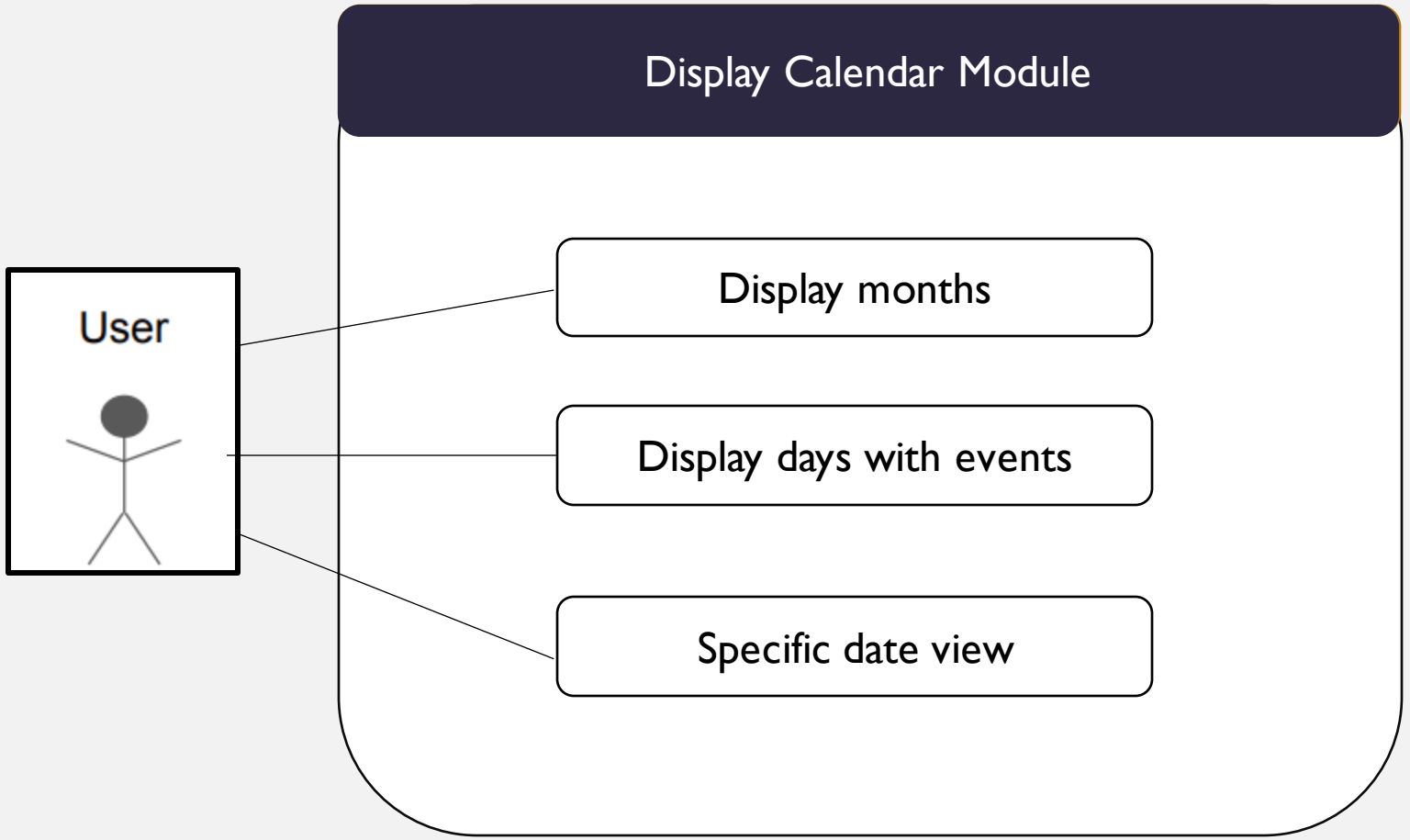
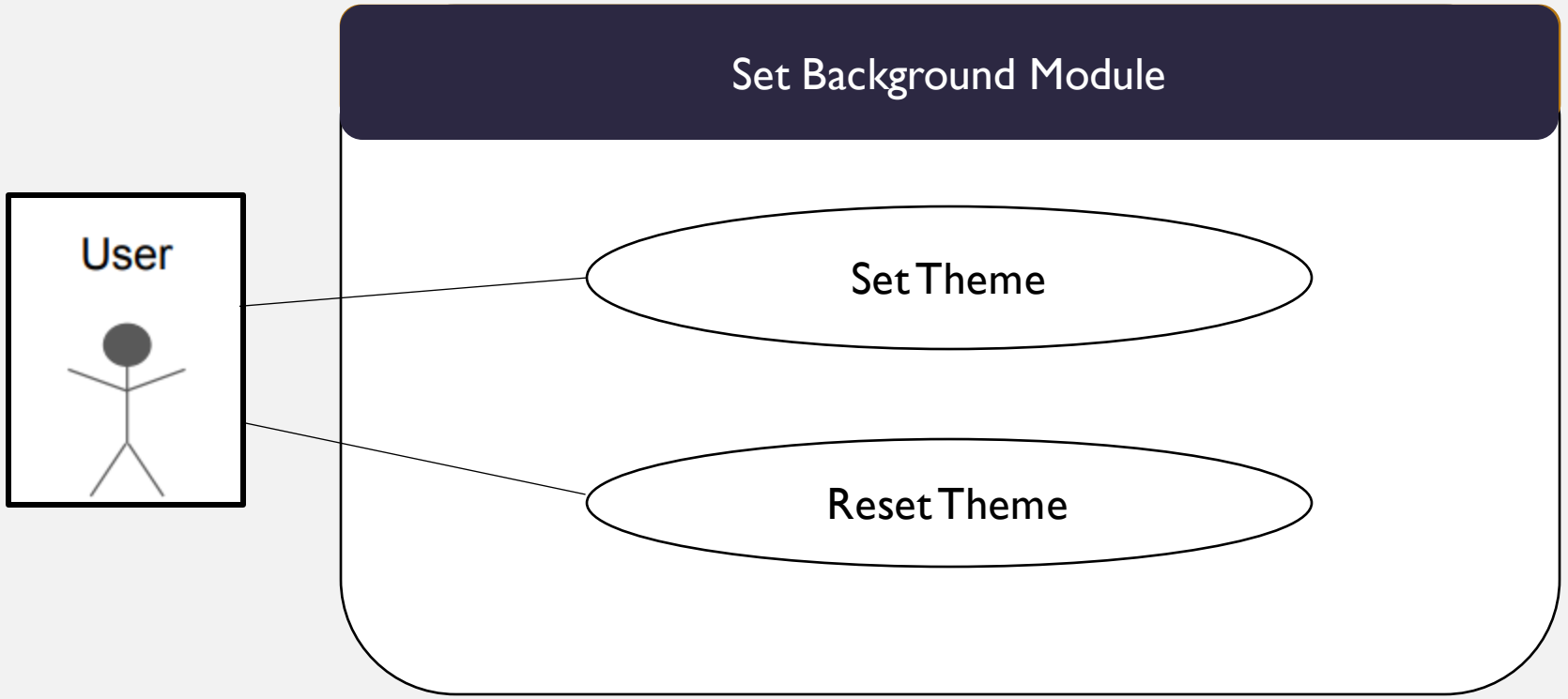
UML FOR CALENDAR APP

