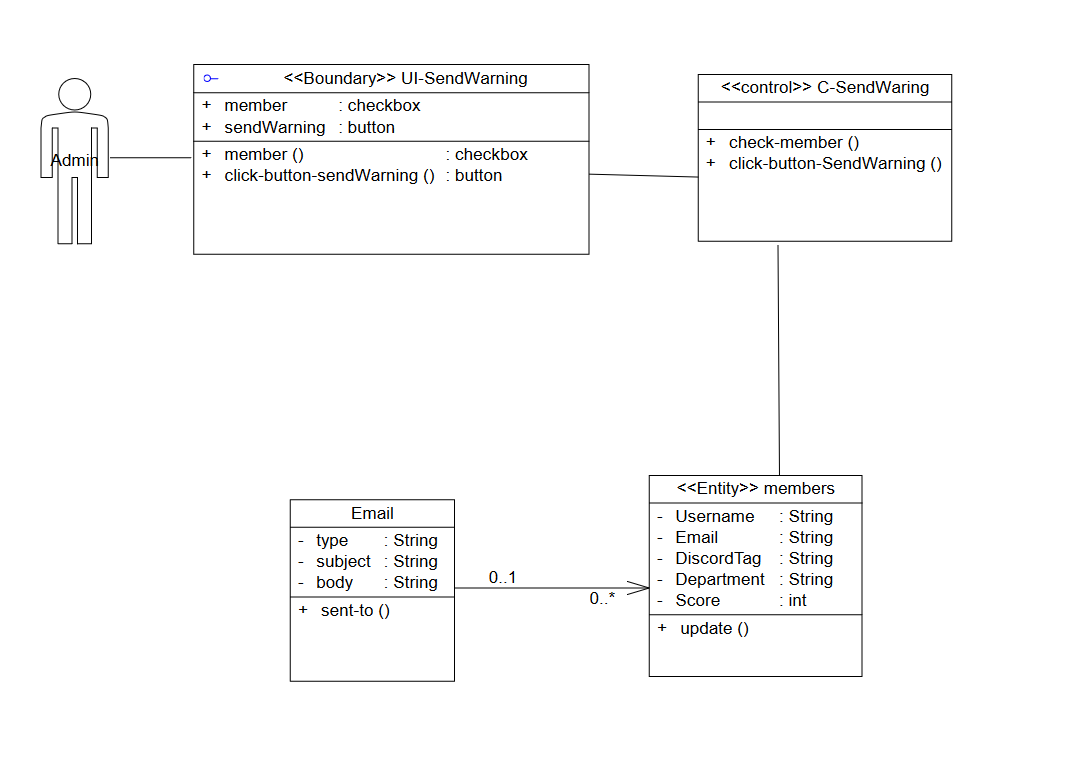
In other words, a sequence diagram allows us to visualize the sequence of interactions between the different parts of a system, which can be really helpful for understanding how the system works and identifying potential issues or improvements. It can also be useful for communicating with stakeholders and other members of the development team, as it provides a clear and concise overview of the system's behavior.

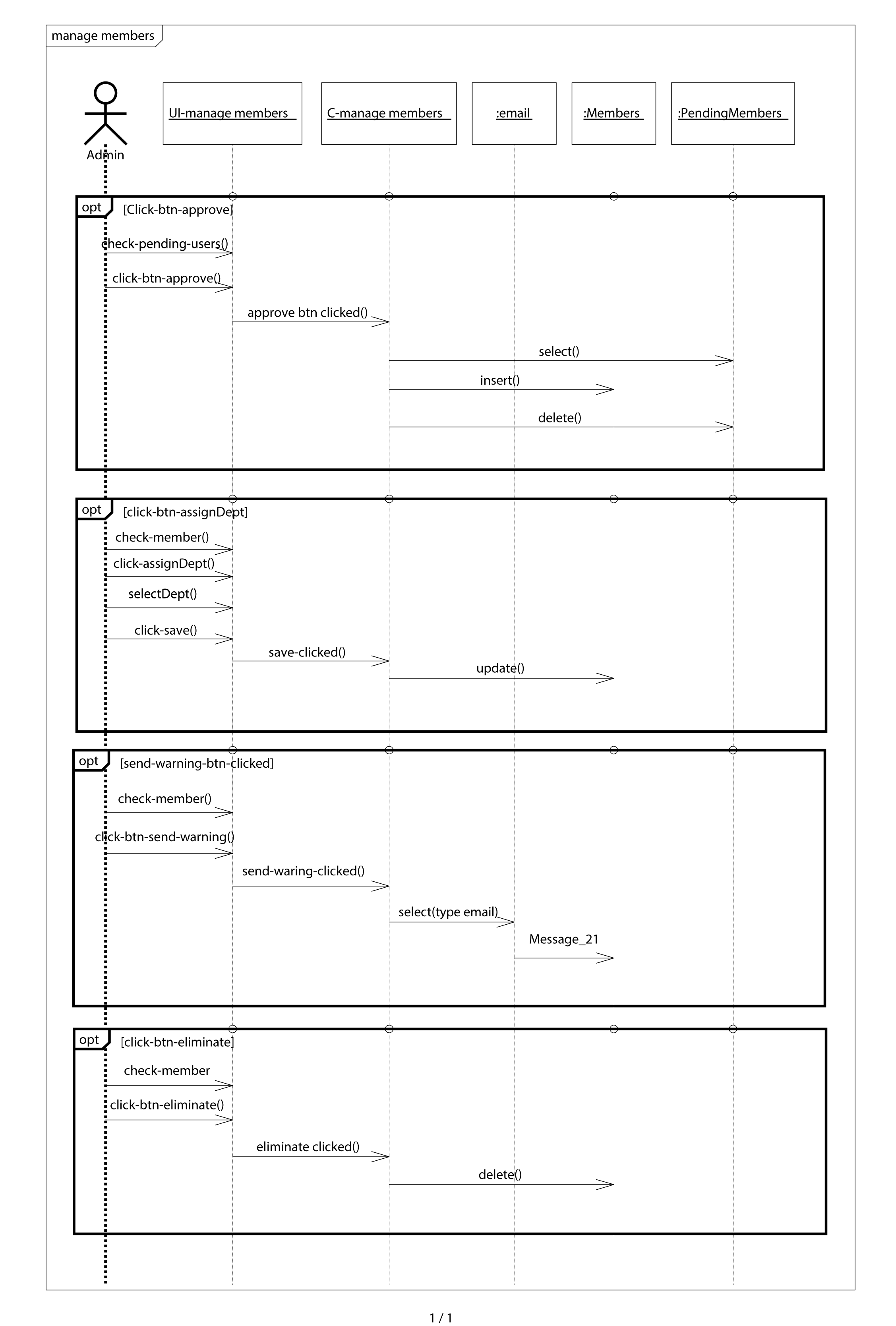
#### III.3.1- “Manage members” use case modeling:



