Changes Made To Original Design

The original design had several calendar requirements that weren't supported so I decided to add them to make the calendar easier and better to use meeting the requirements for this assignment.

The original UML design lacked support for several required functionalities:

1. No Support for Multiple Time Zones:

- Requirement: Changing time zones should update event times accordingly.
- Issue: The original design did not associate time zones with users or calendars.
- Change: Added current_timezone and current_timezone_offset in User, along with time_zone in Calendar.

2. No Concept of Public vs. Private Calendars:

- Requirement: Users must be able to toggle calendars between private and public.
- Issue: Original design lacked a privacy setting for calendars.
- Change: Introduced private: Boolean attribute in Calendar and toggle_public_private() method.

3. No Support for Sharing Individual Events:

- Requirement: Users should be able to share specific events even from a private calendar.
- Issue: The original design only supported calendar-level sharing.
- Change: Introduced shared: List<User> in Event and methods shareEvent(user) and unshareEvent(user).

4. No Day/Week/Month/Year View for Events:

- Requirement: The system should allow visualization for different time frames.
- Issue: Original design had no methods for filtering events by day, week, or month.
- Change: Implemented view_calendar(user, year, month), which can be extended to support different views.

Structural Changes

1. User Management & Authentication
   Original Design:
   Users were loosely coupled with calendars.
   No authentication handling.

   New Design:
   Added CalendarsApp as the central manager:
   Handles user login, registration, and session tracking.
   Provides a command-based interface for user interactions.

   Added User.calendars: List<Calendar>:
   This ensures each user owns multiple calendars while still allowing shared access.

   Without a centralized application manager, users would be disconnected from authentication, making account handling difficult.

2. Calendar Class Modifications

   Original Design:
   Calendars had no owner field, making it unclear who created or controlled them.
   No support for public vs. private settings.

   New Design:
   Added owner: User in Calendar:

   Ensures every calendar is tied to a specific owner.
   Added toggle_public_private():

   Allows users to switch between public and private visibility.
   Added shareCalendar(user):
   Enables controlled sharing of calendars with specific users.

   Without ownership tracking, shared calendars could be modified by anyone.
   Public/private settings were crucial for access control.

Why These Changes Were Necessary

Feature Compliance:
The original design did not fully meet the project requirements (e.g., time zones, event sharing).

Scalability & Usability:
Without proper event conflict checks and time zones, the application would become difficult to use as you start to add a lot of users.

Access Control & Privacy:
The previous system lacked security measures, leading to potential data exposure.