Personal Calendar System

Overview:

Personal Calendar System is an electronic version of a calendar. It is designed so that users can have easy access to all their important dates and various events, tasks or appointments associated with them. It is meant to be a user-friendly electronic calendar which will have easy-to-use design and flexible features such that it will be suitable for any non-technical person to enter their tasks, events, memos or appointments. The system is meant to be for the personal use.

Feature List:

The following will be the features of our Personal Calendar System:

- Calendar view as per year, month and week.
- Events, tasks and memos can be added or edited
- Synchronization of events, tasks from various accounts
- Different View settings based on the view styles the user selects
- Event notification settings to set the type of alerts and notifications the user wants
- Search events and tasks based on the keywords entered by the user in search box
- Apply themes to the calendar based on the user choice
- List of public holidays based on the selected country

Requirements:

1) Calendar view:

- User can toggle between the view of Calendar based on year, month and week.
- When selected year, user will have the entire calendar of the year in a single view. He can switch between years based on his requirements.
- When selected month, user can see the detailed view of the month with days and date associated with it. He can switch between the months as per his requirements.
- When selected week, the user will have view of the week days and date associated with it. He can switch between the weeks of the month as per his needs.

2) Add events:

- User can add title of the event with an option to select a sticker to identify the type of event.
- Event can be:
    - Birthday
    - Meeting
    - Anniversary

    - Movies
    - Seminar
- He can choose ‘From’ and ‘To’ dates and time zone for the event.
- He can repeat the event:
    - One-time event
    - Daily
    - Every weekday (Mon-Fri)
    - Weekly
    - Monthly
    - Yearly
- User can set reminder:
    - 15 minutes before
    - 30 minutes before
    - 1 hour before
    - 1 day before
    - Or can customize it
- Use can add description and image to the event
- Event can be edited by clicking on the particular date the event was set in the calendar.

3) Add tasks:

- User can add title of the task.
- User can select due date such that due date cannot be before current date.
- He can set reminder:
    - Off
    - On due date
    - Or can customize it
- Priority of the task can be set by the user. It can be
    - High
    - Medium
    - Low
- Description and image can be added to the task.
- Task can be edited by clicking on the particular date the task was created in the calendar.

4) Add memo:

- User can attach memo to the current date.
- Memo can be edited/ deleted by clicking on the date that memo was created.

5) Synchronize events and tasks:

- User can synchronize events and tasks from various accounts like “Gmail” and “Yahoo”.

6) Settings:

- User can change first day of the week to:
    - Locale Default
    - Saturday

    - Sunday
    - Monday
- User can lock event times and dates to the selected time zone. Times and dates will not change even if the user move to another time zone.
- User can either show week numbers or not.
- He can also opt for seeing weather of the respective day and date.

7) Set Notification type:

- User can choose:
    - Via push notification
    - Via Alert

8) Search functionality:

- User will have search bar at the top where he can directly search for the events and tasks added to the calendar by typing appropriate keywords.
- List of tasks or events that matches the keywords will be shown in the form of the list to the user.

9) Apply themes:

- User can apply several themes based on his liking. User can choose:
    - Dark theme - will display calendar in the dark background with light colored texts.
    - Light theme – will display calendar in light background with dark colored texts.

10) See public holidays:

- The user will be able to see the list of public holidays based on the list of holidays that has been entered by the administrator for the particular country.

Use cases:

Following are some important use cases for personal calendar system.

1. Check Calendar:
    a. For this use case user is actor and he/she launches the application
2. Change Calendar View
    a. User will choose to view calendar yearly, monthly or weekly
3. Create/Add new event:
    a. Use case of adding new event to calendar.
4. Edit event in calendar
    a. Edit already created event in calendar.
5. Delete event in calendar
    a. Delete unnecessary event from calendar.
6. Create/Add new task
    a. Use case of adding new task to calendar.
7. Edit task in calendar
    a. Edit already created task in calendar.
8. Delete task in calendar
    a. Use case for deleting task created earlier.
9. Check personal tasks in calendar
    a. User will check personal tasks.
10. Check personal events
    a. User will check personal events.
11. Create/Add memo
    a. User will create new memo for the current date
12. Edit Memo
    a. User will edit the memo
13. Delete Memo
    a. User can delete the created memo
14. Sync Events/ Tasks
    a. This use case syncs the events/tasks from personal calendar to email.
15. Change settings
    a. User can change first date of week and time zone depending upon his location.
16. Set alert and notifications
    a. Set alerts for event and notification type as email or push notification.
17. Search event
    a. Search facility provides searching of event from calendar.
18. Search task
    a. Search facility provides searching of task from calendar.
19. See public holidays
    a. User can see public holidays depending upon location.
20. Change theme
    a. User can change theme of the calendar

Use cases in fully dressed format:

1) Create/Add new event

Use Case Name	Create/Add new event
Level	User goal
Brief description	When user wants to create any event in his/her personal calendar he will click on create event and give time, date and alert type for that event.
Primary Actor	User
Stakeholders	- User: Want to ensure successful creation of event and details of event has been entered correctly.
Triggering event	When user chooses to add event in calendar
Preconditions	- The User must exist. - The User should order create an event.
Post conditions	- Event created successfully note. - Alert email or push notification should receive to user for event.
Flow of event	- User creates an event. - User must give valid date and time for event - User will select alert type for event. - User will receive notification at the time of event.
Exception Condition	If user give invalid date and time event notification will not work.

2) Sync Events/Task

Use Case Name	Sync Event/Task
Level	User goal
Brief description	When user wants to sync any event/tasks in his/her personal calendar, with his accounts, he will click on sync and the events/tasks will be synced to the calendar
Primary Actor	User
Stakeholders	- User: Want to ensure successful synchronization of event/tasks from his accounts to the personal calendar properly so that he can have easy access of all his events/ tasks to the one system
Triggering event	When user decides to sync his data
Preconditions	- The User must exist. - The User must have added some task/events in his accounts.
Post conditions	- Synchronization successful note. - Alert email or push notification should receive to user for events/tasks.
Flow of event	- User synch his data. - User have some added list of tasks/events in his accounts - User will receive notification at the time of event/task.
Exception Condition	If no data is stored by the user in his acccount.

3) Check Memo

Use Case Name	Check Memo
Level	User Goal
Brief Description	When the user would like to check an existing memo the user will first press check calendar, then press check memo.
Primary Actor	User
Stakeholders	User: the user has a vested interest in the accurate reporting and digesting of any pertinent memos and their relevant information.
Triggering Event	When user presses check memo button
Preconditions	The user must exist. The user must have received an existing memo in the past.
Post Conditions	Memo is displayed on the screen. Option to edit and option to delete memo should appear.
Flow of Events	-User receives memo -User presses check calendar -User presses check memo -Memo information is displayed
Exception Condition	If the user has not received any memo, the option to check an existing memo would not exist.

4) Search Events

Use Case Name	Search Events
Level	User Goal
Brief Description	When the user would like to search his or her calendar for existing events, either for quick access or for an overview of any relevant events.
Primary Actor	User
Stakeholders	User: the user has a vested interest in receiving accurate and relevant event information.
Triggering Event	When the user presses check calendar
Preconditions	-User must exist -Events must exist in the user calendar
Post Conditions	All events matching the inputted search keyword(s) is displayed.
Flow of Events	-User presses check calendar -User presses the search events button -User inputs accurate keywords relevant to at least one existing event
Exception Condition	No events will return from the search events function if the user inputs inaccurate search parameters (such as due to misspelling) or if the user does not have any existing events in his or her calendar.

5) Check Personal Tasks:

Use Case Name	Check Personal Tasks
Level	User Goal
Brief Description	When the user would like to check an existing personal task the user will first press check calendar, then press check personal tasks.
Primary Actor	User
Stakeholders	User: the user has a vested interest in the accurate reporting and digesting of any pertinent tasks and their relevant information.
Triggering Event	When user presses check task button
Preconditions	The user must exist. The user must have received an existing personal task in the past.
Post Conditions	Task is displayed on the screen. Option to edit and option to delete task should appear.
Flow of Events	-User has a task set in his or her calendar -User presses check calendar -User presses check tasks -Task information is displayed
Exception Condition	If the user has not set any tasks, the option to check an existing task would not exist.

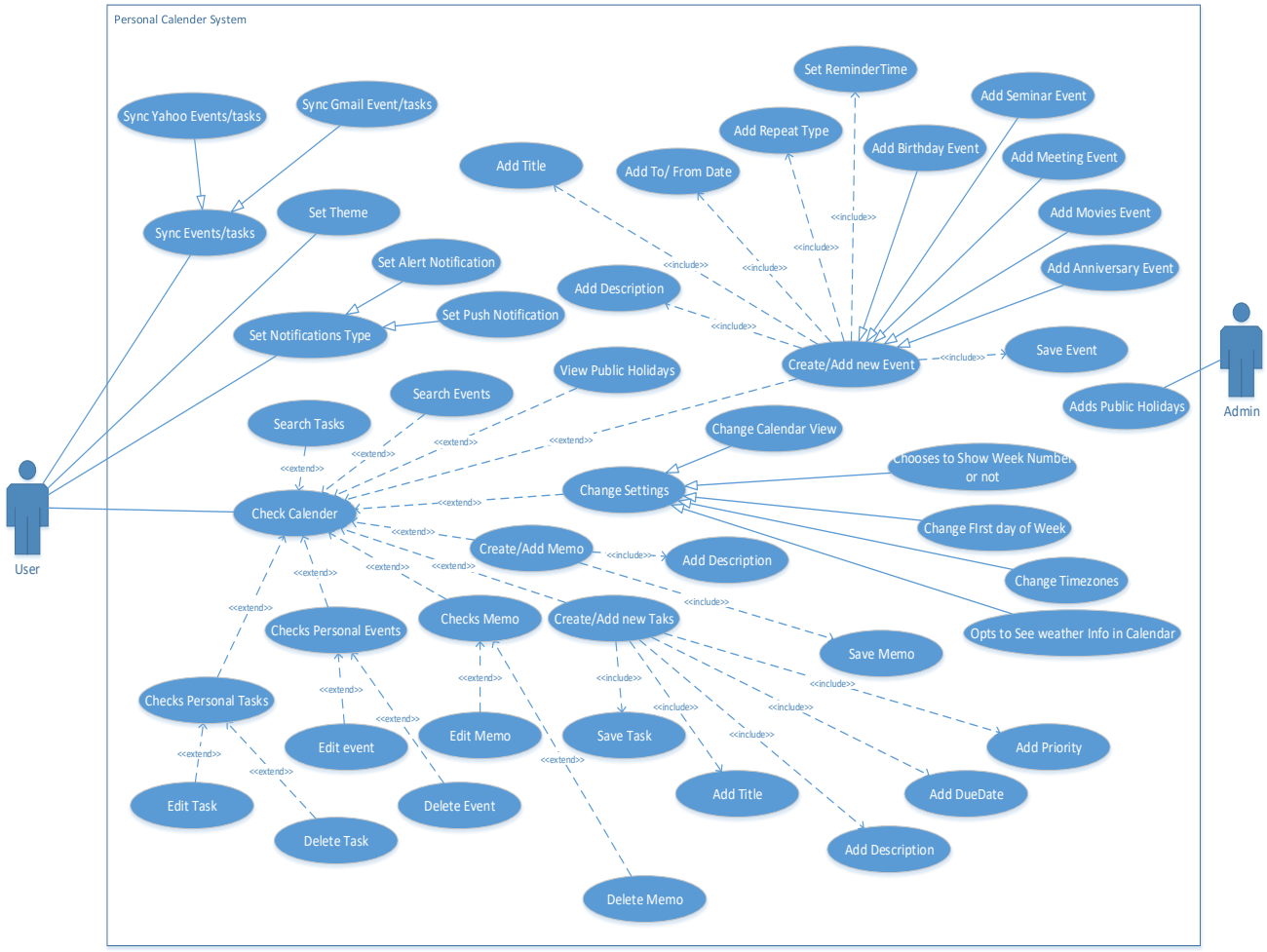
6) Set Notification

Use Case Name	Set Notification
Level	User Goal
Brief Description	When the user would like to set more personal, specific notification settings for his or her app regarding alert and push notifications.
Primary Actor	User
Stakeholders	User: the user has a vested interest in being able to set his or her notification settings in specific configurations that enhance his or her experience with using the application.
Triggering Event	When the user presses the set notification type button.
Preconditions	-User must exist
Post Conditions	Notification settings options should be displayed on the screen.
Flow of Events-	-User profile exists in the application system. -User presses the set notification type button
Exception Condition	If the user does not exist in the application system, there would not be a settings configuration that could be saved and applied to any existing user.