- 190101028 – Bolleboina Madhumitha
- 190101030 – Chappidi Shreya
- 190101042 – Kalapati Kasvitha
- 190101056 – Mushanolla Pranathi

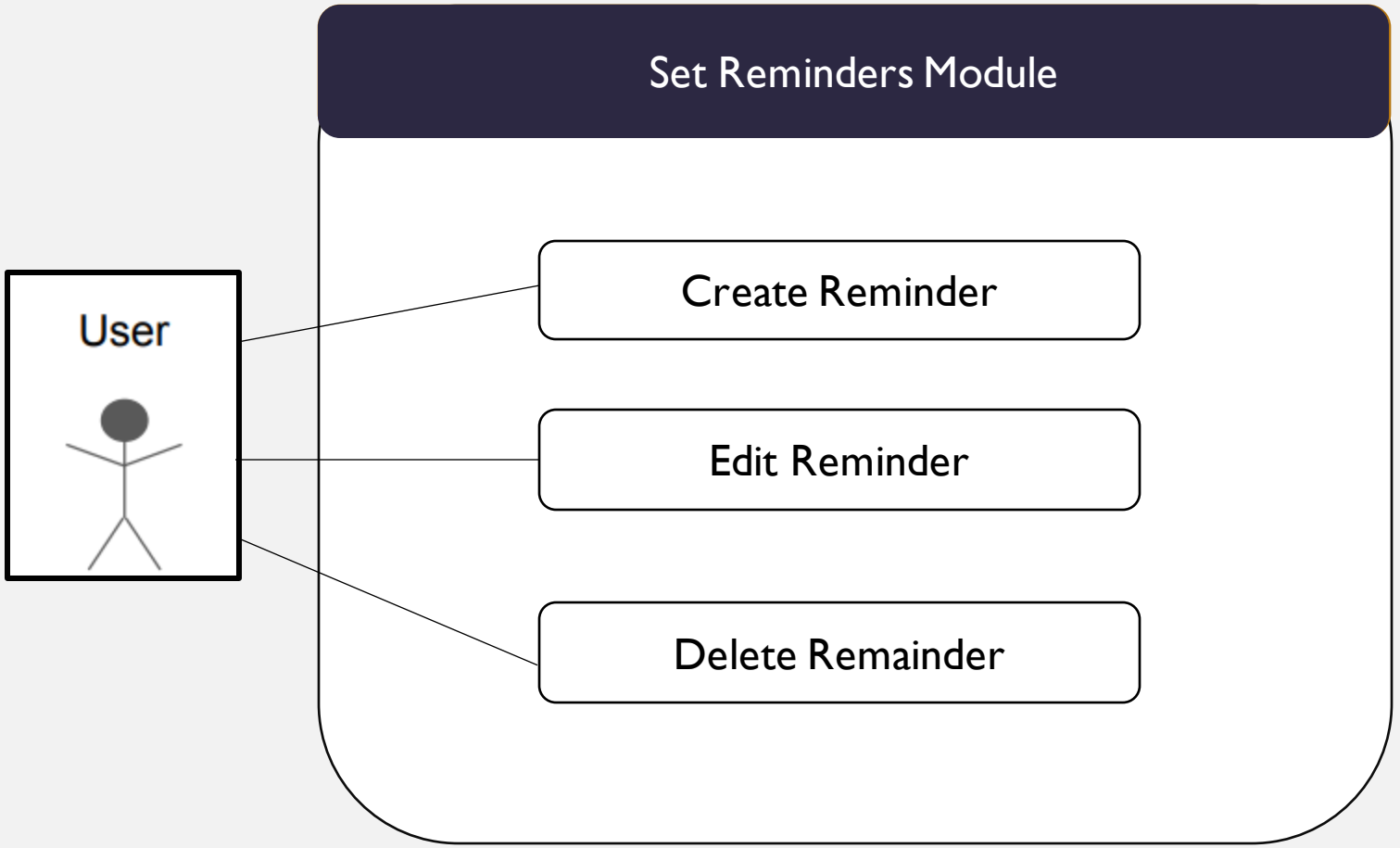
USER DIAGRAM 1



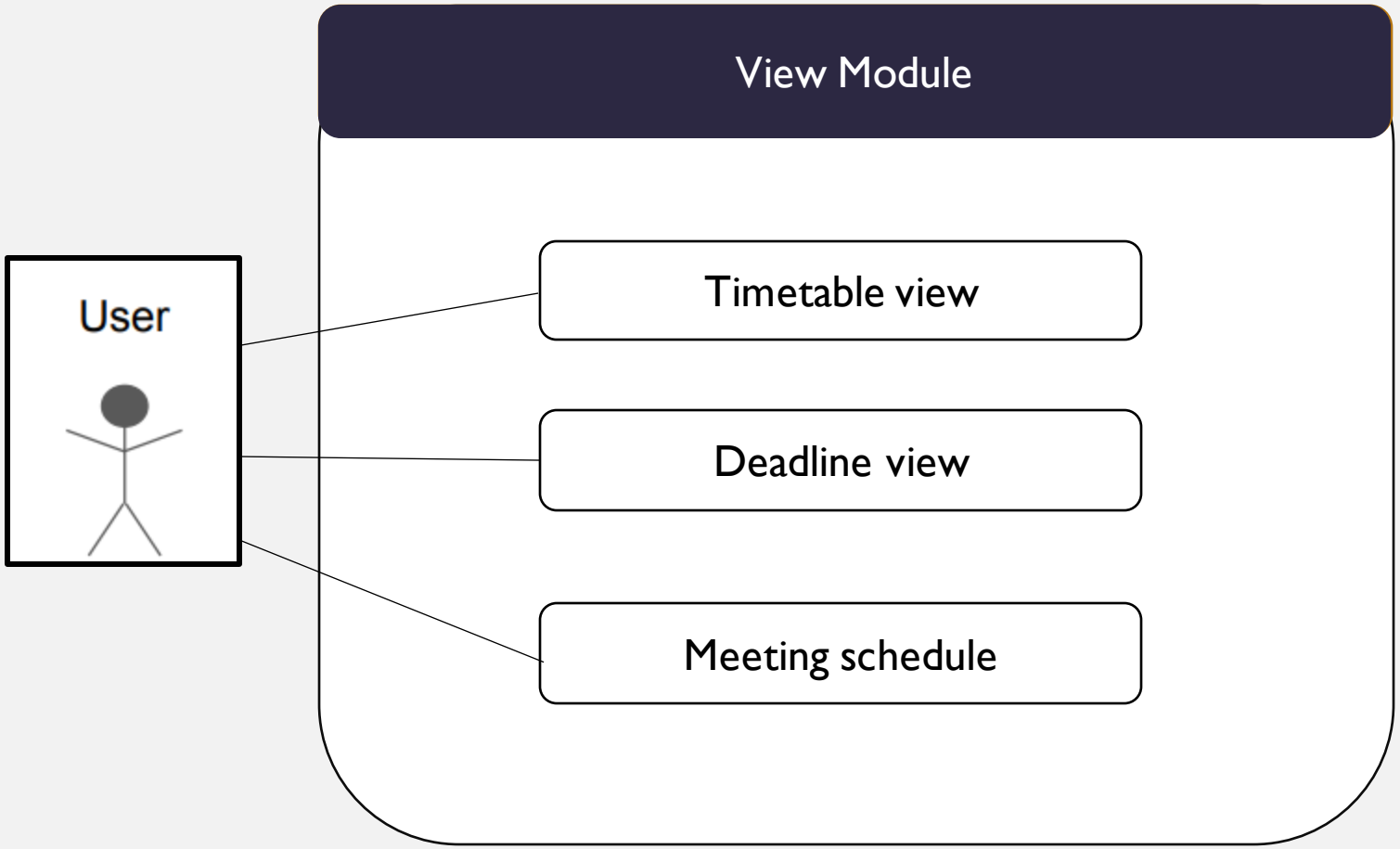
USER DIAGRAM 2



USER DIAGRAM 3



USER DIAGRAM 4



Display Months
MainLine Sequence
1. User: Selects Year
2. System: Displays prompt to enter Year
3. Customer: Enters the data.
4. System: Displays all months in that year

Alternative Seq1: at step 4
4. System: Displays message that user entered Incorrect year which is not a positive number.

Alternative Seq2: at step 4
4. System: Displays the message that some input value wasn't entered.
Displays prompt to enter missing value.

Display Days with events
MainLine Sequence
1. User: Selects month
2. System: Displays prompt to select month
3. Customer: selects the month in the prompt.
4. System: Displays all the days in that month

Specific day view
MainLine Sequence
1. User: Selects Date on the display
2. System: Displays all events on that day.

Alternative Seq1: at step 2
2.System: Displays message that user don't have any events on that day.

Set Reminders
MainLine Sequence
1. User: Selects three lines button on the top-left corner.
2. System: Displays prompt to select between Reminders, Background, Views
3. Customer: selects Reminders.
4. System: Displays all the reminders created by user

Create Reminder

MainLine Sequence

1. User: Selects "PLUS" button on the Reminders list page
2. System: Displays prompt to create reminder
3. Customer: Enters the data (date, time, frequency) and clicks on "save" button.
4. System: Displays success message for creating Reminder successfully.

Alternative Seq1: at step 4

4.System: Displays message that user did not Select any date. Displays prompt to enter missing value.

Alternative Seq2: at step 4

4.System: Displays message that user did not Select time. Displays prompt to enter missing value.

Alternative Seq3: at step 4

4.System: Displays message that user did not Select frequency. Displays prompt to enter missing value.

Edit Reminder

MainLine Sequence

1. User: Selects "Dropdown" button on the Reminders list page
2. System: Displays prompt to edit reminder
3. Customer: Enters the data (date, time, frequency) and clicks on "save" button.
4. System: Displays success message for editing Reminder successfully.

Alternative Seq1: at step 4

4.System: Displays message that user did not Select any date. Displays prompt to enter missing value.

Alternative Seq2: at step 4

4.System: Displays message that user did not Select time. Displays prompt to enter missing value.

Alternative Seq3: at step 4

4.System: Displays message that user did not Select frequency. Displays prompt to enter missing value.

