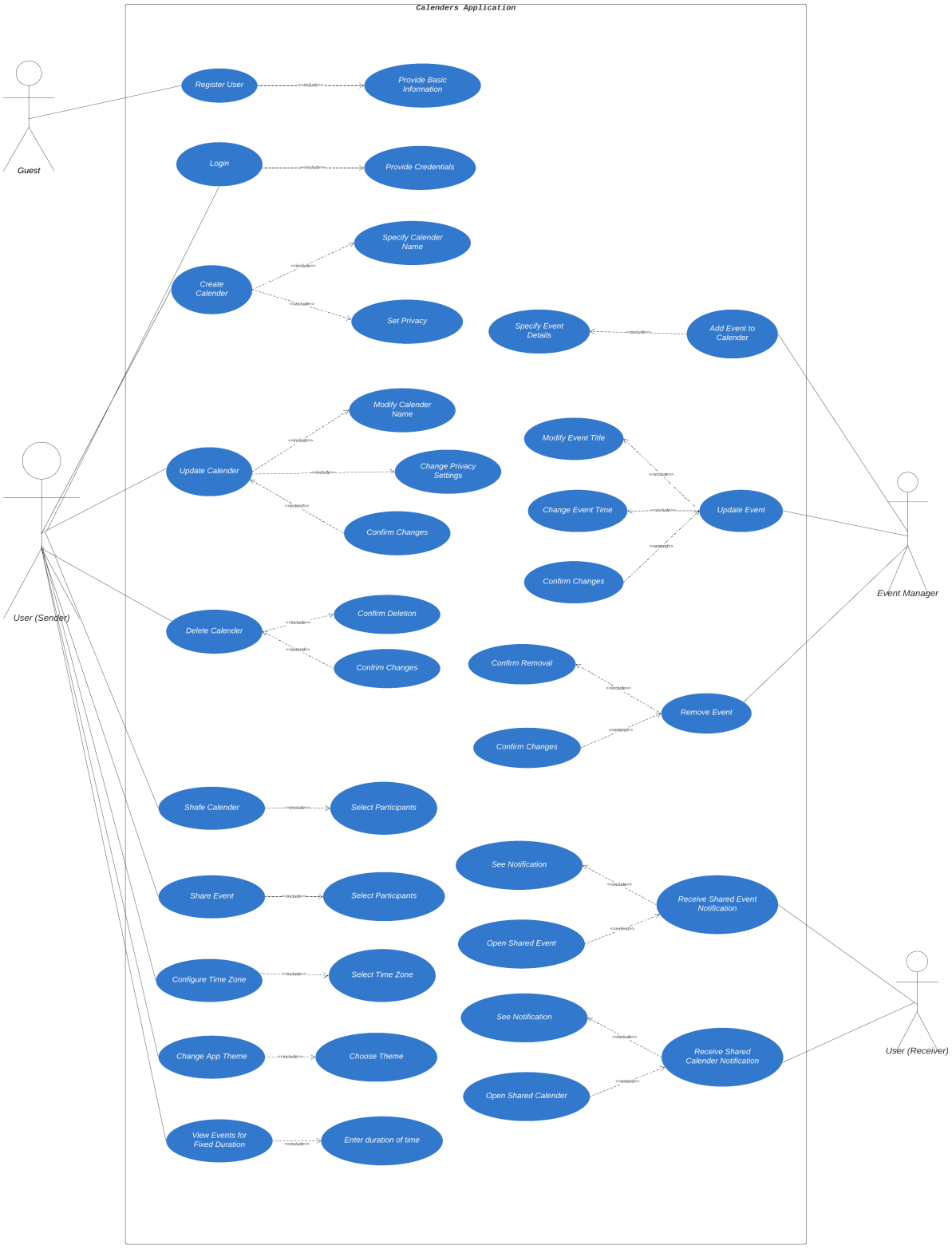