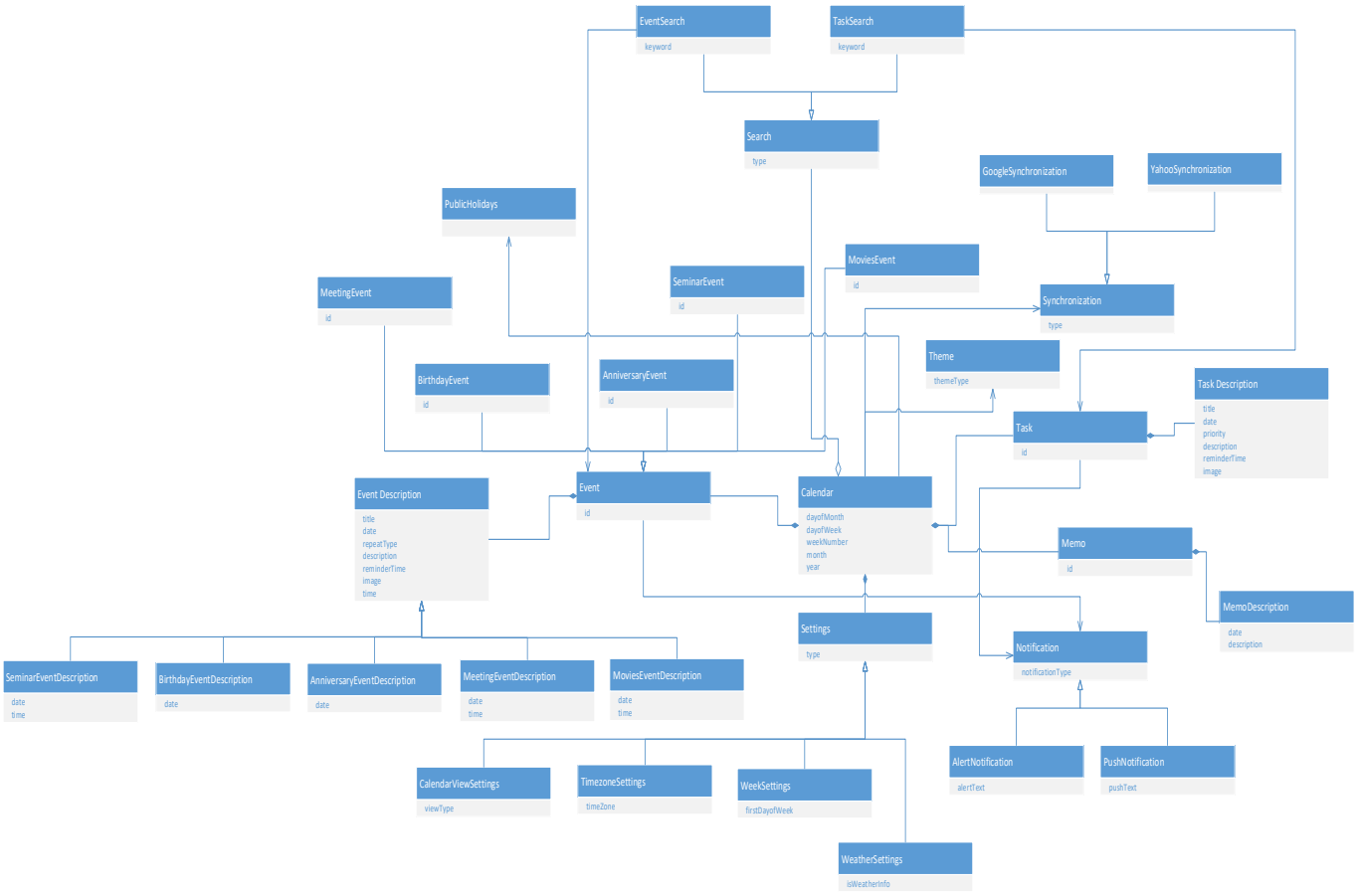
7) Change Calendar View

Use Case Name	Change Calendar View
Level	User goal
Brief description	User can change the views of calendar to different options as he wants. An options provided to user are show calendar by year, show calendar by month and show calendar by day.
Primary Actor	User
Stakeholders	User- User wants to change the view of calendar and switch to one of the views as calendar by year, calendar by month or calendar by day.
Triggering event	When user chooses the view type of calendar
Preconditions	User must exist User should have Calendar application installed. User must switch to particular view using option buttons provided to him.
Post conditions	View of calendar will change to one selected by user. Calendar will be displayed in selected view.
Flow of event	-User opens calendar -User clicks on any one button as he want the calendar view among three buttons provided as show by Year, Month and Day. -Calendar switches its view to selected view.
Exception Condition	-User does not select any other view that displayed view.

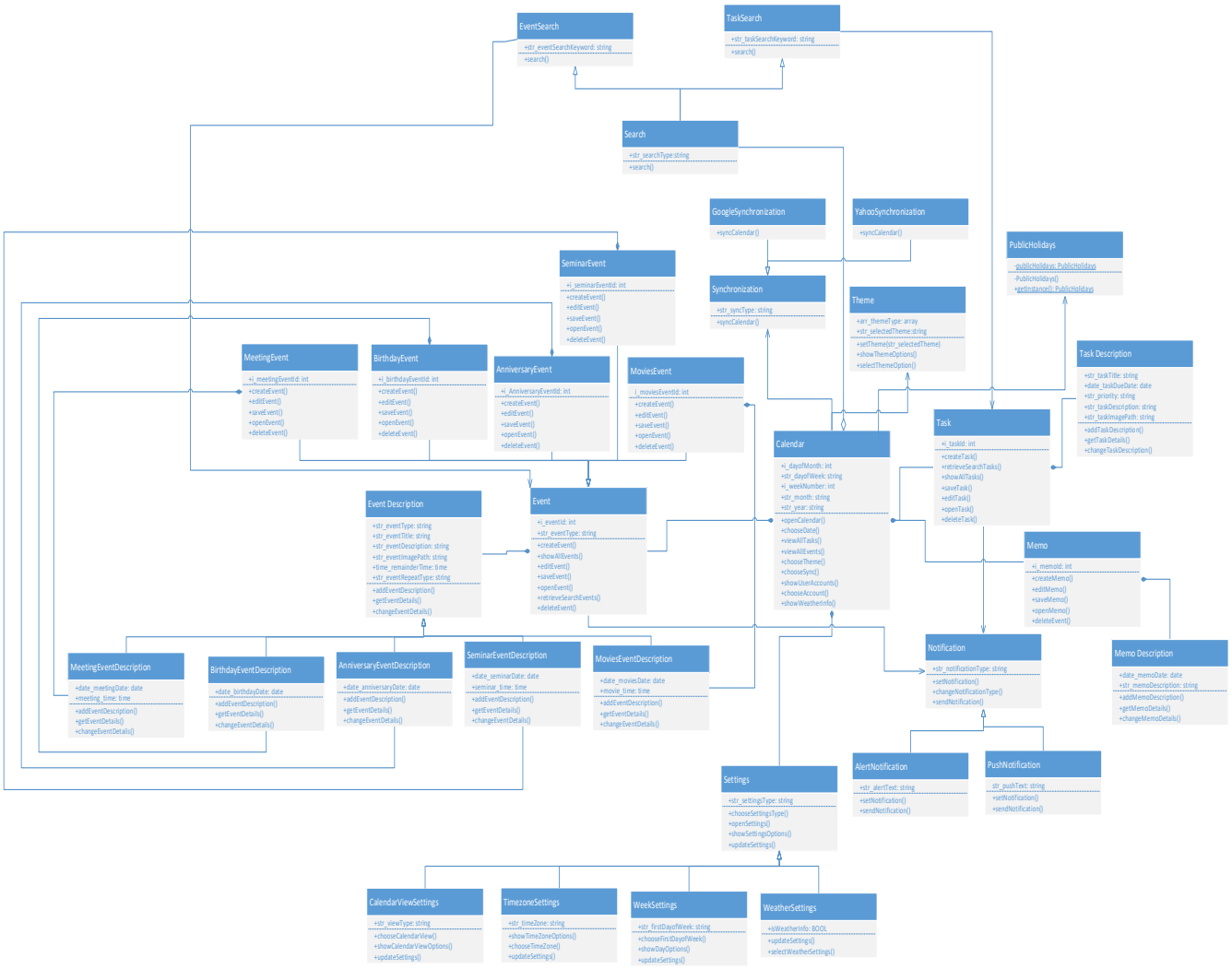
Use Case Diagram:



Domain Model Class Diagram:

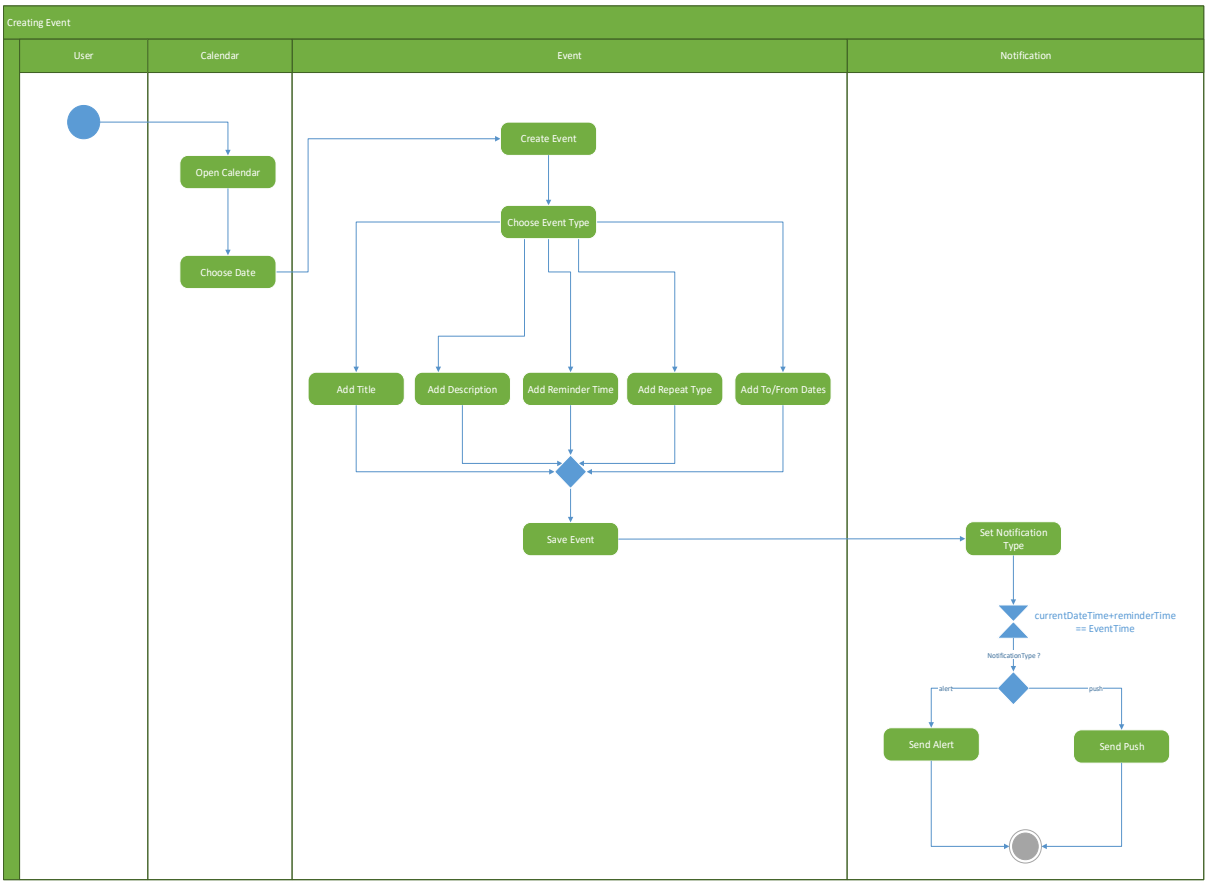


Design Model Class Diagram:

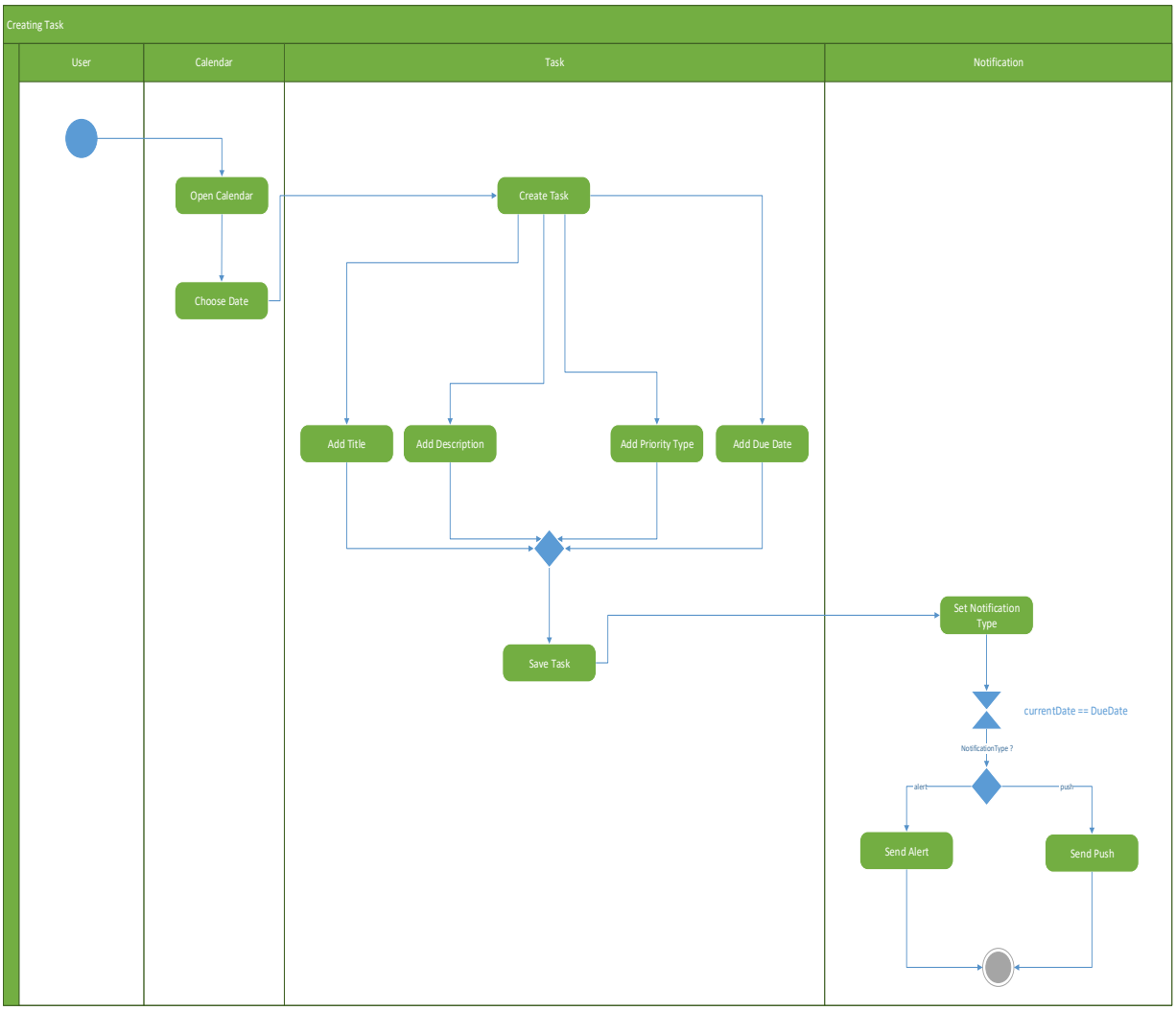


Activity Diagrams:

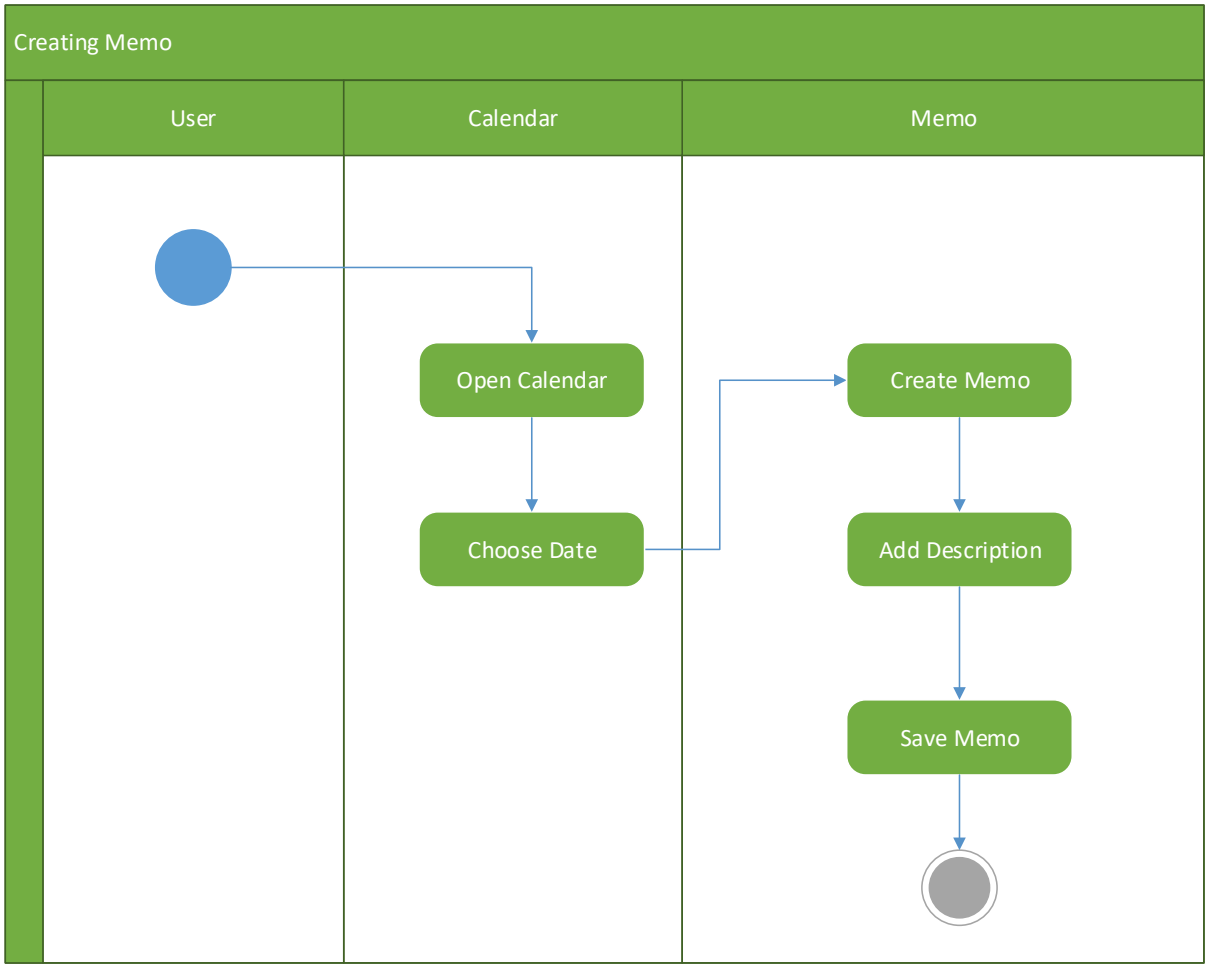
1) Creating Event



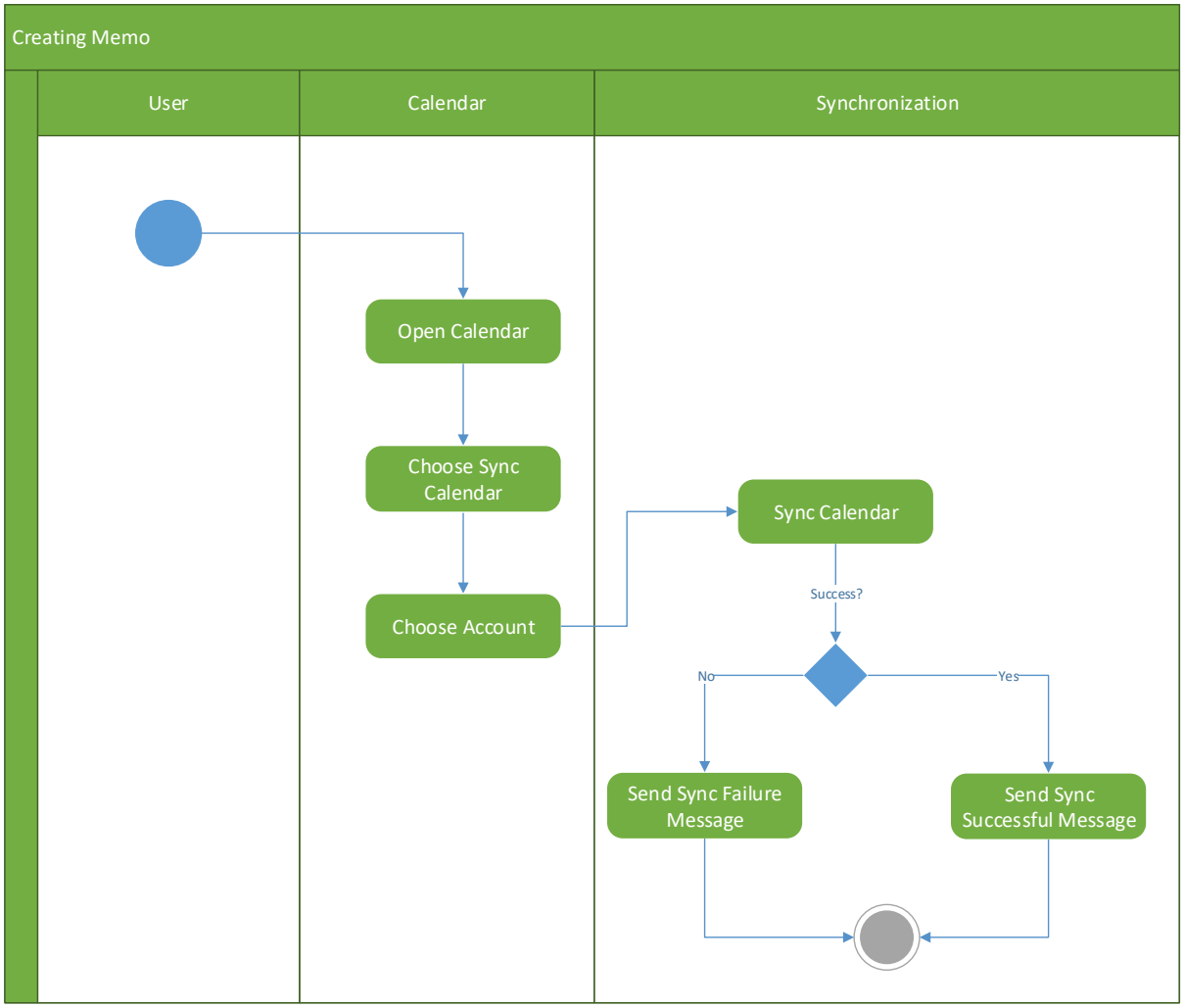
2) Creating Task



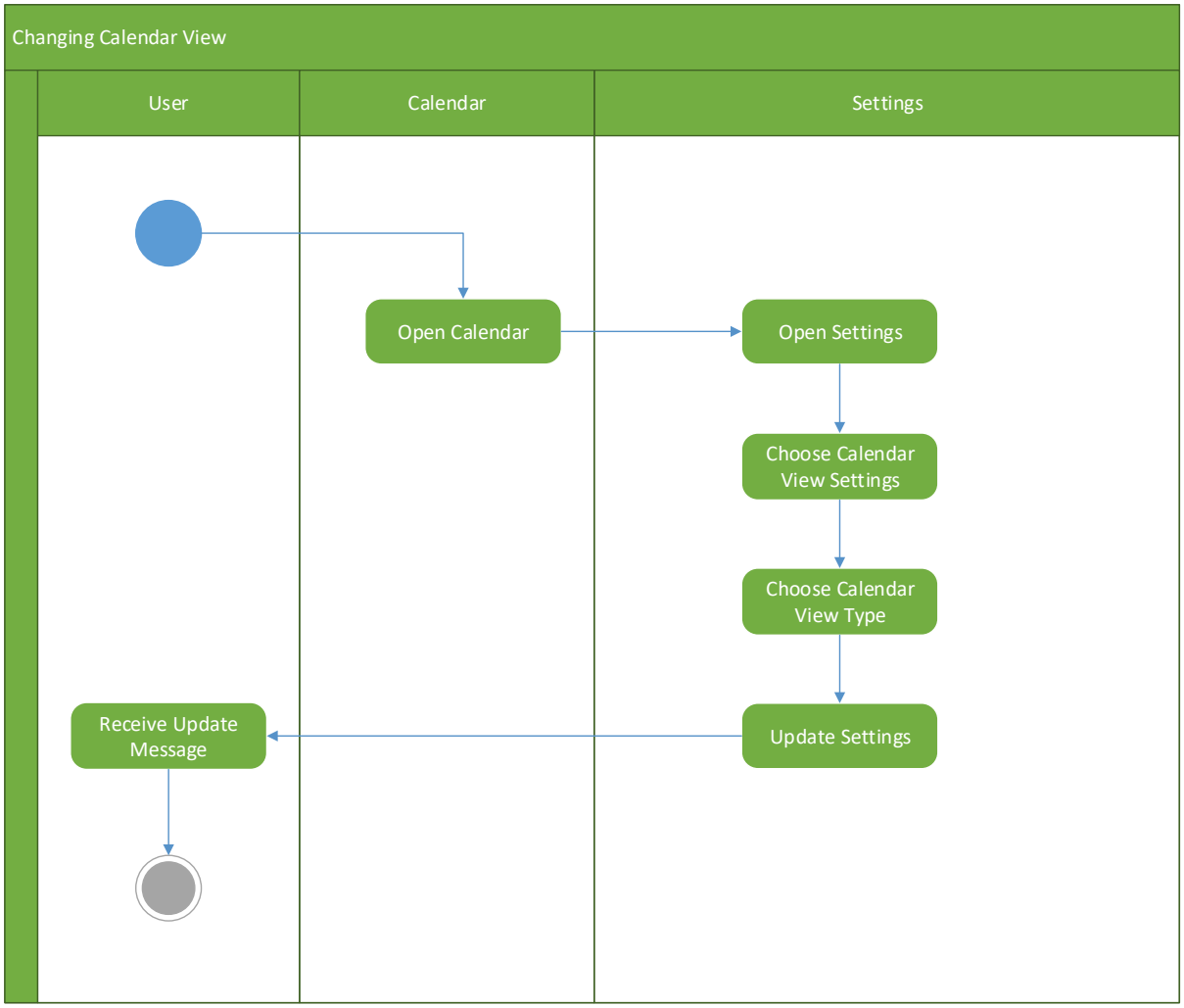
3) Creating Memo



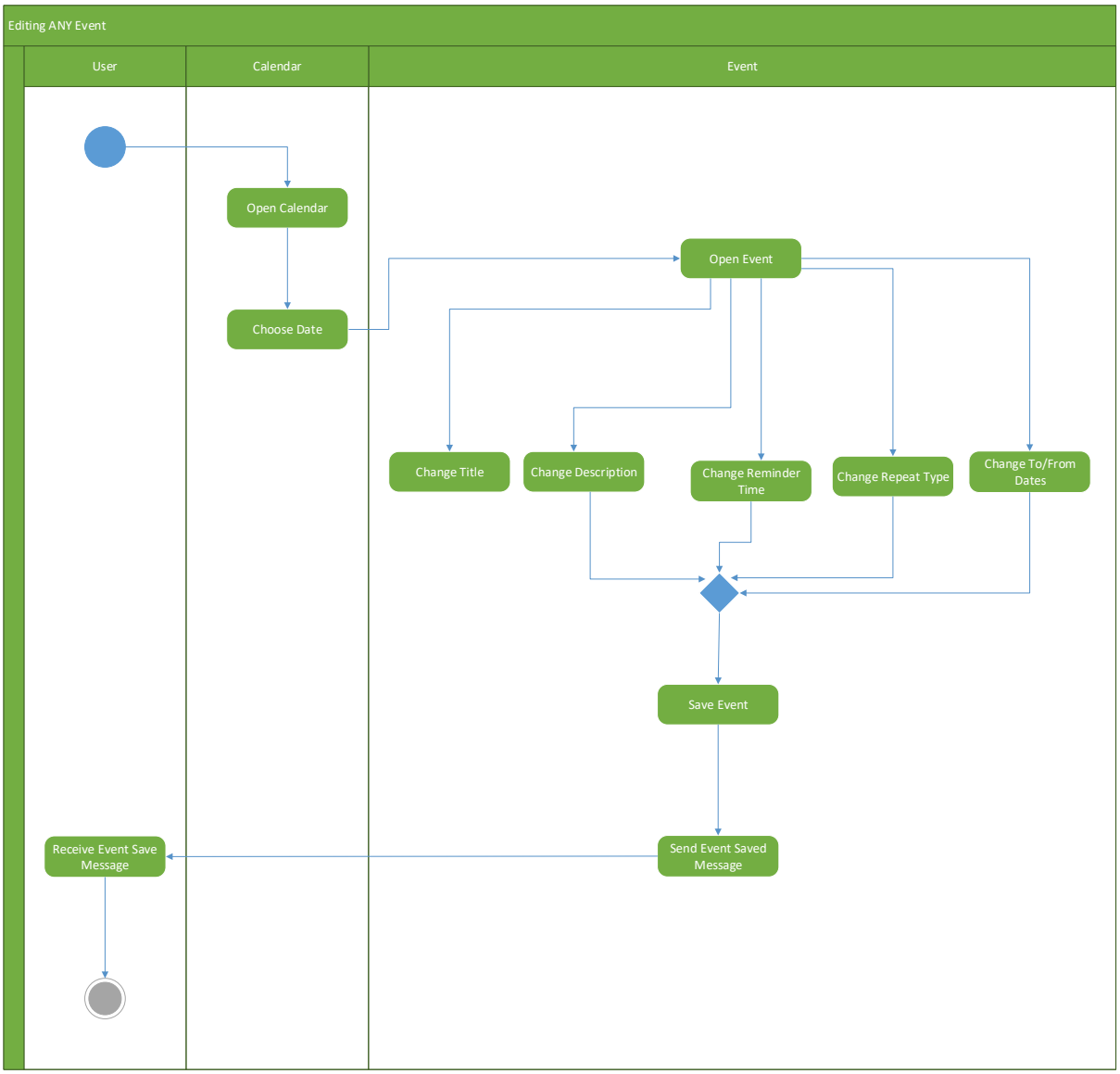
4) Syncing Calendar