Delete Reminder
MainLine Sequence

1. User: Selects "Delete" buttton of the reminder to be deleted
2. System: Displays success message for deleting Reminder successfully.

Set Backgrounds
MainLine Sequence

1. User: Selects three lines button on the top-left corner.
2. System: Displays prompt to select between Reminders,Background, Views
3. Customer: selects Backgrounds.
4. System: Displays Backgrounds page

Set Theme
MainLine Sequence

1. User: Selects any theme which is liked by the user
2. System: Displays the page with the new selected theme

Reset Theme
MainLine Sequence

1. User: Selects Reset theme button
2. System: Displays the page with the default theme

Views
MainLine Sequence

1. User: Selects three lines button on the top-left corner.
2. System: Displays prompt to select between Reminders, Background,Views
3. Customer: selects Views.
4. System: Displays the Views Page.

Time Table View

MainLine Sequence

1. User: Selects "Timetable" button on the Views page
2. System: Displays Timetable
3. Customer: Enters the data (date).
4. System: Displays the timetable view of that date.

Alternative Seq1: at step 4

4.System: Displays message that user did not select any date. Displays prompt to enter missing value.

Alternative Seq2: at step 4

4.System: Displays message that user entered invalid date.

Deadline View

MainLine Sequence

1. User: Selects "Deadline" button on the Views page
2. System: Displays all deadlines on the current date
3. Customer: Enters the data (date).
4. System: Displays the deadlines view of that date.

Alternative Seq1: at step 4

4.System: Displays message that user did not select any date. Displays prompt to enter missing value.

Alternative Seq2: at step 4

4.System: Displays message that user entered invalid date.

Meetings View

MainLine Sequence

1. User: Selects "Meetings" button on the Views page
2. System: Displays Meetings on the current date.
3. Customer: Enters the data (date).
4. System: Displays the meetings view of that date.