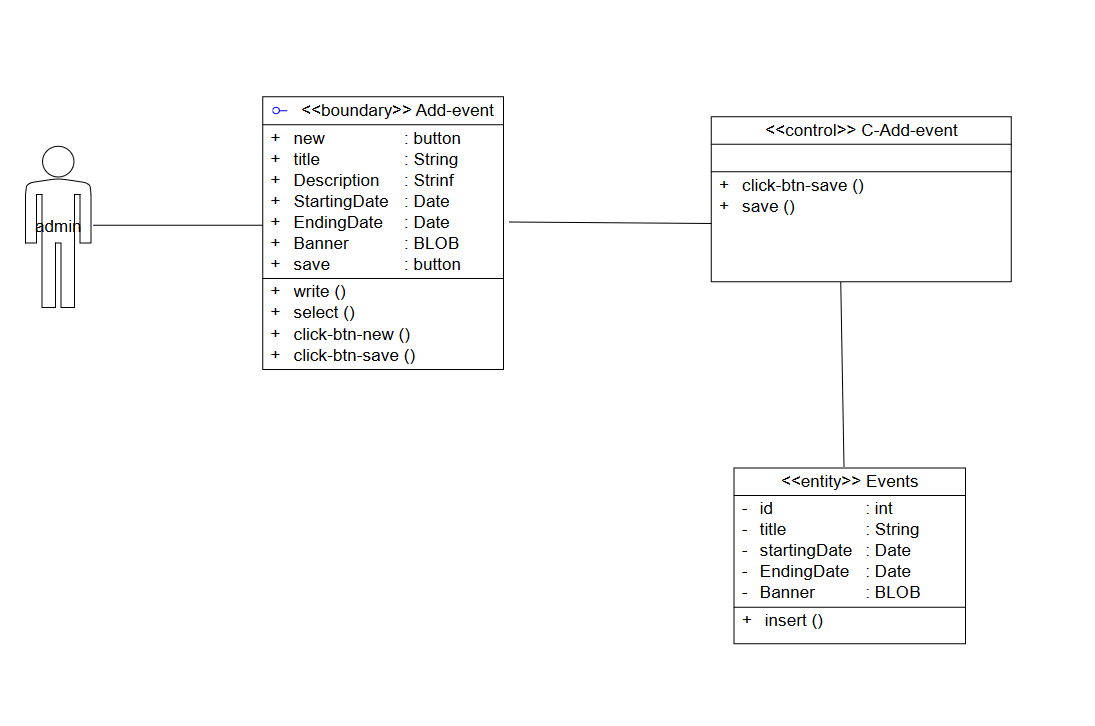


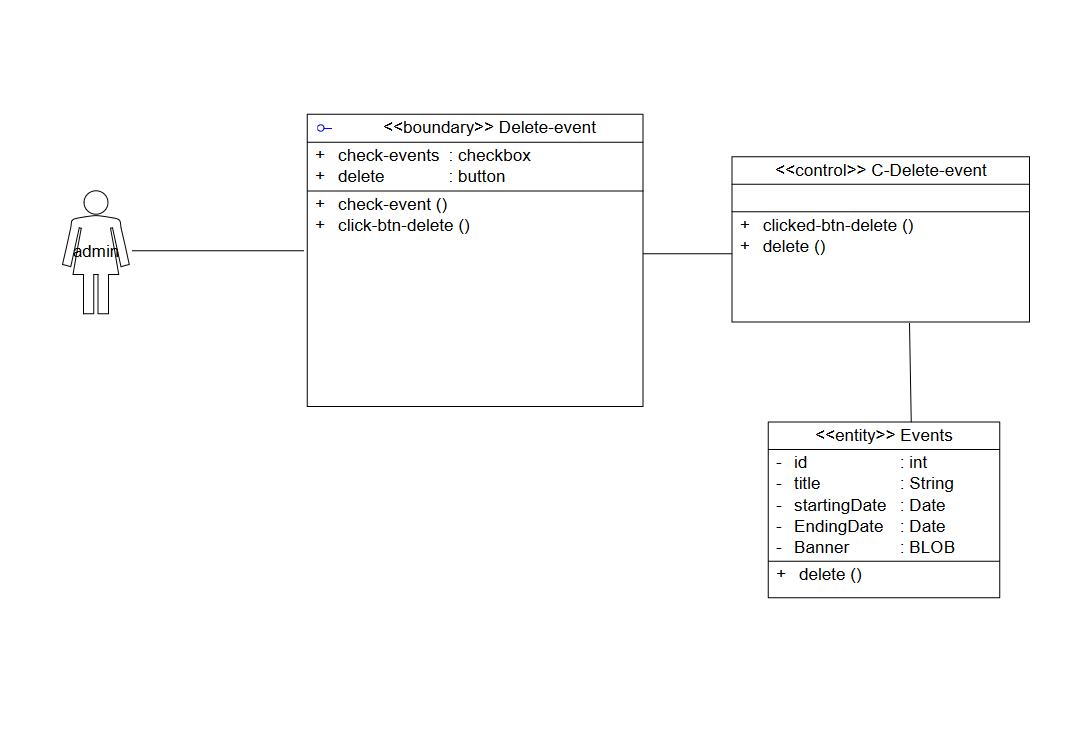


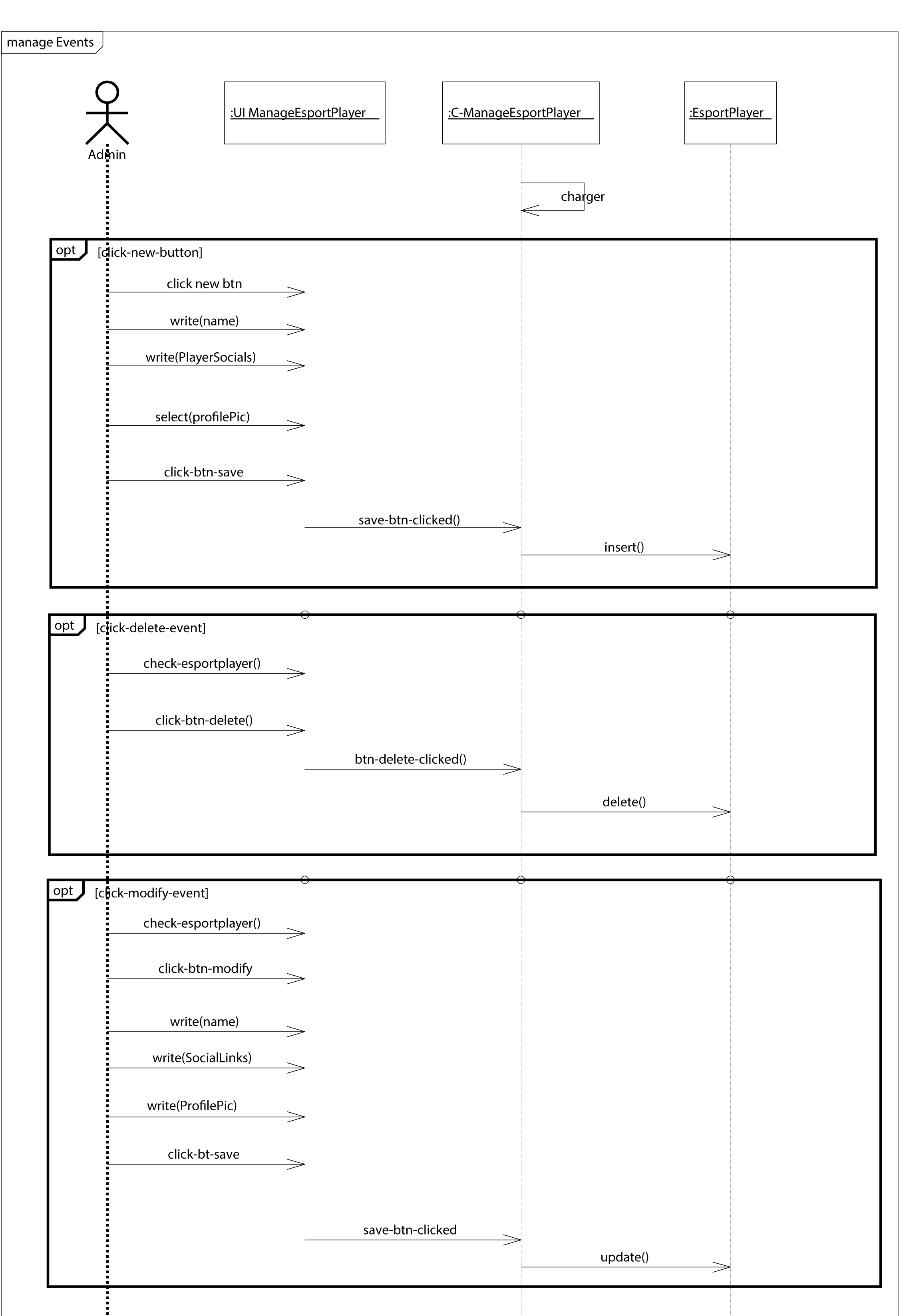
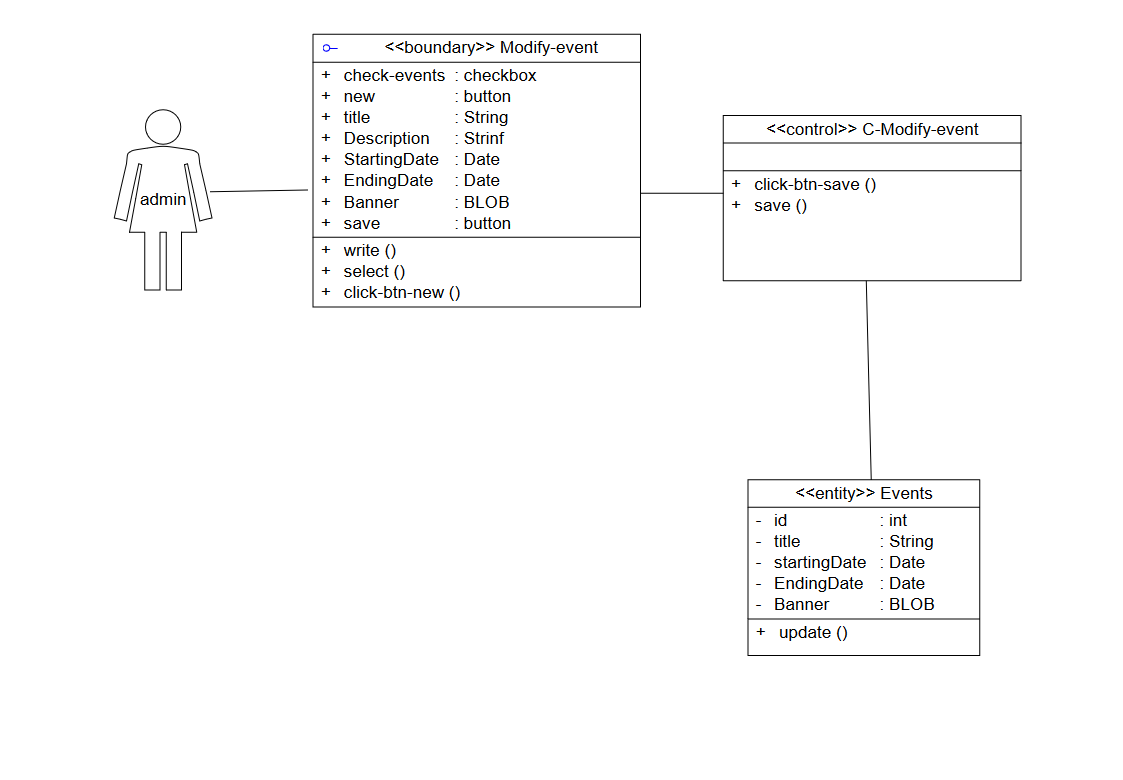