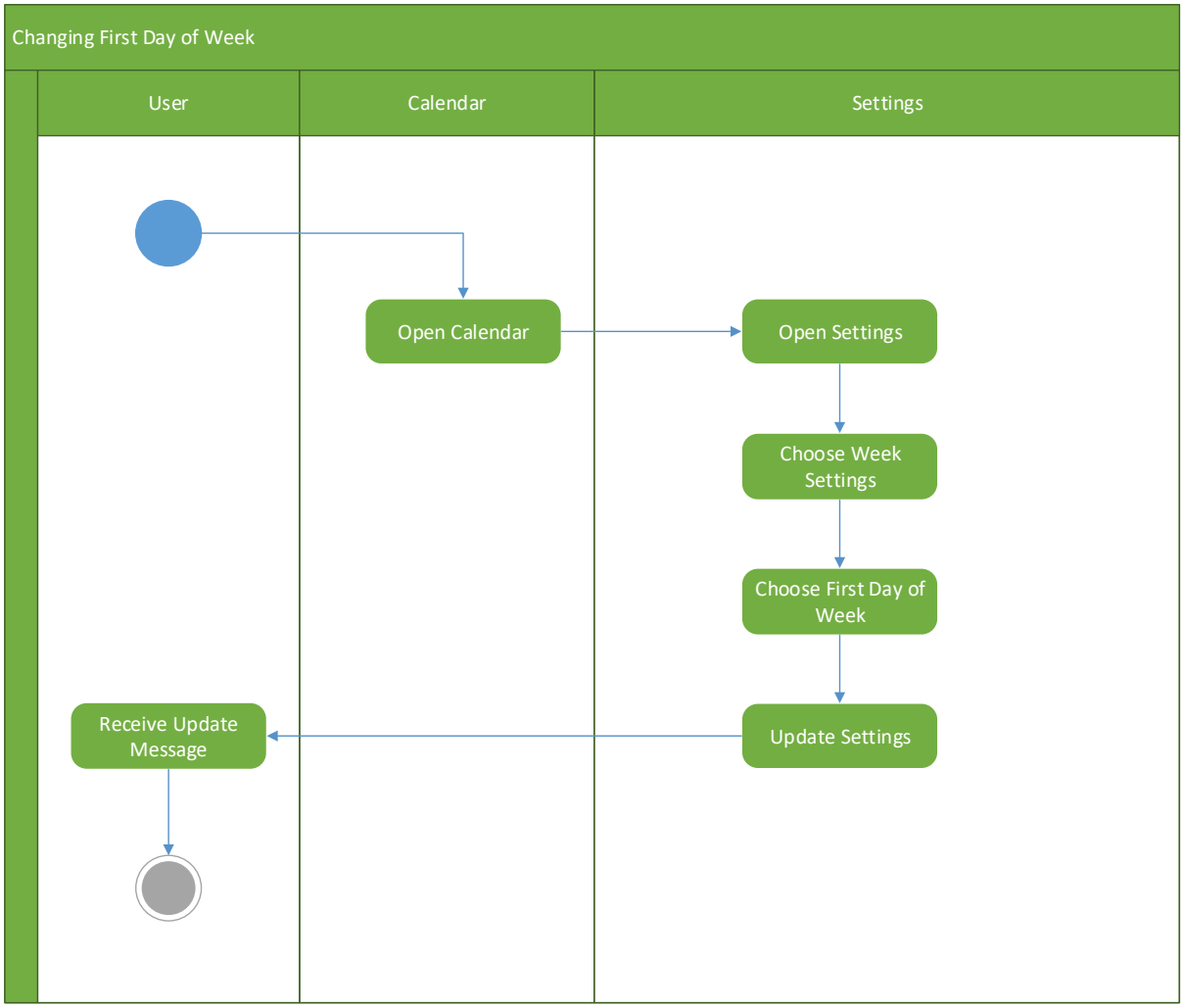
5) Changing Calendar View



6) Editing ANY Event

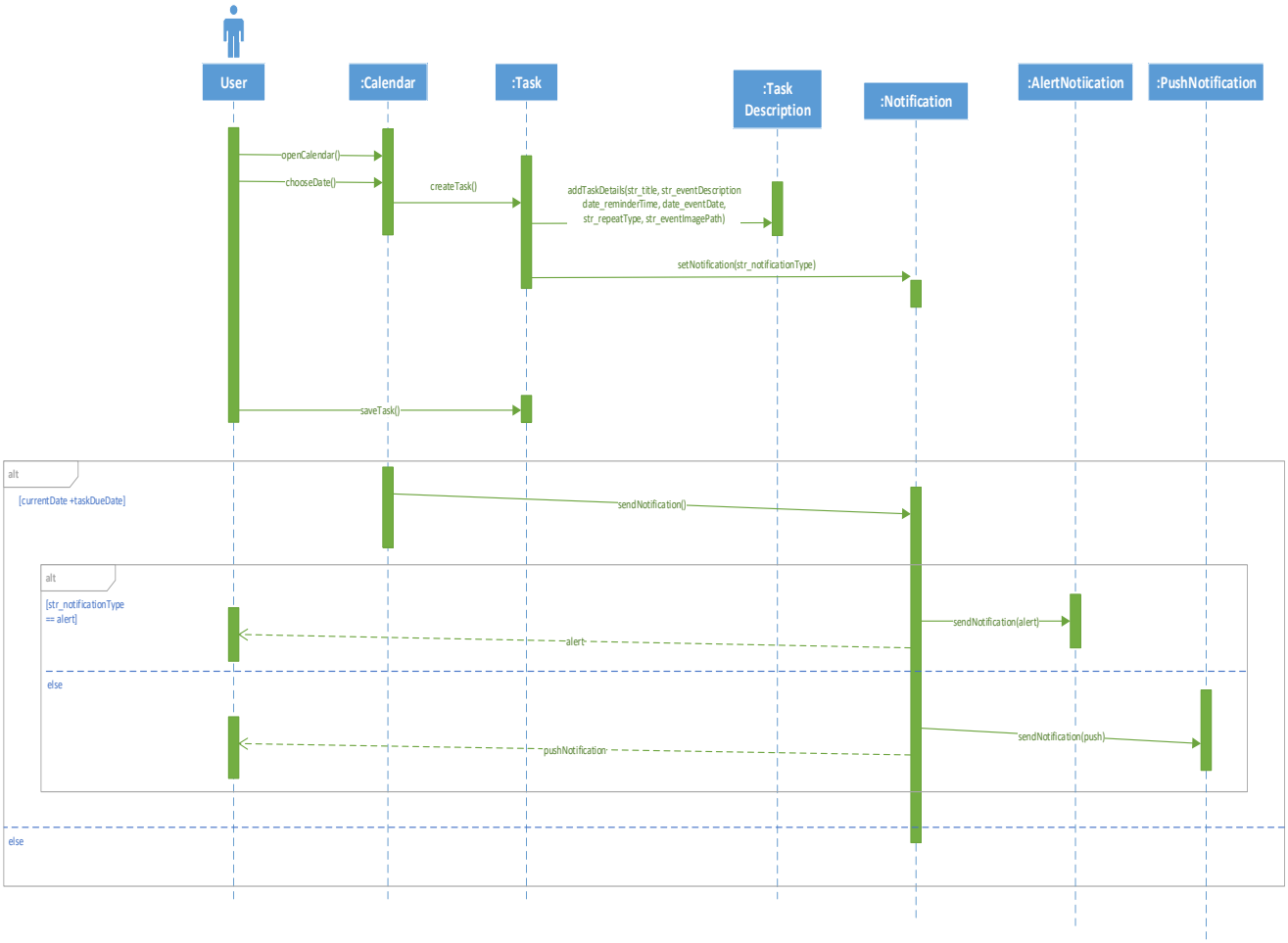


7) Changing First Day of Week

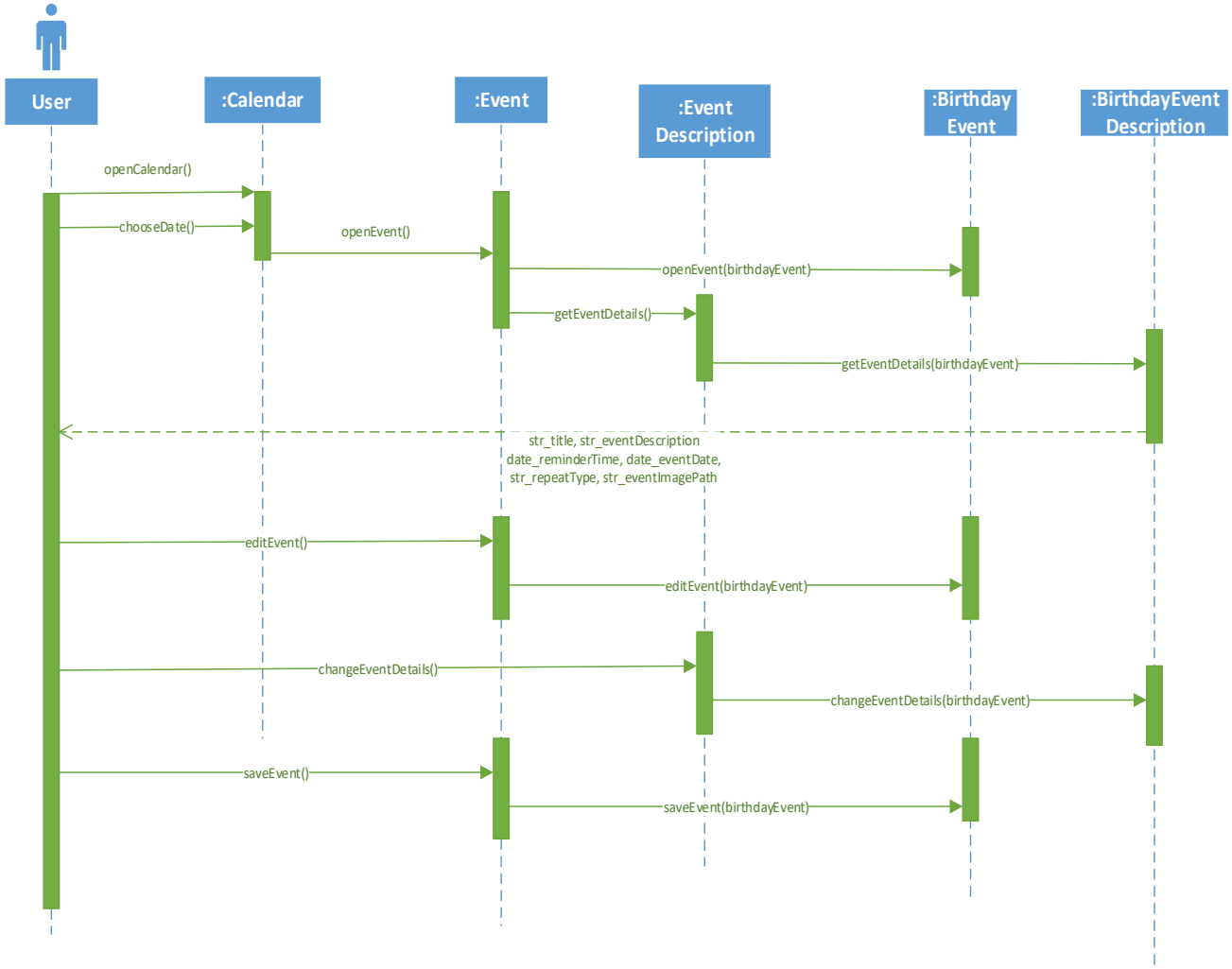


Sequence Diagrams:

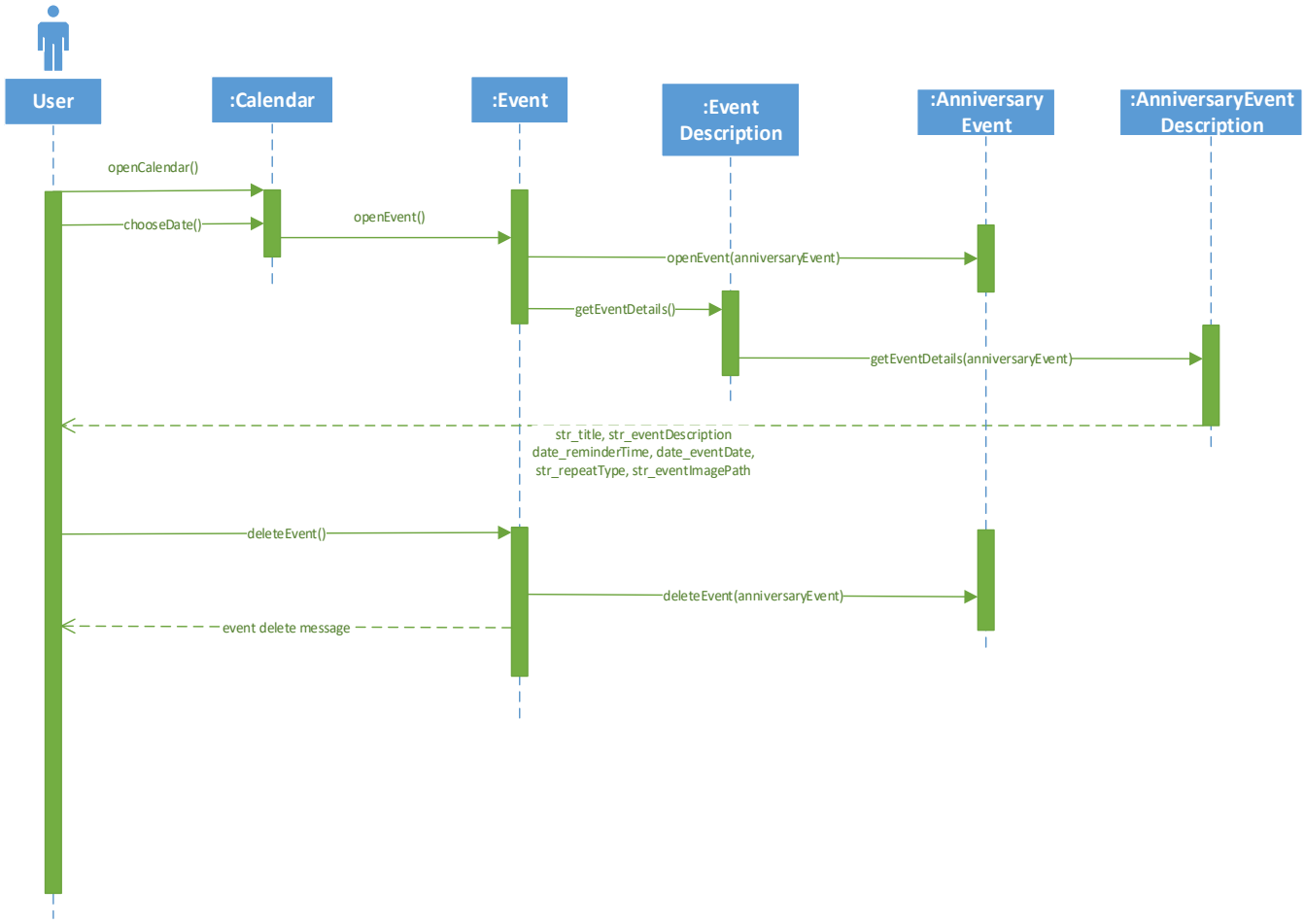
1) Create Task



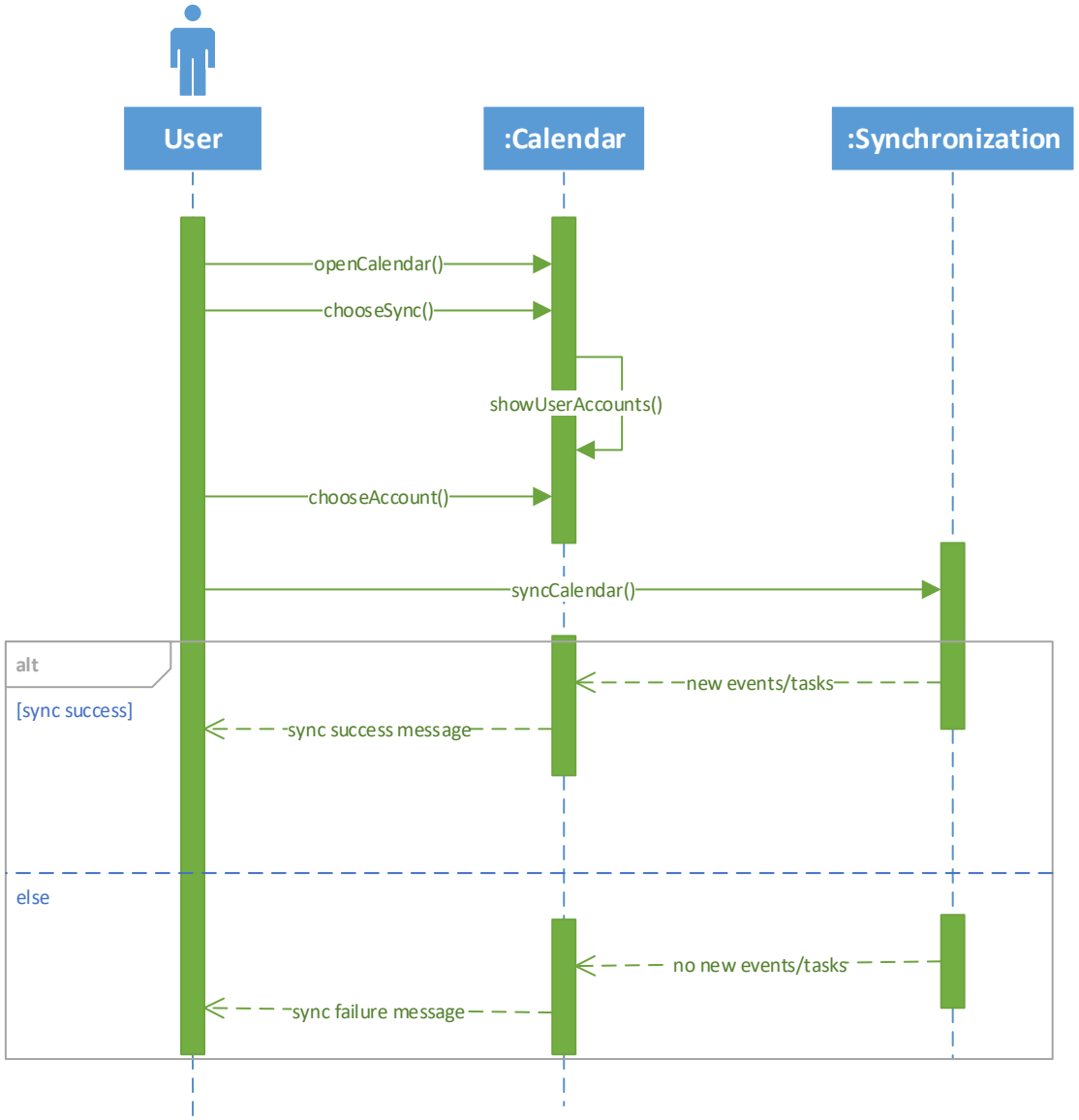
2) Edit Birthday Event



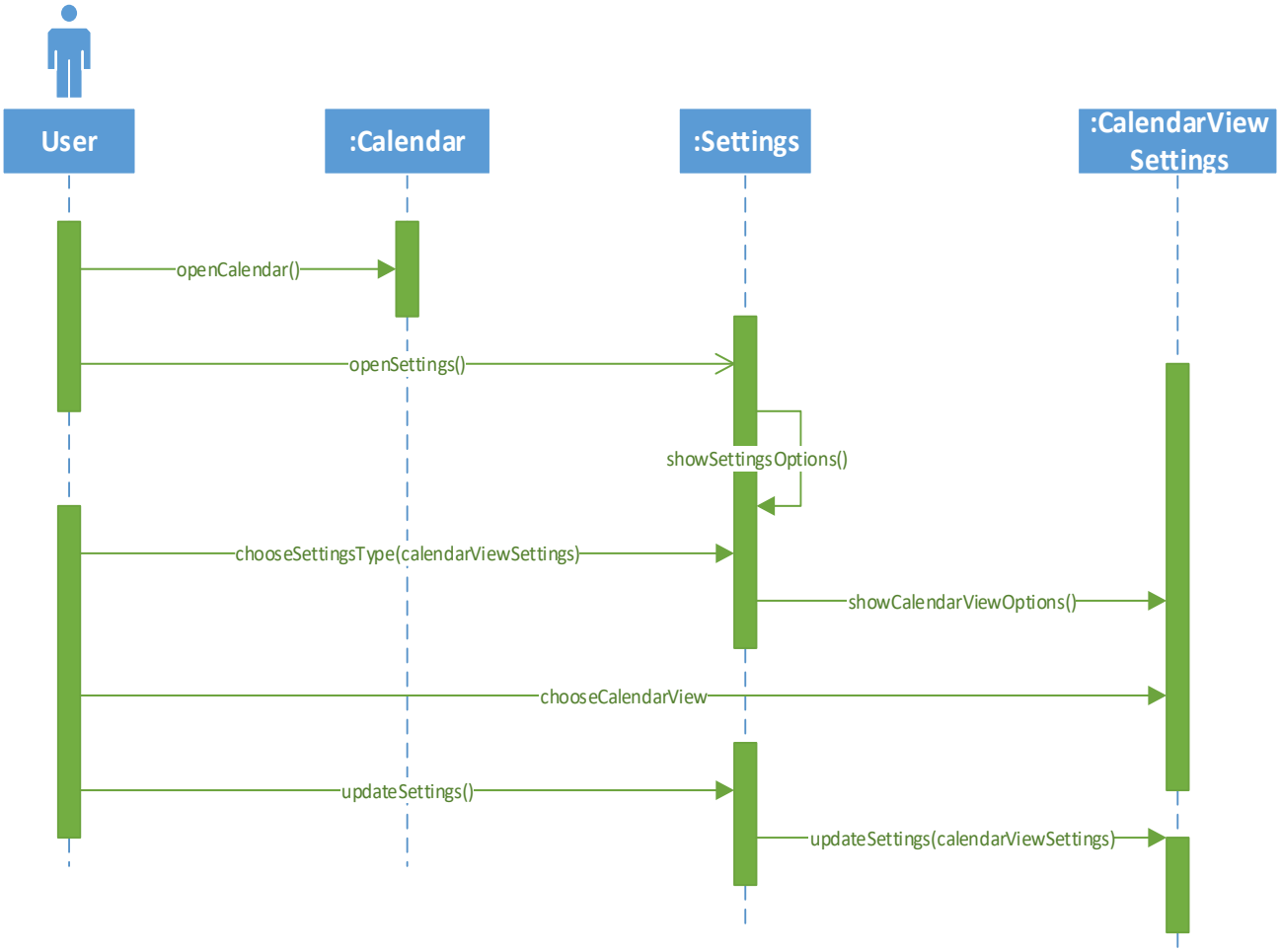
3) Delete Anniversary Event



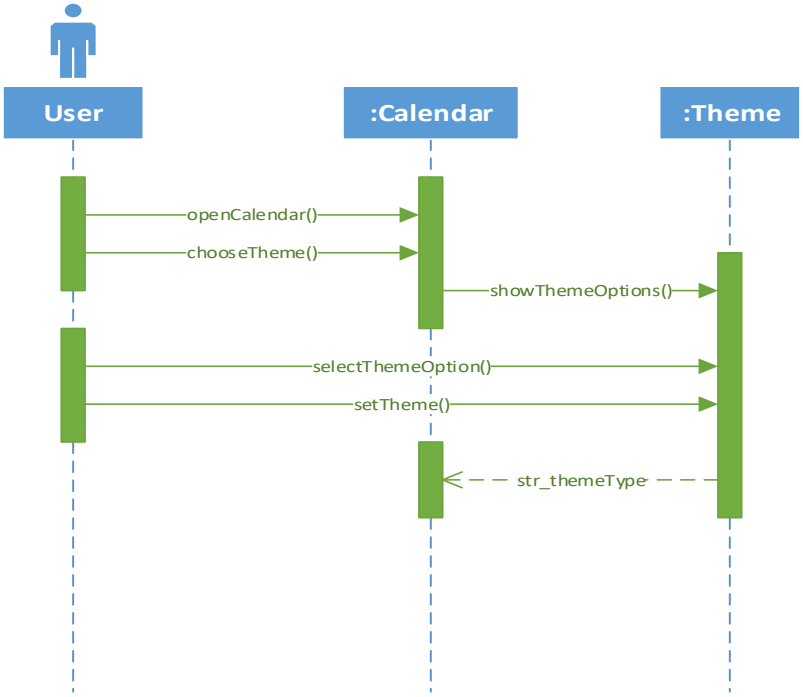
4) Sync Calendar



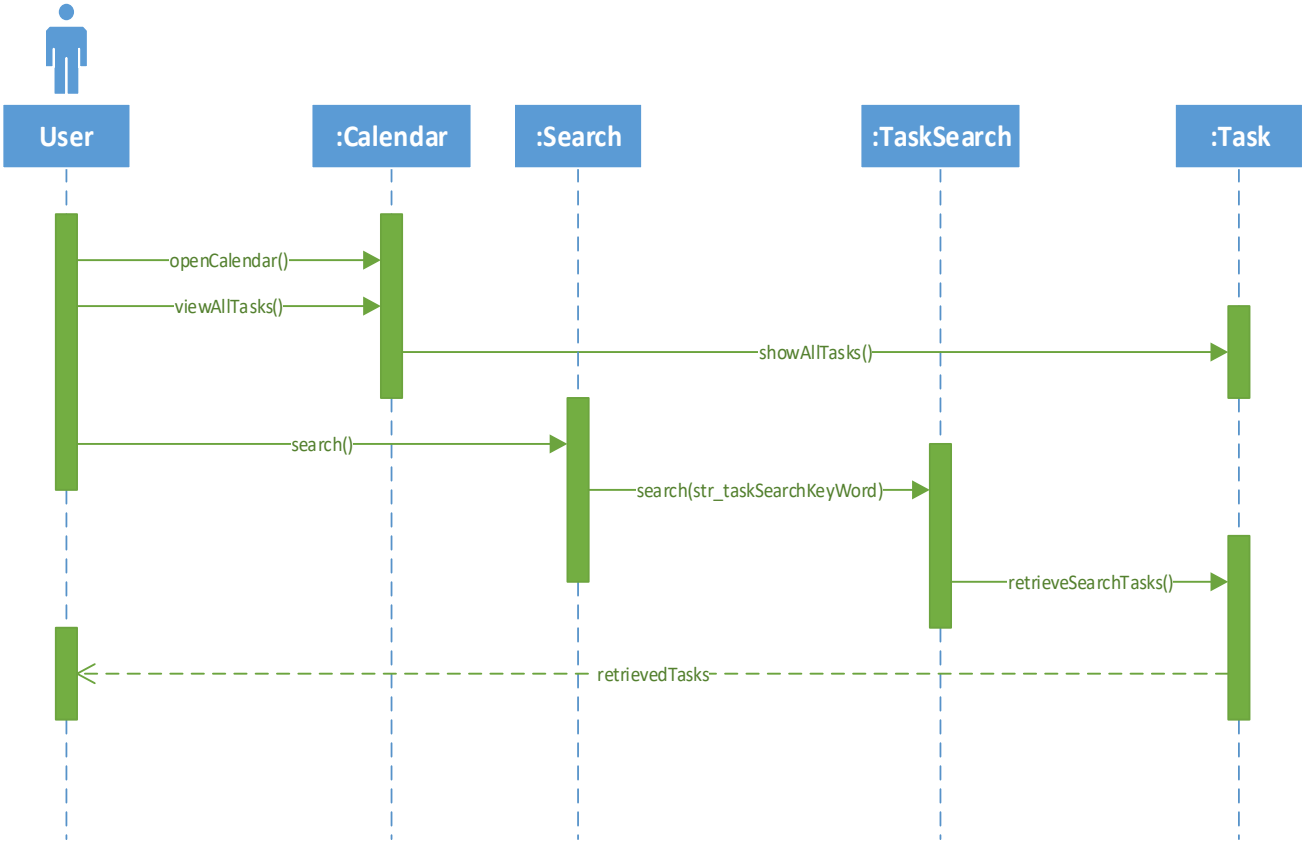
5) Update CalendarView Settings



6) Change Calendar Theme

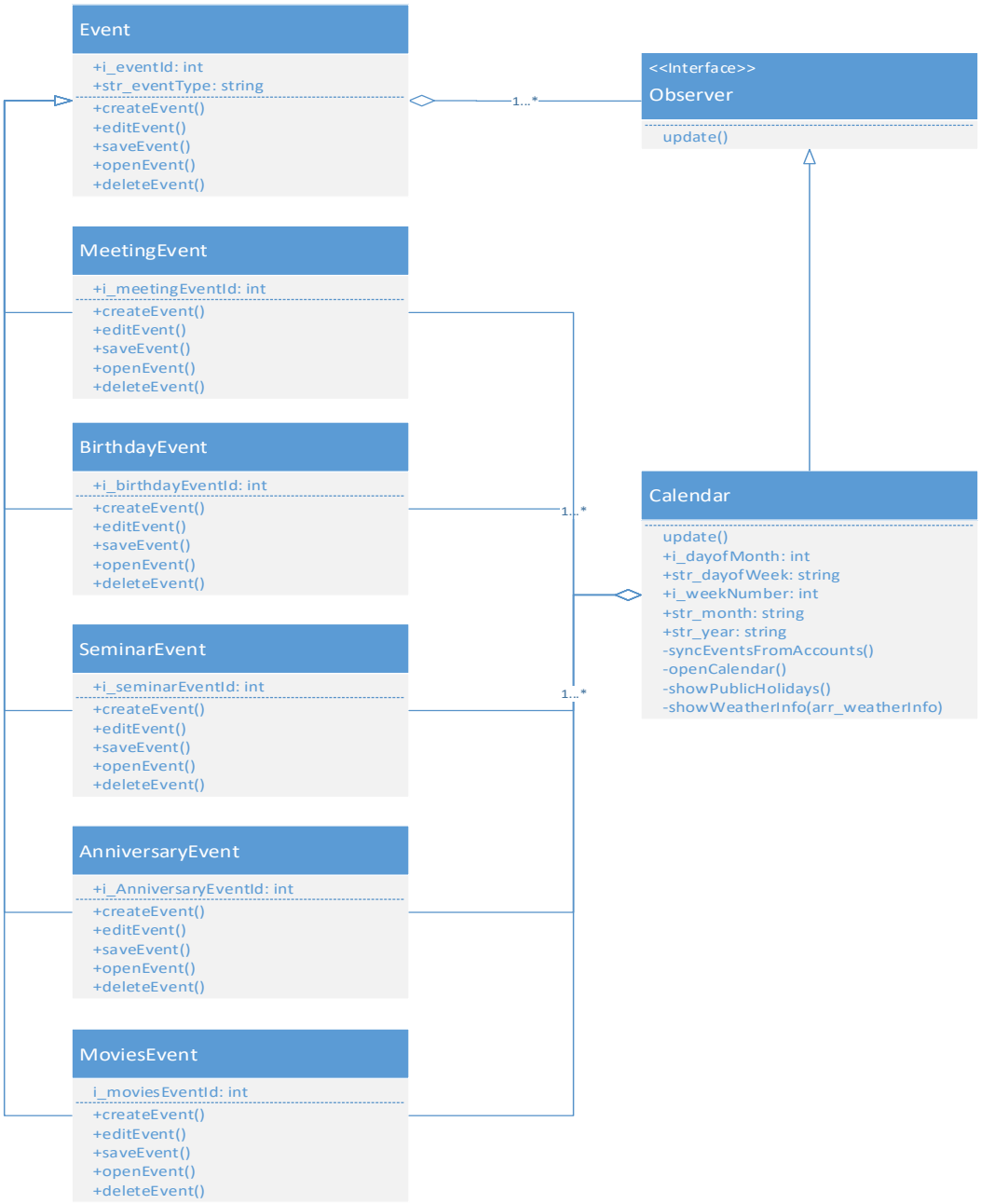


7) Search Tasks



Design Patterns:

1) Observer Design Pattern



Introduction:

The observer pattern is suitable for when multiple entities wish to “listen” for updates from the same source, and permits that source to update all registered entities when its state is changed.

Implementation:

Calendar objects inherit from the observer interface and aggregate the concrete event objects, which in this case are MeetingEvent, BirthdayEvent, SeminarEvent, AnniversaryEvent, and MoviesEvent. These concrete events themselves