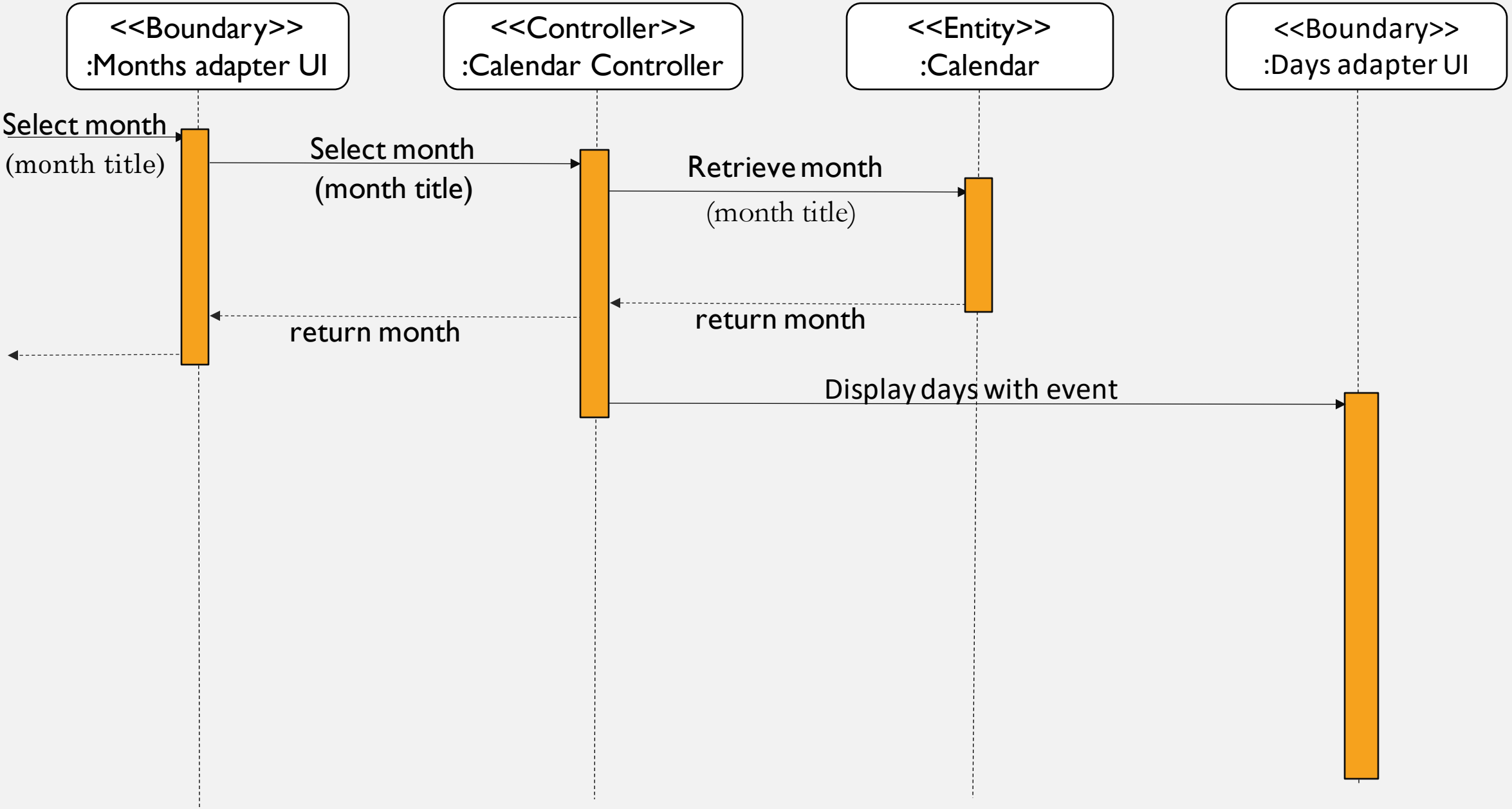
Alternative Seq1: at step 4

4.System: Displays message that user did not select any date. Displays prompt to enter missing value.