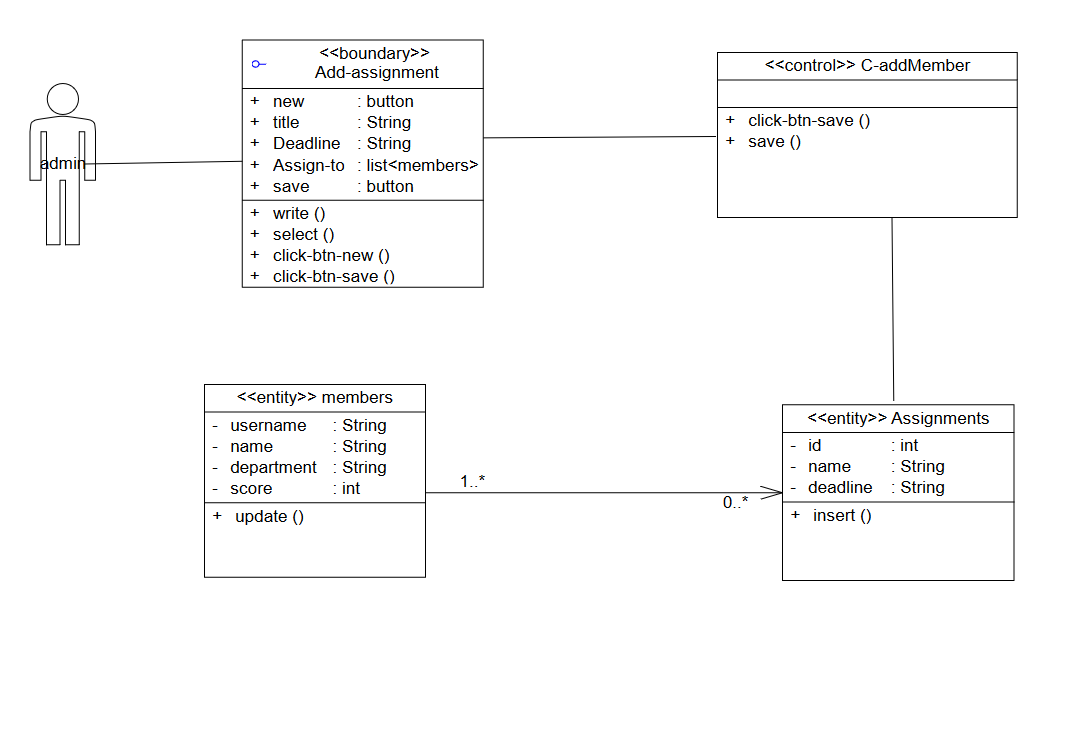
#### III.3.2- “manage events” use case modeling:

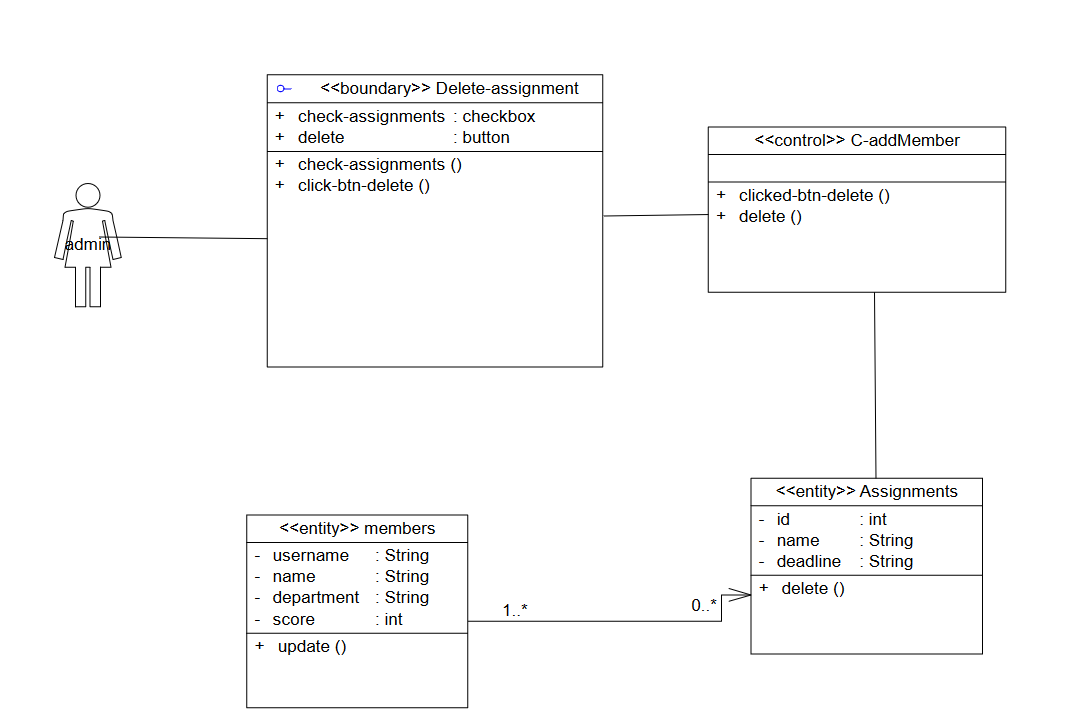


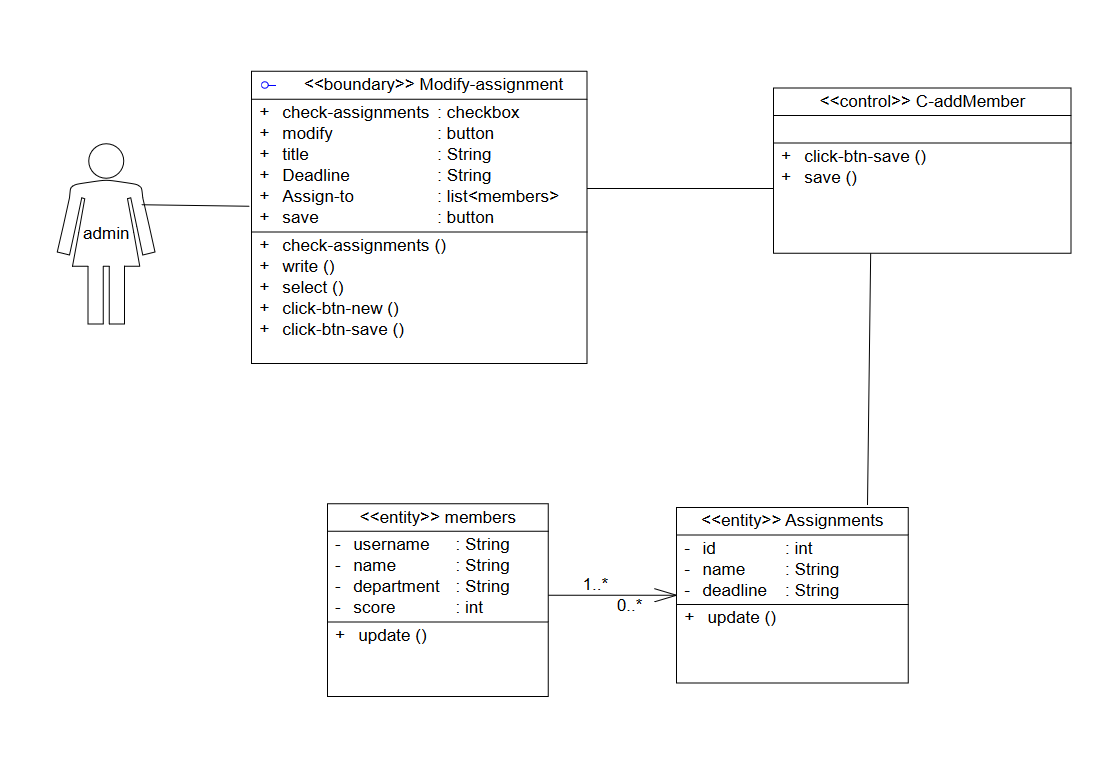


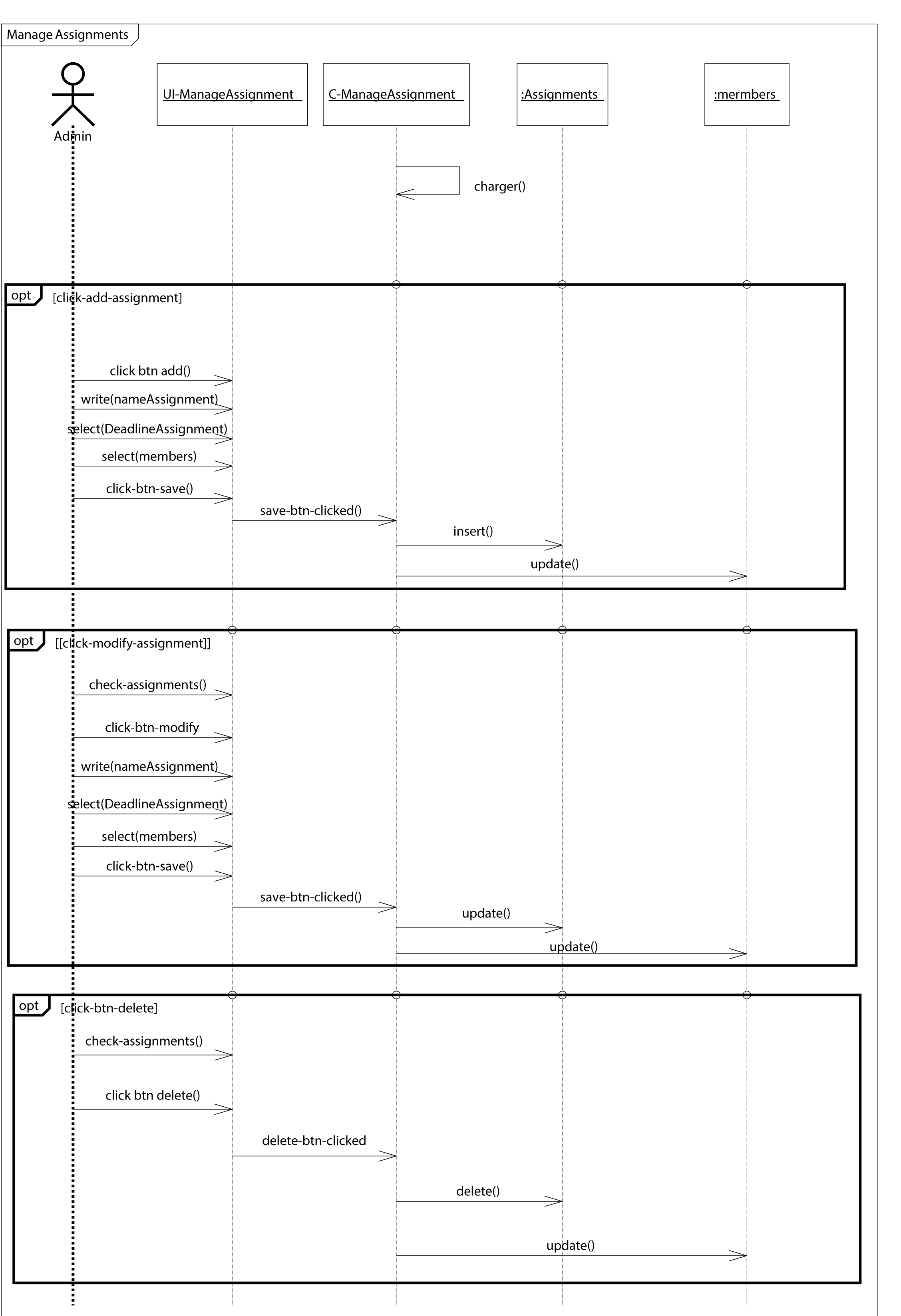


#### III.3.3- “manage assignments” use case modeling:

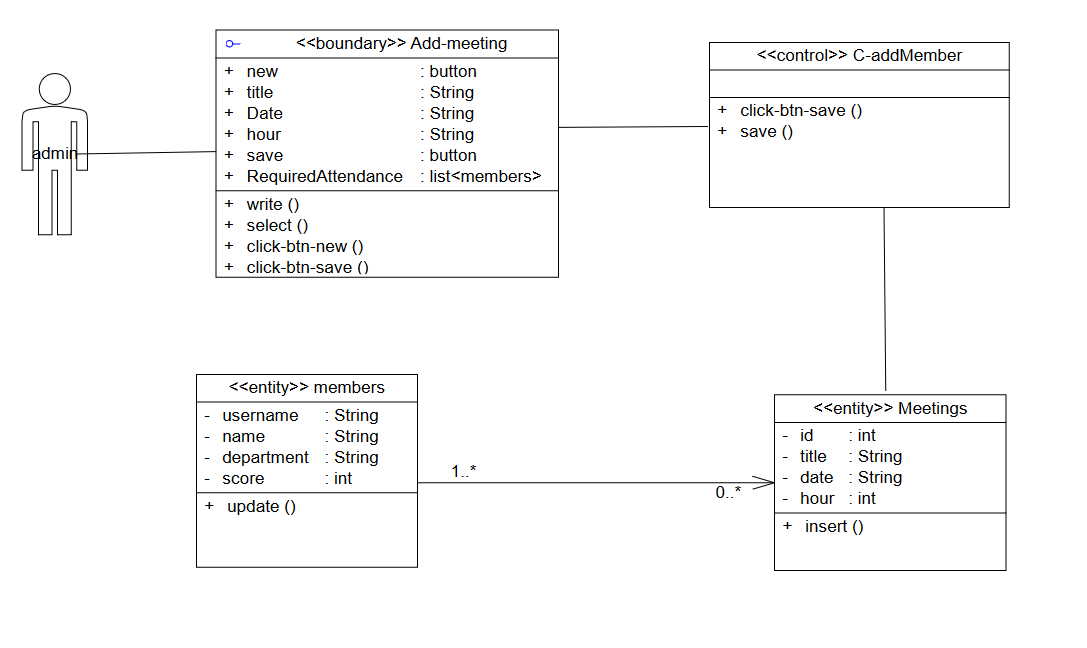


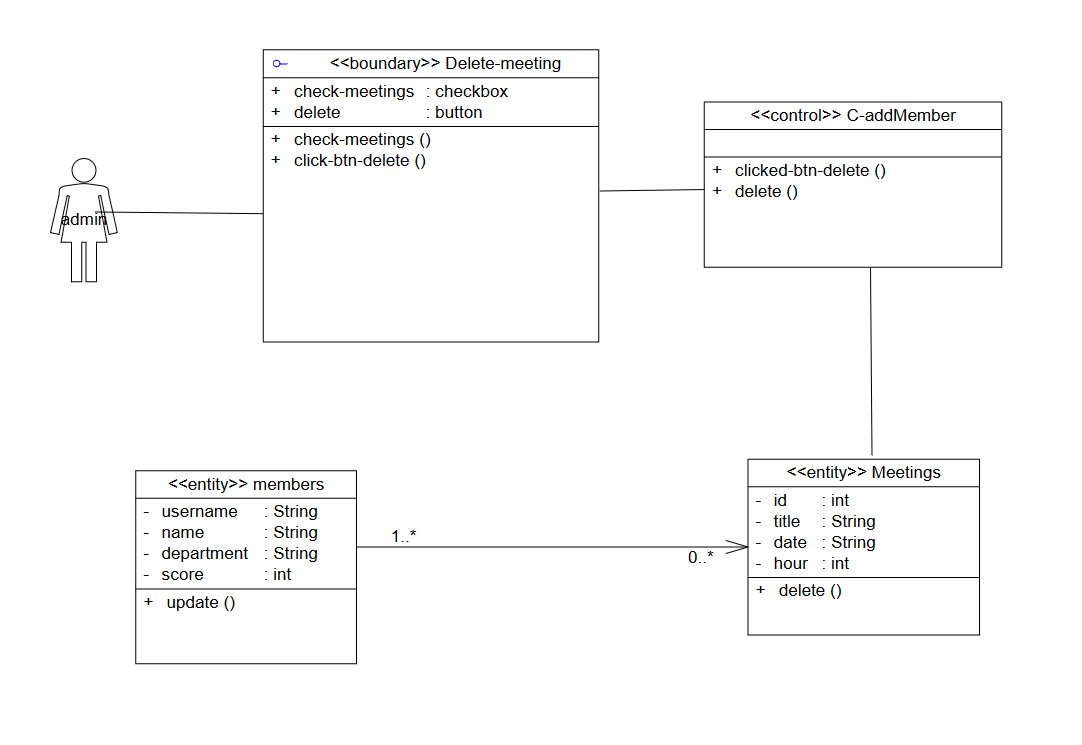


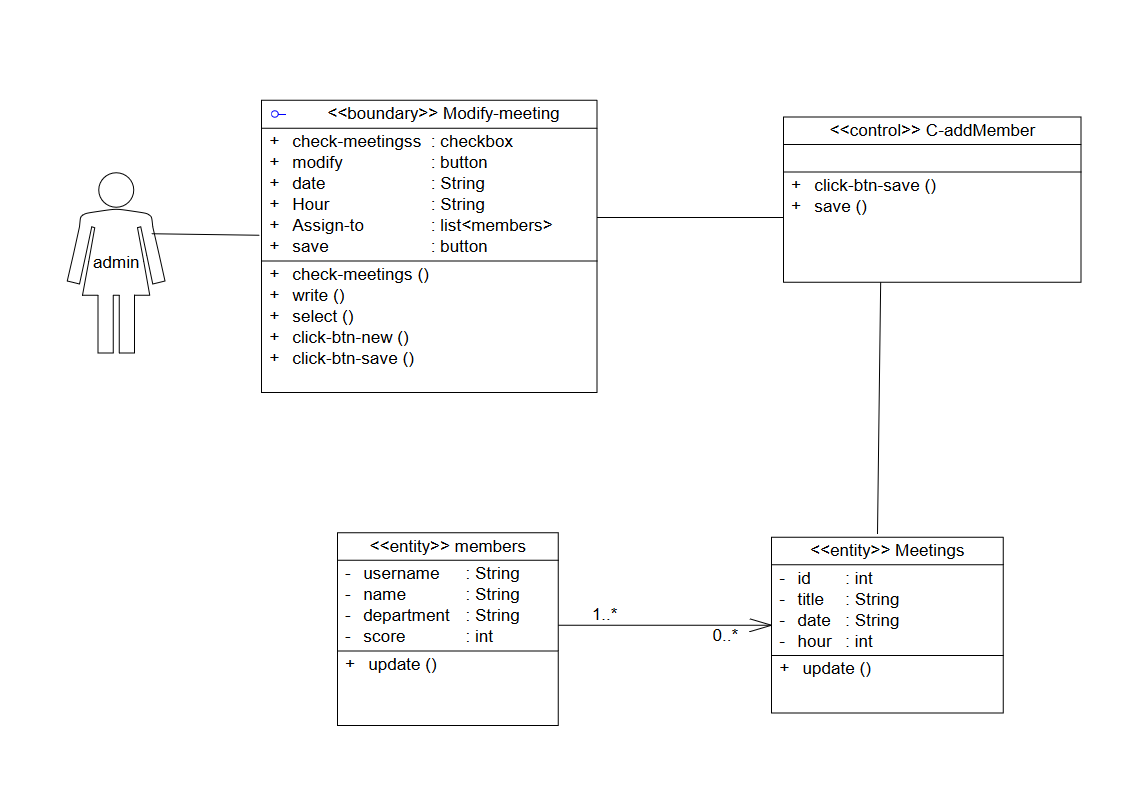


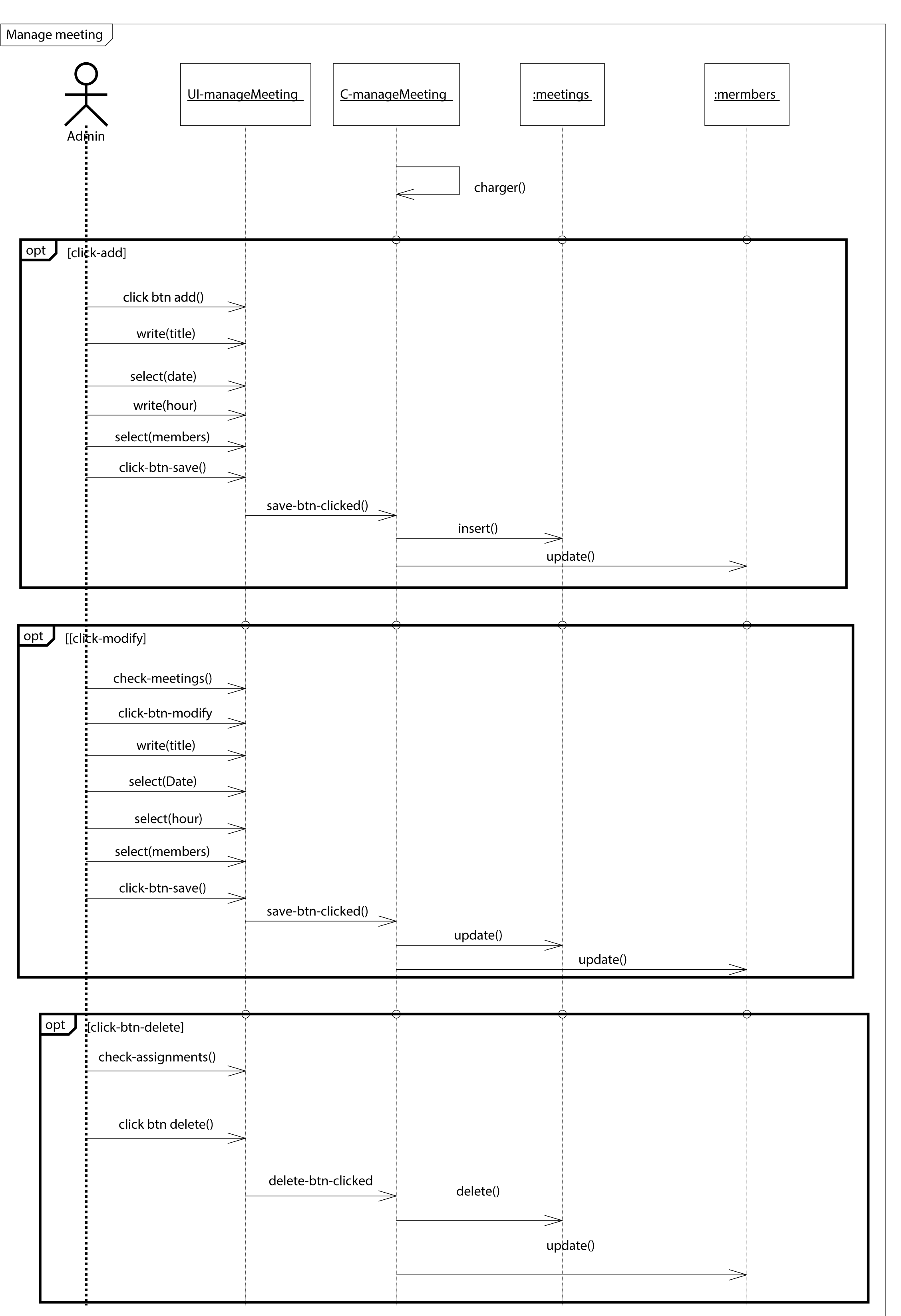


#### III.3.4- “manage meetings” use