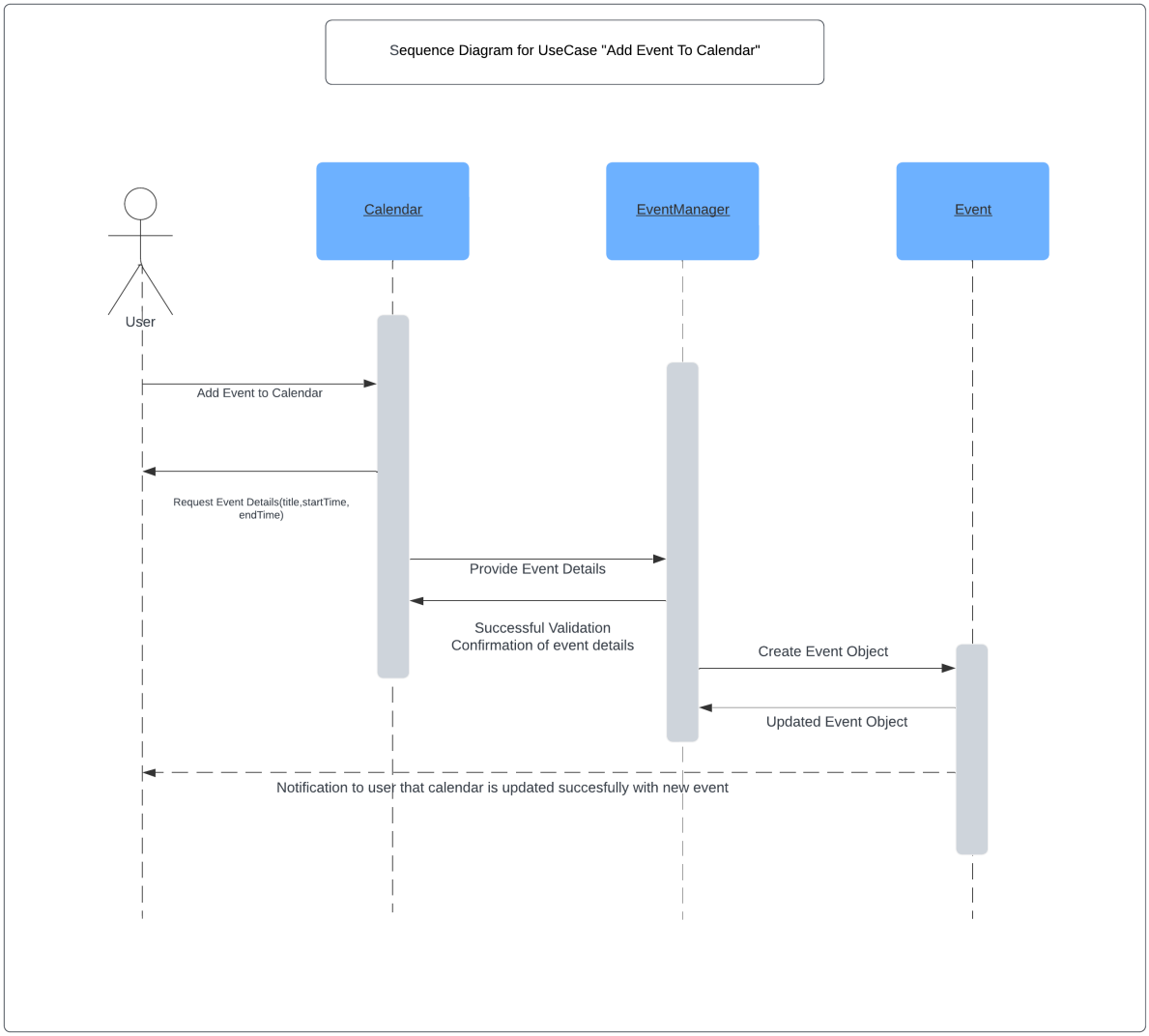
# 1. UML Use Case Diagram