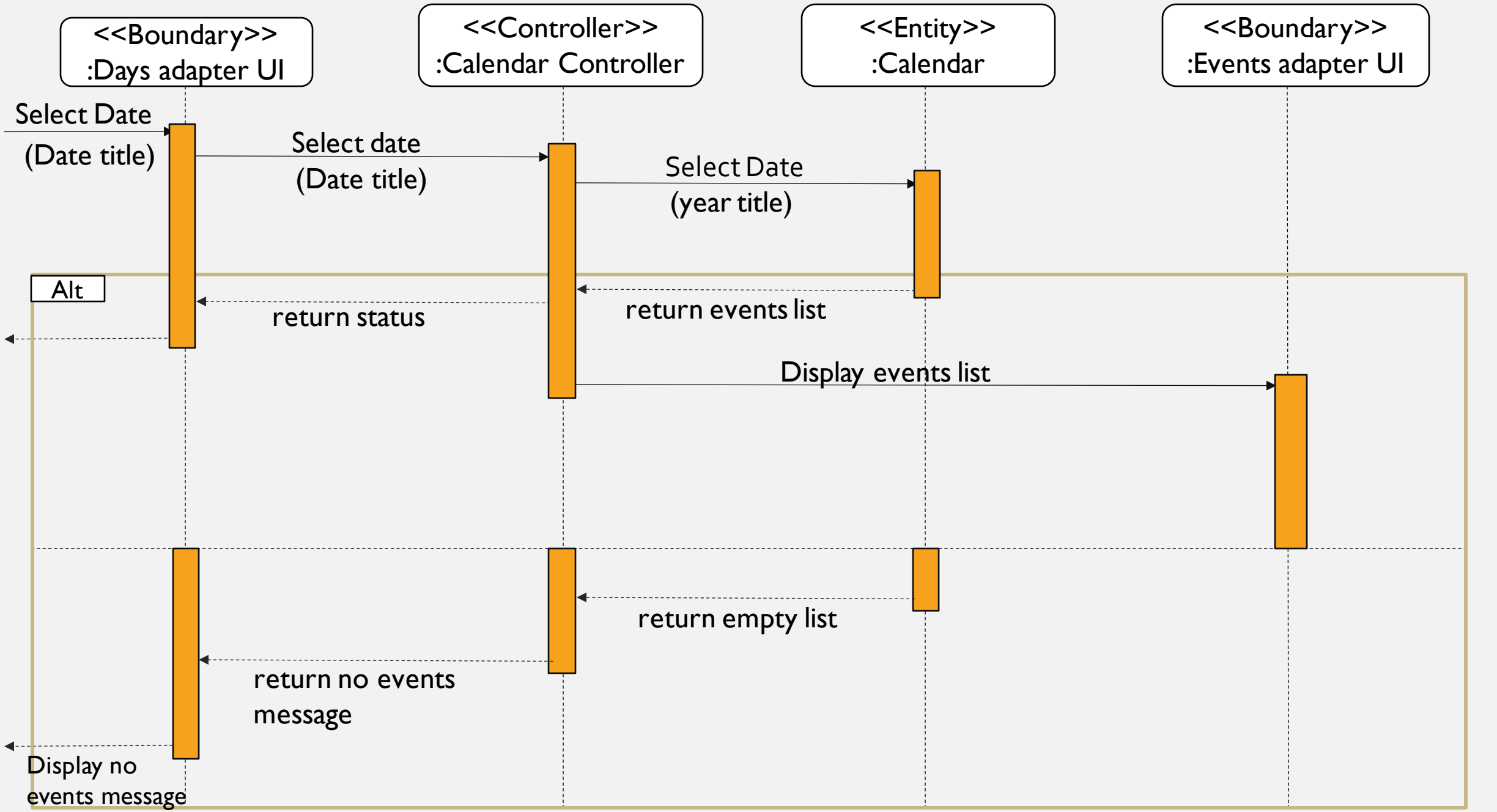
Alternative Seq2: at step 4

4.System: Displays message that user entered invalid date.