then inherit from the Event interface. The relationship between the Event interface and observer interface allows all concrete registered observers to be notified at the same time when notifyObservers is called by a ConcreteEvent, an observer in this case being the Calendar objects.

2) Singleton Design Pattern

Introduction:

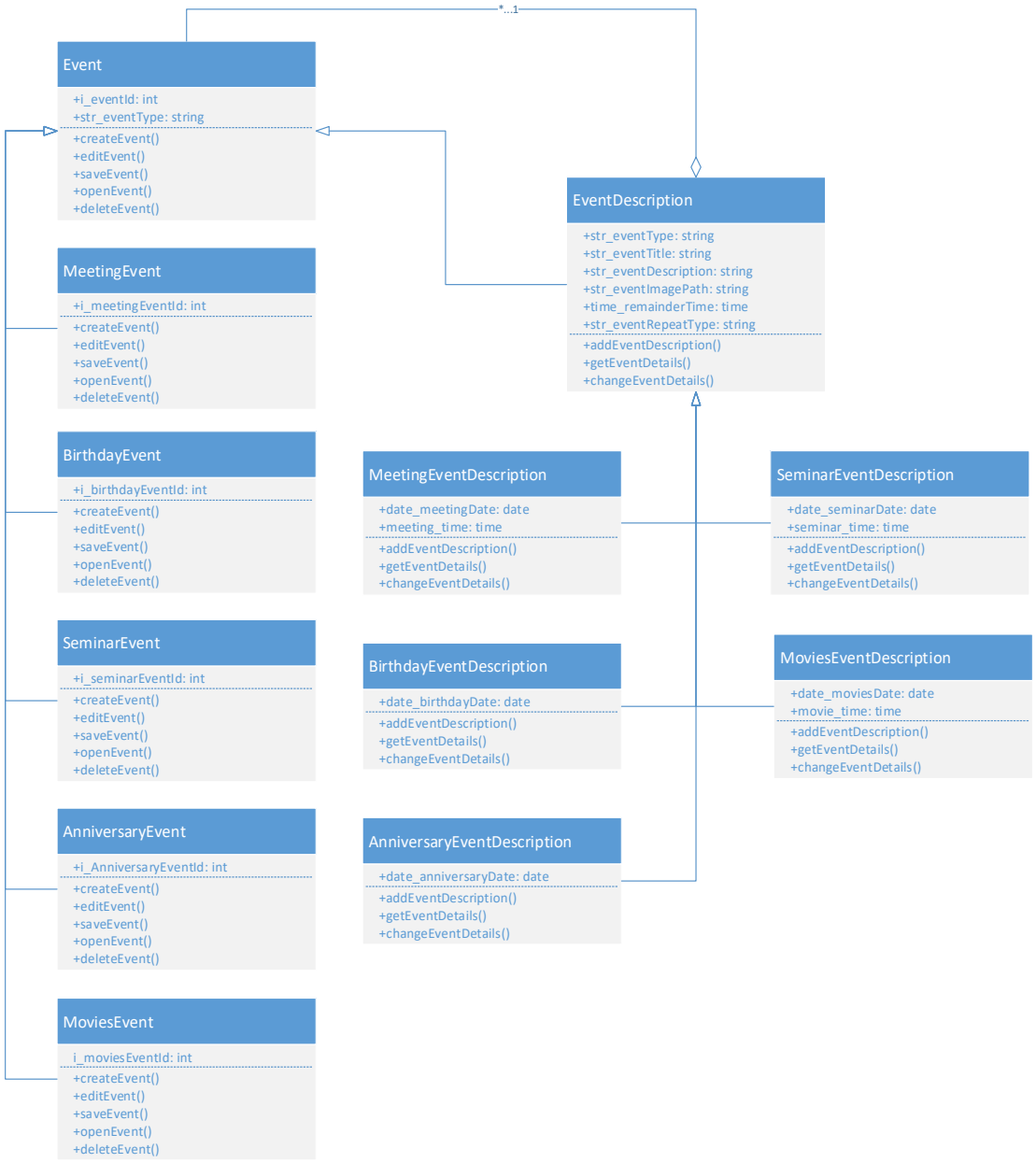
The singleton design pattern ensures a class has only one instance and provide global point of access to it.

Implementation:

Here publicHolidays is a private static variable to hold one instance of the class PublicHolidays. The constructor PublicHolidays() is also private, so only PublicHolidays can instantiate this class.

The getInstance() method gives us a way to instantiate a class and also to return an instance of it. It is a static method which means it is a class method, so you can conveniently access this method from anywhere in your code.