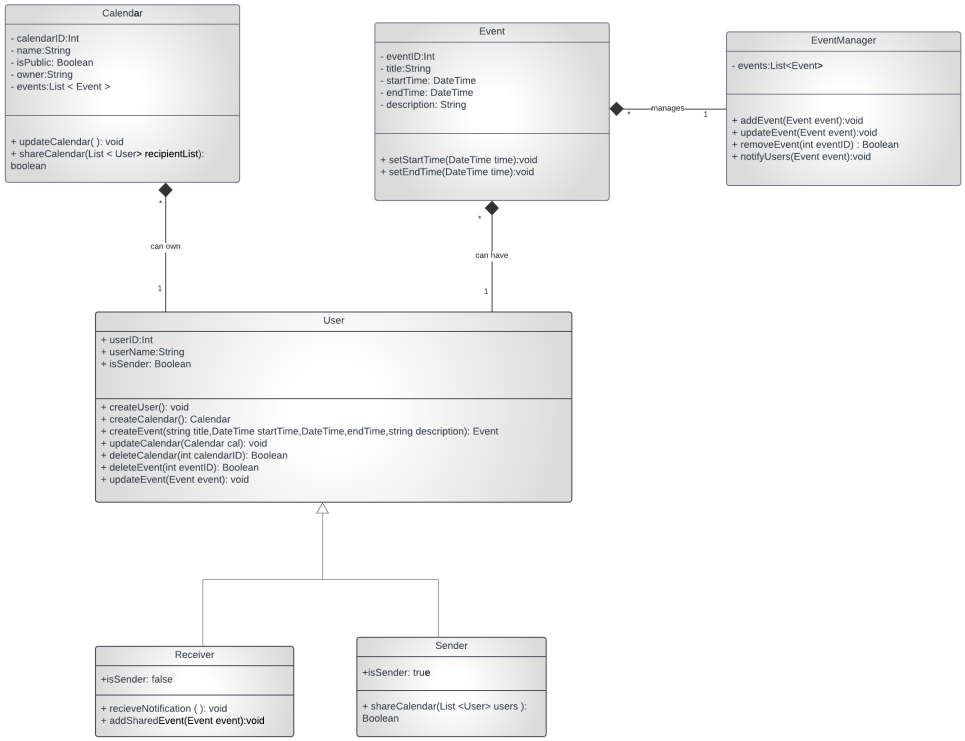


## 2. UML Sequence Diagram



# 3. UML Class Diagram

UML Class Diagram for Calendar Application



# Documentation

- The first approach that I used to design a Calendars Application is UML Use Case Diagram. This diagram is basically showing almost all the possible functional requirements of the Calendars Application. The primary user of this system is initially a guest which registers and then become a normal user of this app. The user interacts with the Calendars app to create, update and delete calendar and add particular events to the calendar. The user then can share calendar and events with other users based on the user selection. Upon successful sharing of calendar or event user( receiver) gets notified. The Receiver User will see the notification and can open it too. I used extend relationship here with the opening of shared event/calendar as this is users will to open the shared event/calendar or not. The secondary actors involved in our system is Event Manager that is responsible for managing these events ( add event to calendar, update event, delete event, notifies user of successful sharing, notifying receiver). And the other secondary actor involved in our system is User (Receiver). It can be a list of users too as a user can select multiple other users to share event or calendar.
- The second approach is to use “UML Sequence Diagram” as it demonstrate the dynamic interaction between different objects of different classes. I chose the use case “ Add Event to Calendar”. The primary actor involved in this is user who initiates the process of adding event to calendar by interacting with the Calendar. The Calendar Application requests user to enter details of then which are the validated and the event object is then created by the event manager and upon successful update, the event manager notifies he user about recent update.

- The third approach that I used is UML Class Diagram as it shows the classes and relationships between them with their multiplicities. The user class is classified into two different types that is sender and receiver based on their some different functionalities. It makes easier for the developers to make objects according to the specific role of the user interacting with the system.

## Flexibility to Future Changes

➢ The Event Manager Class used in our system for managing calendar events is the centralized point for handling events in Calendar. This class can play a vital role in adjusting future changes related to the events in the Calendar Application.

➢ The use of inheritance in the User class provides a useful way of incorporating different types of users required later in the User Class.

➢ The use of inner classes in the class diagram makes the tasks of adding different types of calendars, events easy. Extending the basic classes is the key to incorporate different types of base classes.

➢ The modular structure of class diagram allows the designer to accommodate different attributes or methods related to “notes and description” in the calendar or event.

➢ The user notification step which is done by the event manager provides a flexible way to make changes in the notification mechanism by using the centralized class “Event Manager”