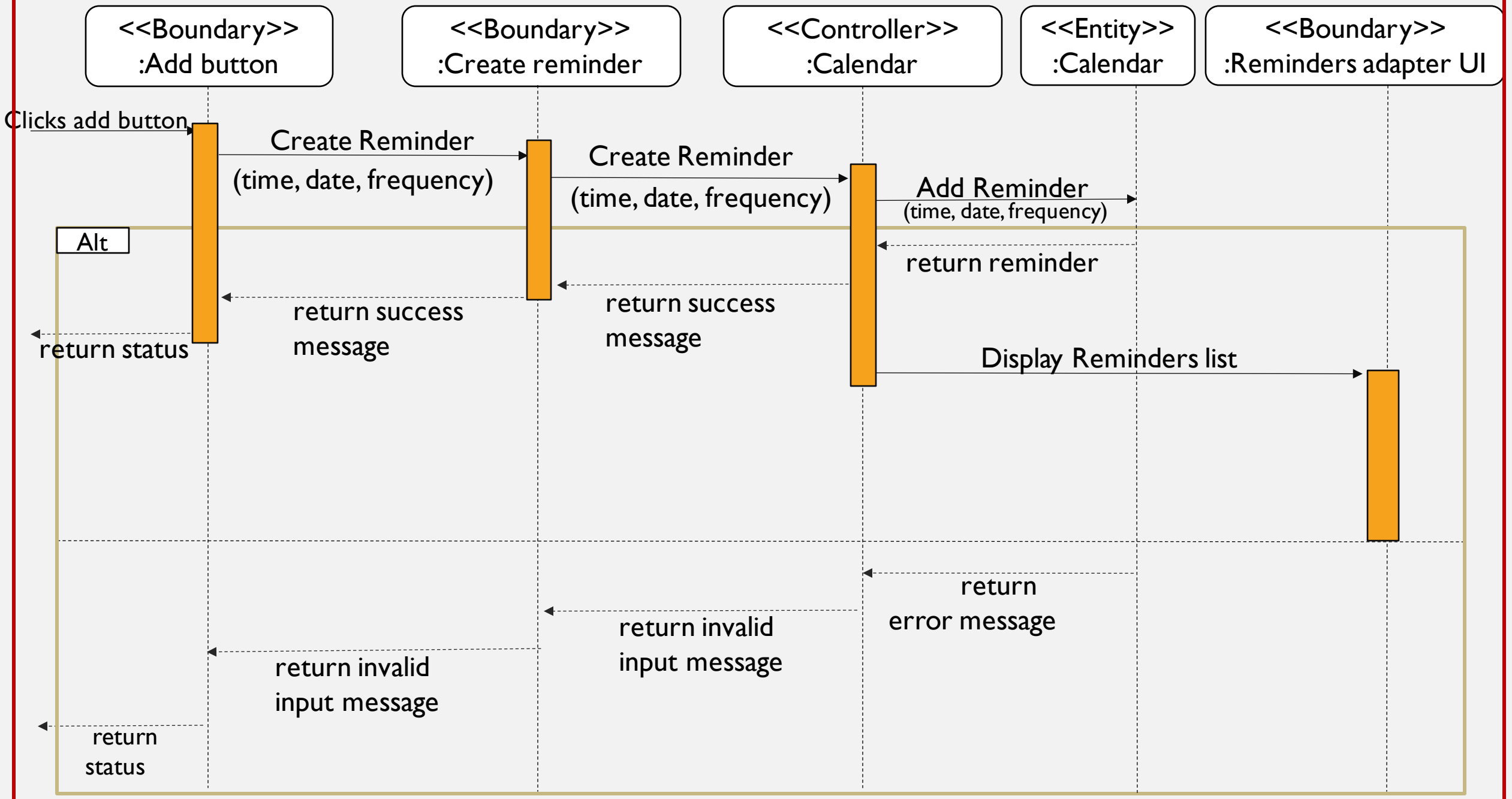
DISPLAY DAYS WITH EVENTS