case modeling:

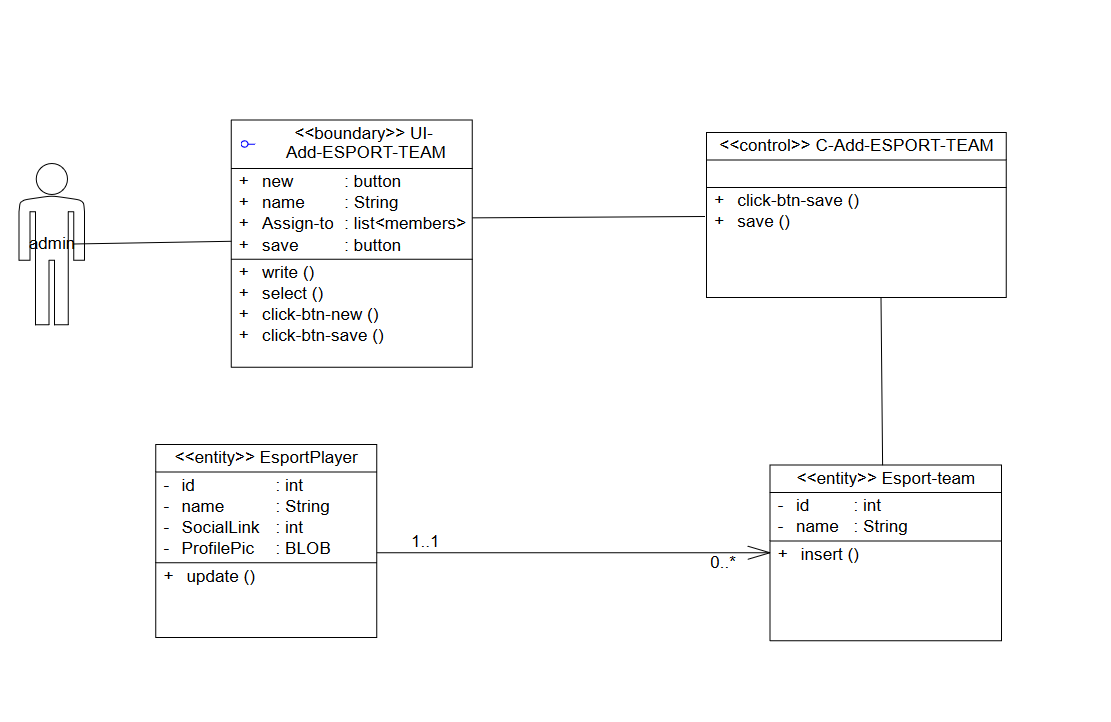


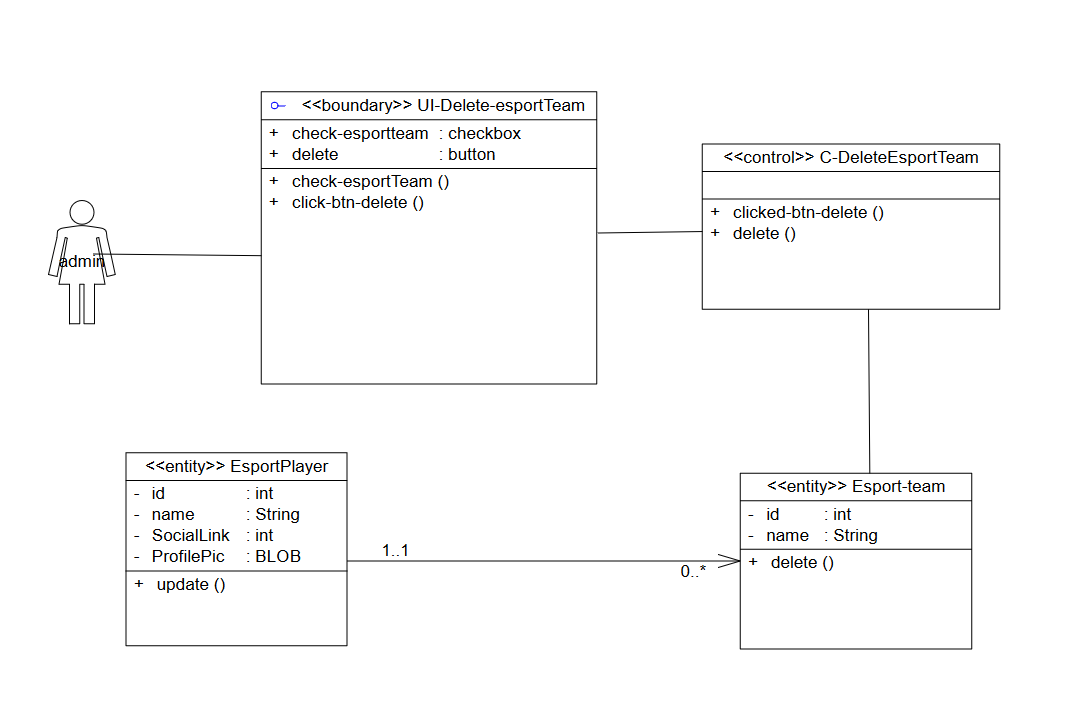


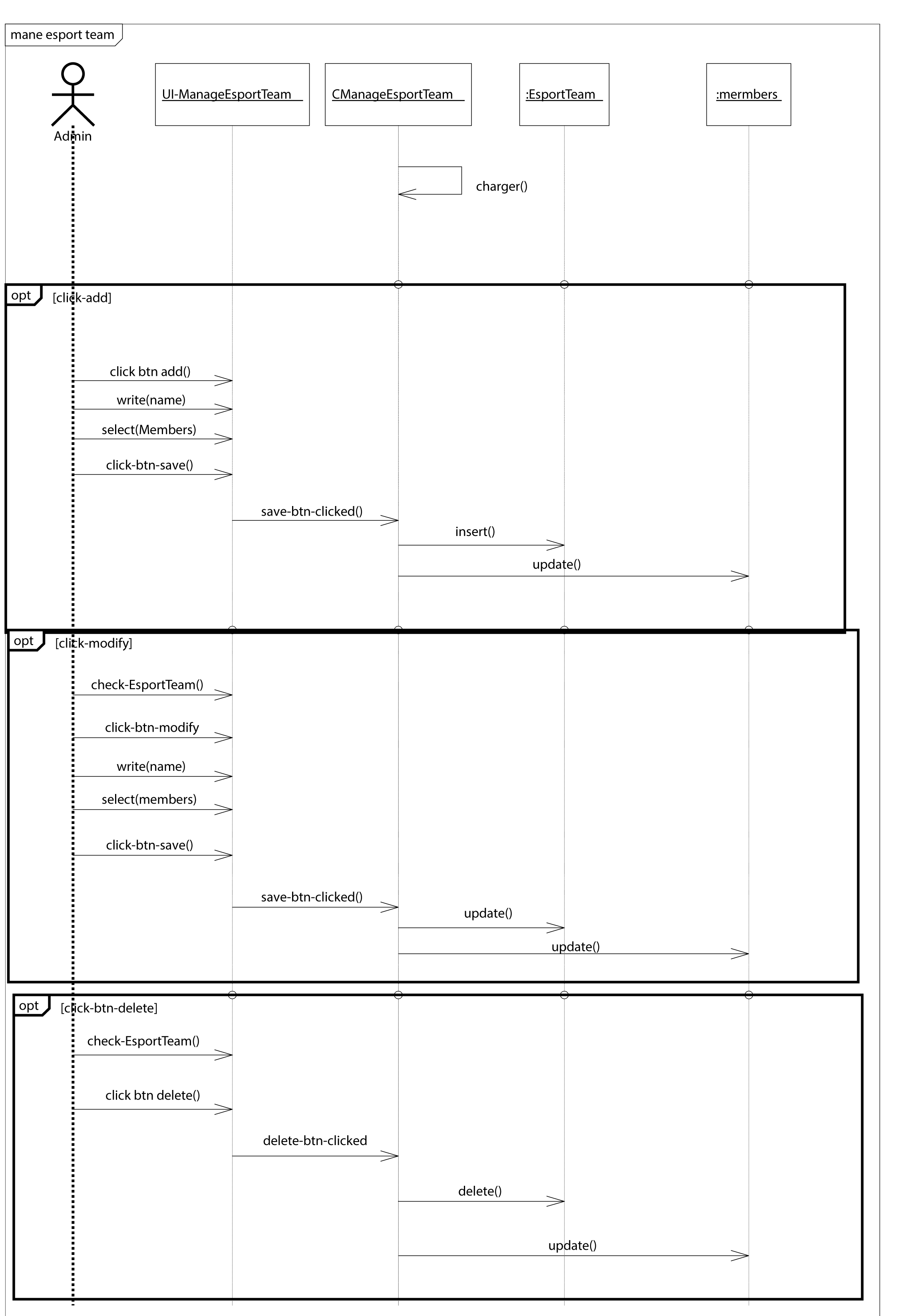
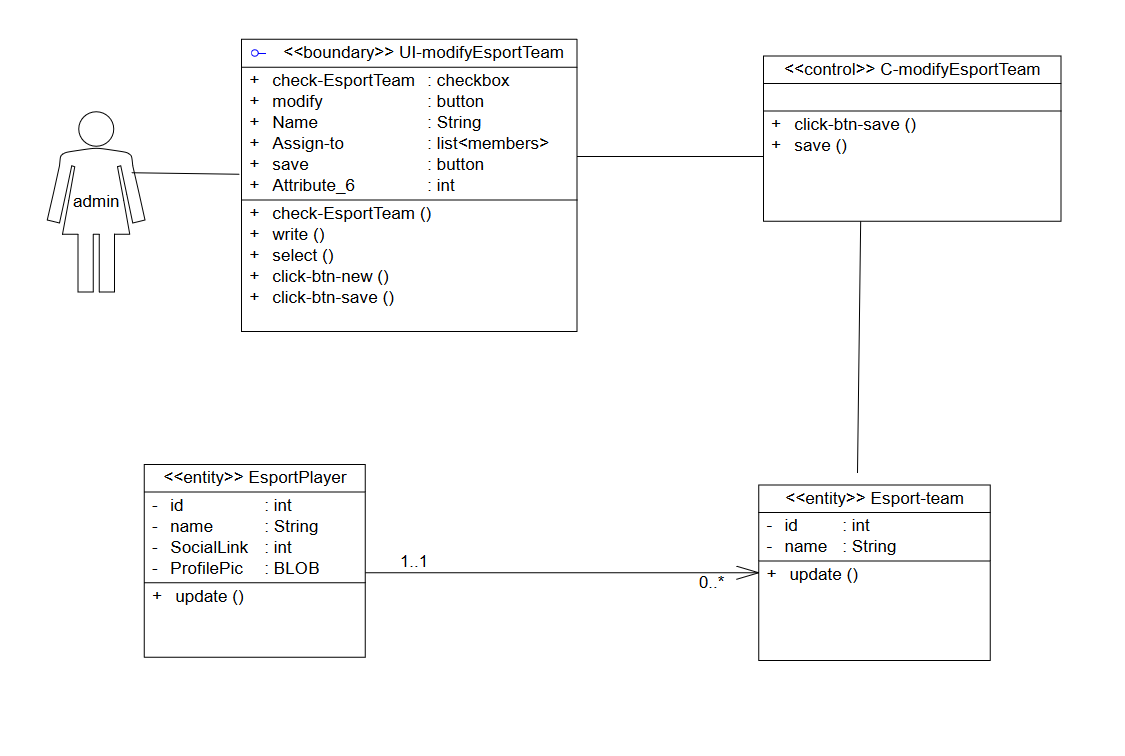