3) Decorator Design Pattern



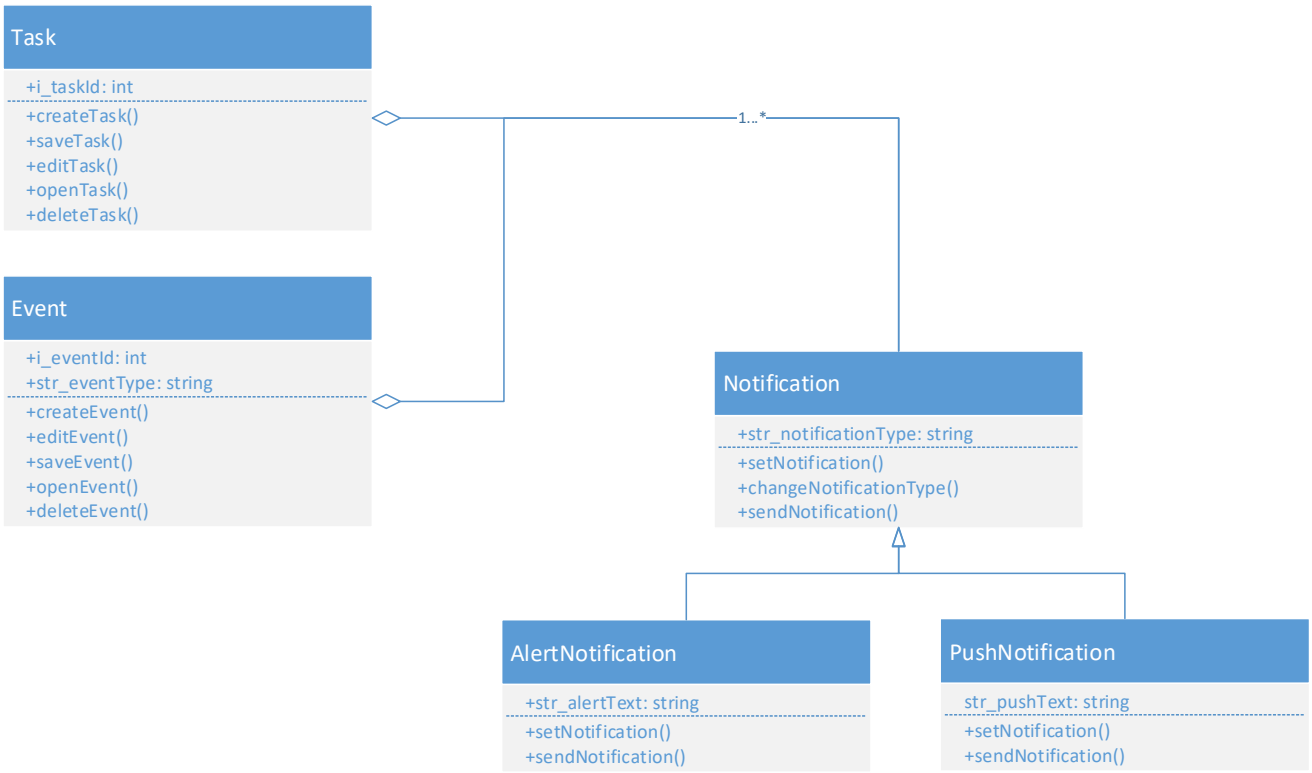
Introduction:

The decorator pattern is useful when there is a need to dynamically attach responsibilities to an object. This need for a dynamic approach arises since it is often the case when the responsibilities of an object are unforeseen, and it is too cumbersome or impractical to attach those responsibilities preemptively. The decorator pattern adds flexibility and adaptability to the creation process of the involved objects.

Implementation:

In our implementation, Event objects are built using the decorator pattern. The decorator hierarchy, in this case the EventDescription classes, simplifies and enhances the flexibility of our Event objects by wrapping the concrete EventDescription classes around Event objects as needed. Description wrappers will override the appropriate functions related to that specific type of event.

4) Strategy Design Pattern



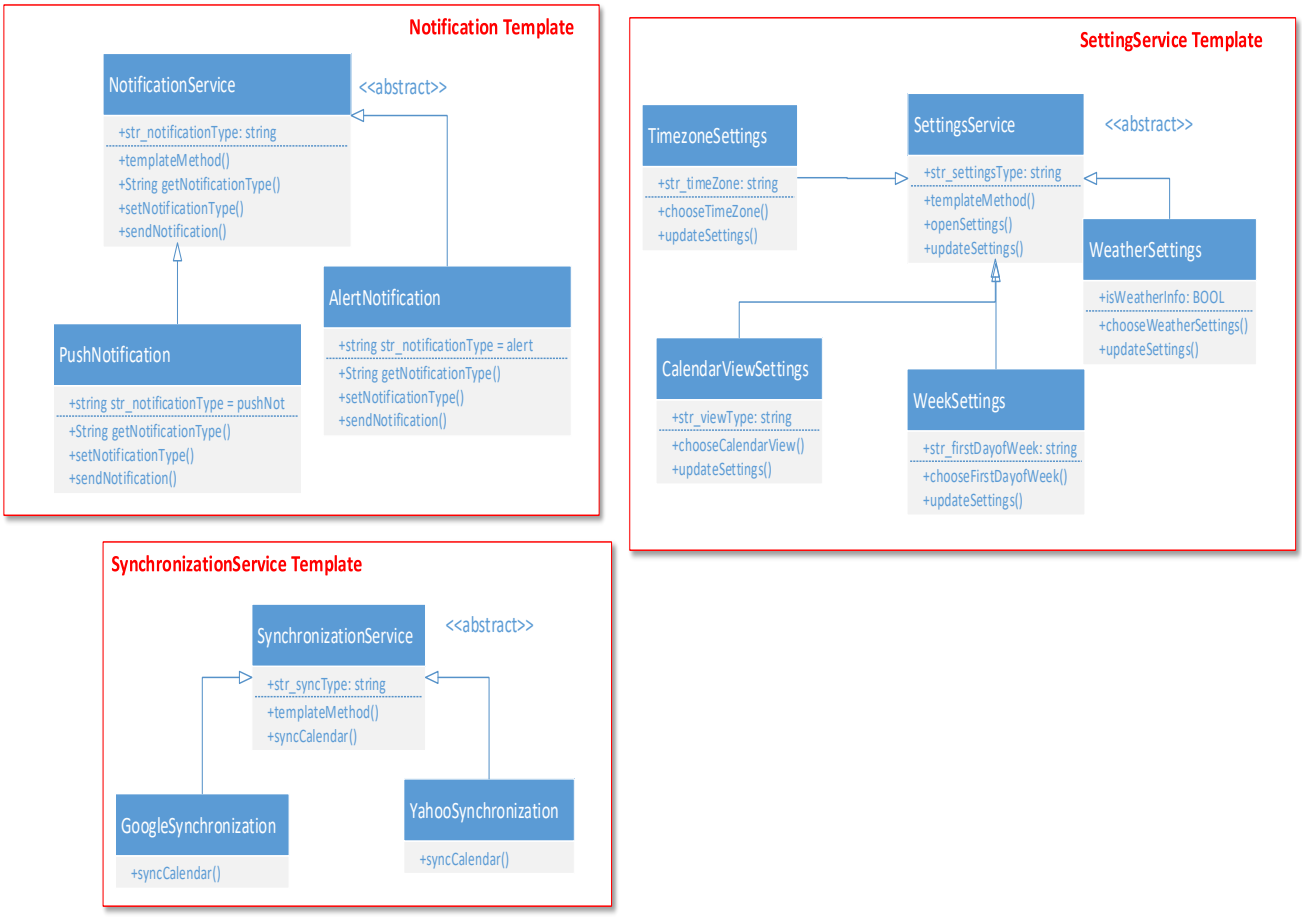
Introduction:

The strategy pattern is used when various different methodologies, or strategies, are required for the proper handling of a system. Typically there are several different approaches or algorithms available for implementation when handling specific problems or operations, with each approach being slightly nuanced and more effective for certain situations. The strategy pattern is excellent at allowing a system the flexibility and capacity to leverage more than one approach.

Implementation:

In our implementation, concrete "notification" classes, in this case the AlertNotification and PushNotification classes inherit from the Notification class, which together serve to constitute the strategy hierarchy. Strategies are aggregated by the task and event classes, which serve as our context classes. When an alert notification or push notification is required, the context established by the specific task or event class originating the request will determine which notification strategy, either an alert or push, will be implemented.

5) Template Method Design Pattern



Introduction:

In Template pattern, an abstract class exposes defined templates to execute its methods. Its subclasses can override the method implementation as per need but the invocation is to be in the same way as defined by an abstract class. This pattern comes under behavior pattern category.

Implementation:

In our personal calendar system we implemented template methods for NotificationService, SettingService and SynchronizationService.

There are abstract class for every template and it contains template method and methods that are abstracted from template method.

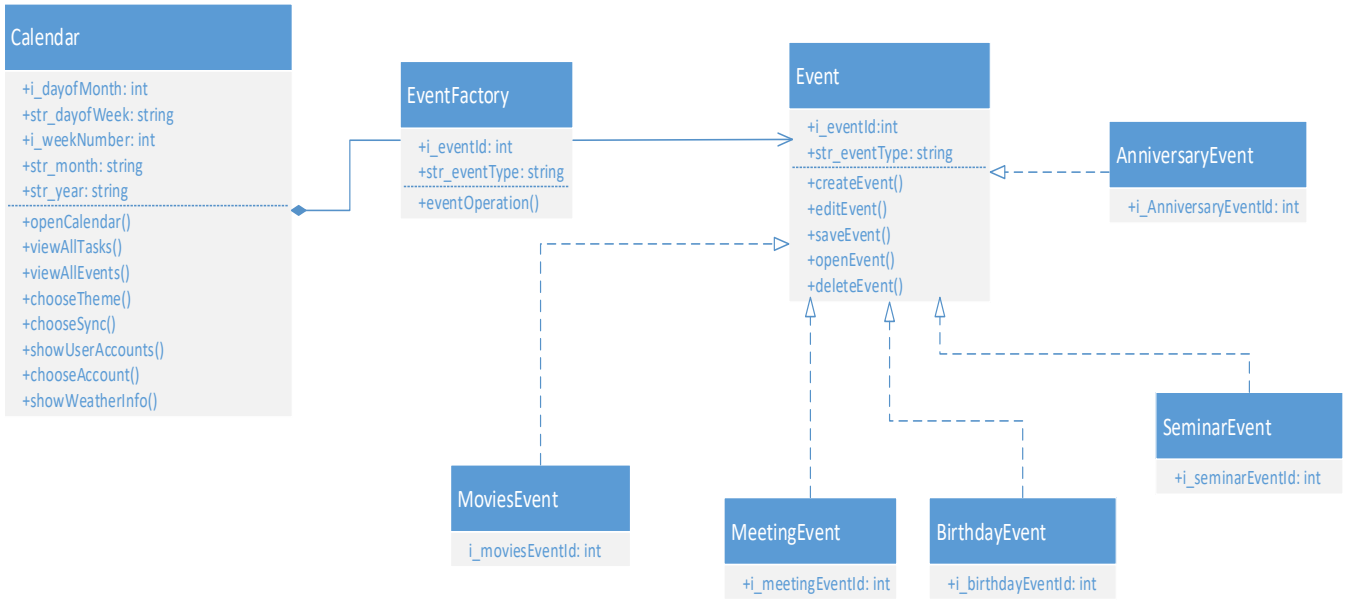
There are many concrete classes for each template. These classe's implements full set of methods from abstract class called by template method.

- Classes abstracting NotificationService class are:
  - PushNotification: Activates Push Notification function for users Event.
  - AlertNotification: Activates Alert to notify user about event.
- Classes abstracting SettingService class are:
  - TimezoneSetting: User can change timezone as he moves from location to location.
  - WeekSetting: This class lets user to change week settings.

- WeatherSetting: This class contains methods that allows user to change the weather settings.
- CalendarViewSetting: This class allows user to change calendar view to show calendar by year, by week or by day.

- Classes abstracting SynchronizationService class are:
  - GoogleSynchronization: This class having method to sync the events with google accounts.
  - YahooSynchronization: This class is for functionality of syncing calendar events with yahoo accounts.

6) Factory Method Design Pattern



Introduction:

In Factory Method pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface. This falls under category of creational patterns.

Implementation:

In personal calendar system, we implemented it for Events functionality. There are five types of events user can create namely MoviesEvent, MeetingEvent, BirthdayEvent, SeminarEvent and AnnivarsaryEvent.

Here EventFactory is the only part of our application that is connected with class Events. Class Events has all methods for creating new event of any type. It is created as an abstract class and method declared inside this are implemented

Classes MoviesEvent, MeetingEvent, BirthdayEvent, SeminarEvent and AnnivarsaryEvent are concrete classes which implements methods from Event.