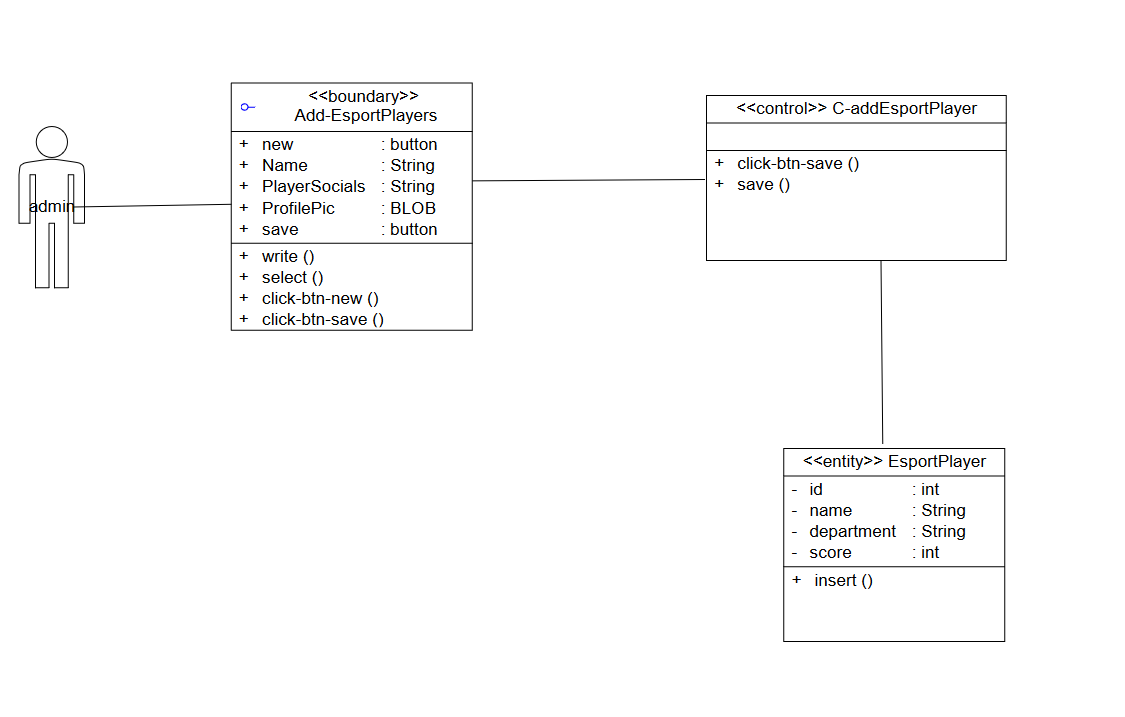
#### III.3.5- “manage e-sport teams” use case modeling:

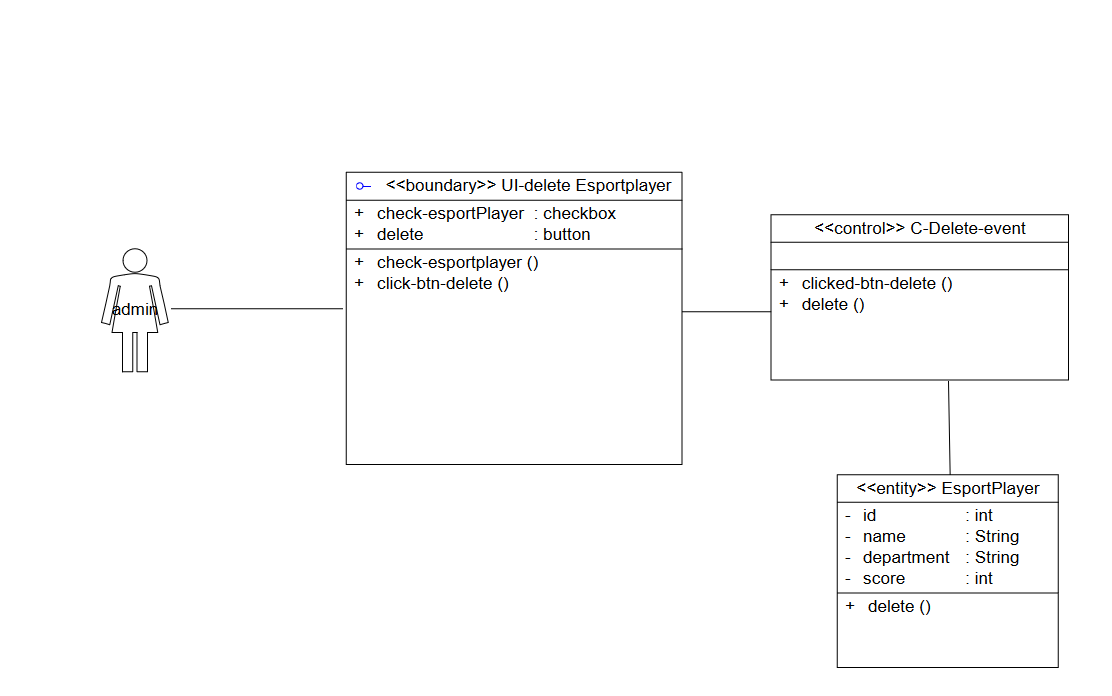


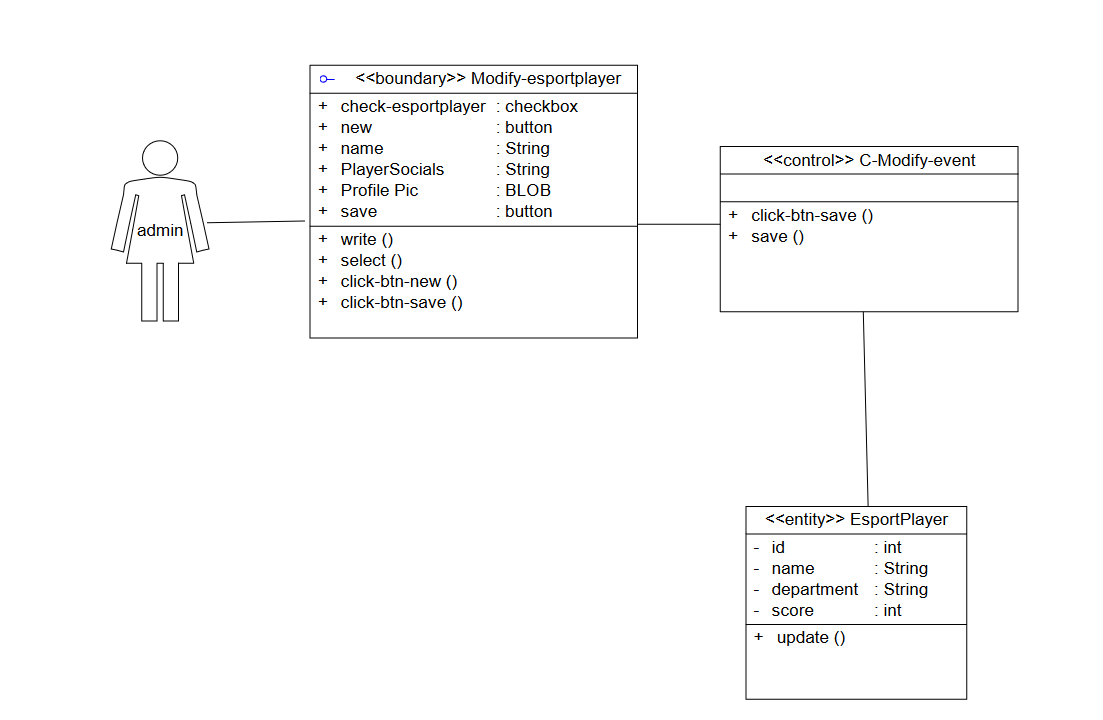


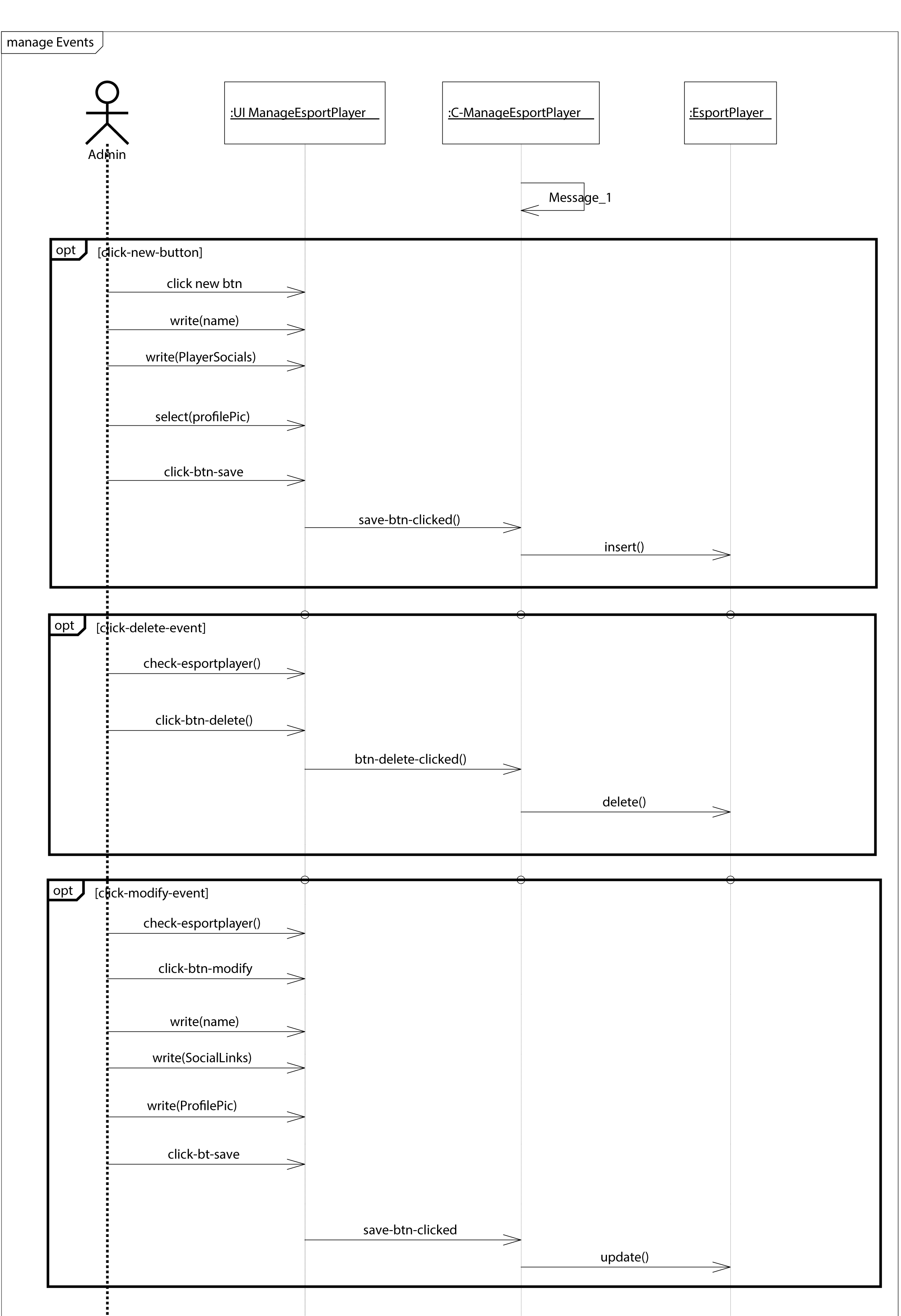


#### III.3.6- “manage e-sport players” use case modeling:





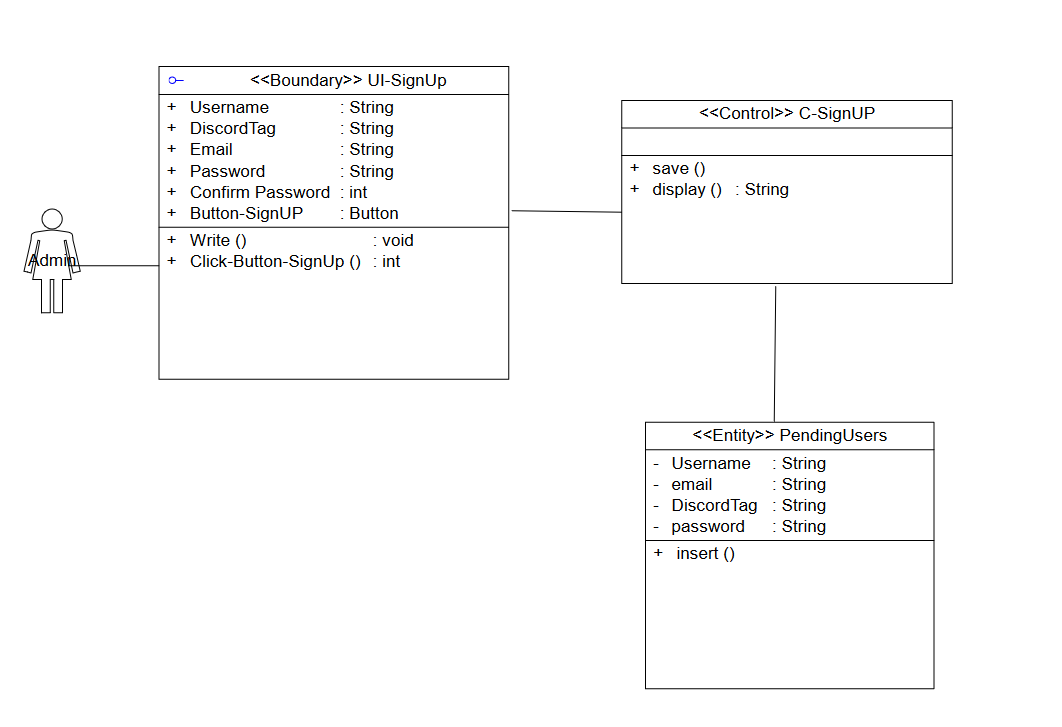


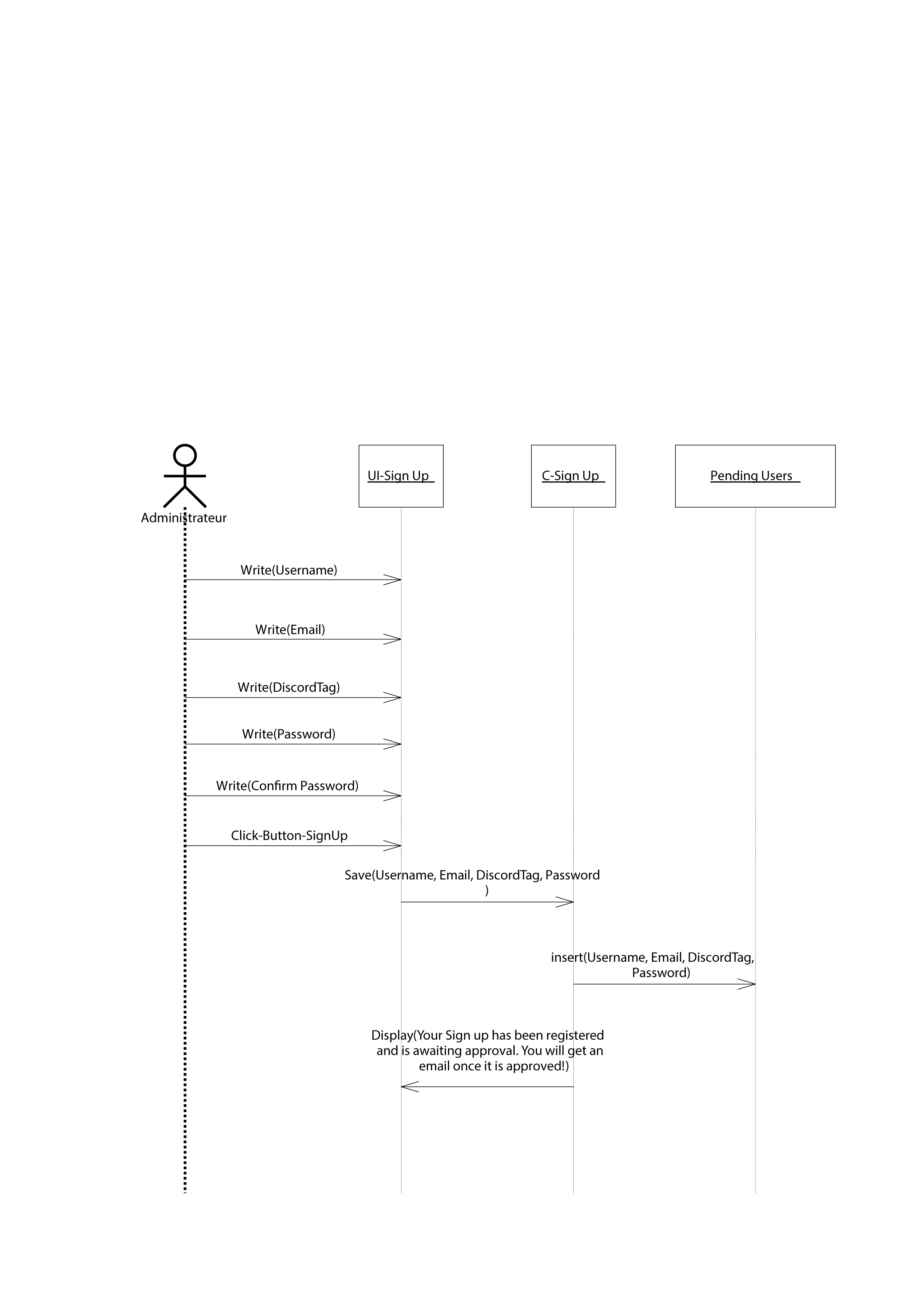


#### III.3.7- “view overview” use case modeling:

#### III.3.8- “sign in” use case modeling:

#### III.3.9- “sign up” use case modeling:





Chapter IV  
CONCLUSION

In conclusion, Chapter 1 "Requirement Specification" has laid out the necessary elements for building a well-built application that meets the club's needs. Functional requirements have been identified and prioritized in the form of expected performance values. Non-functional requirements have been defined to ensure an aesthetically pleasing interface, accessibility, security, and efficient performance. Actors of the system have been identified, including Internet users, club members, and club administrators. A use case diagram has been developed to determine what every user expects from the system. The product backlog has been established to prioritize the development of functionalities based on the identified requirements. The working environment has been defined, and UIX prototypes have been created to guide the user interface design. With these elements in place, the development team can proceed to the next phase of the project with a clear understanding of the requirements and the direction of the development process.