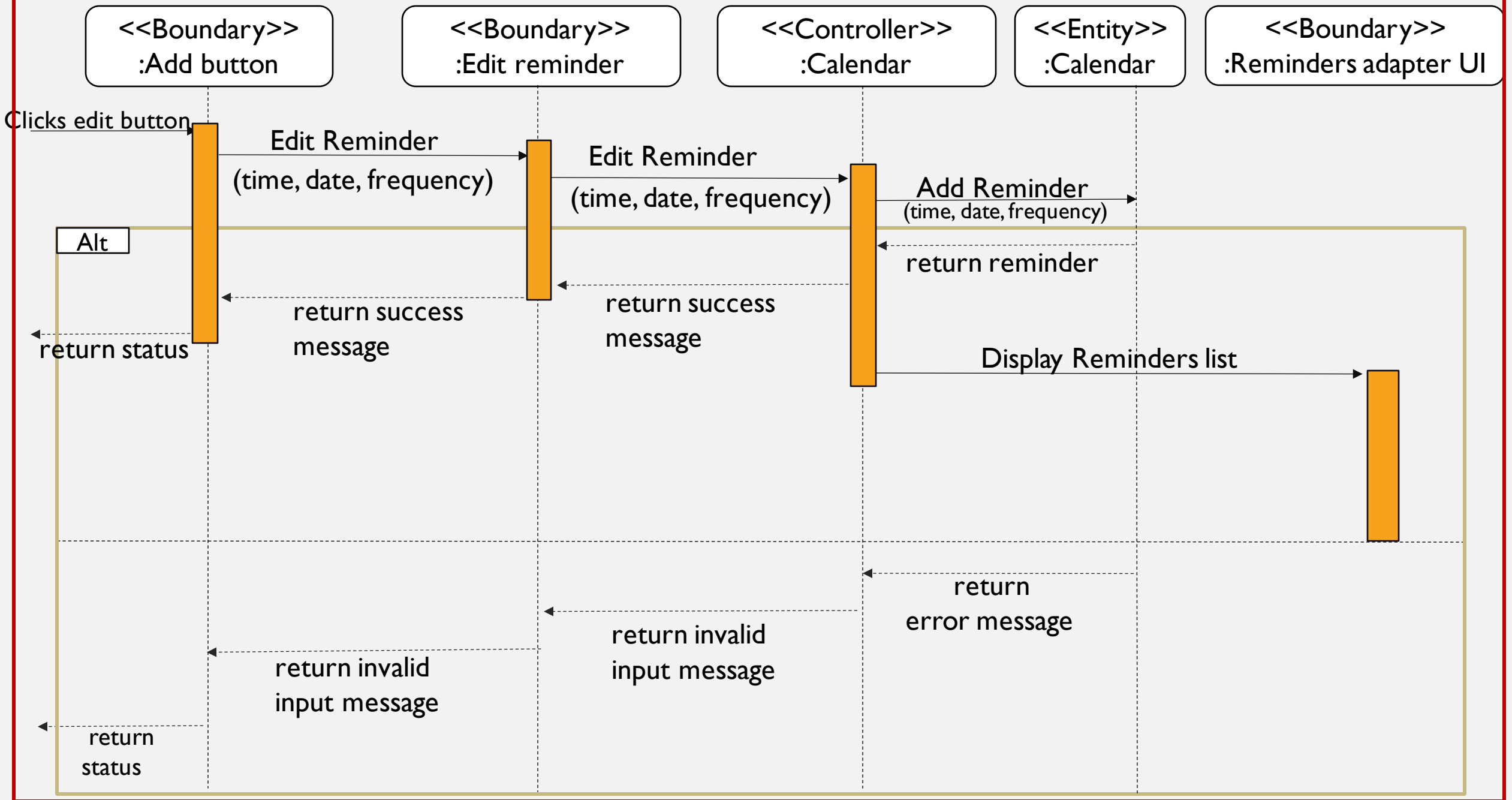
SPECIFIC DATE VIEW



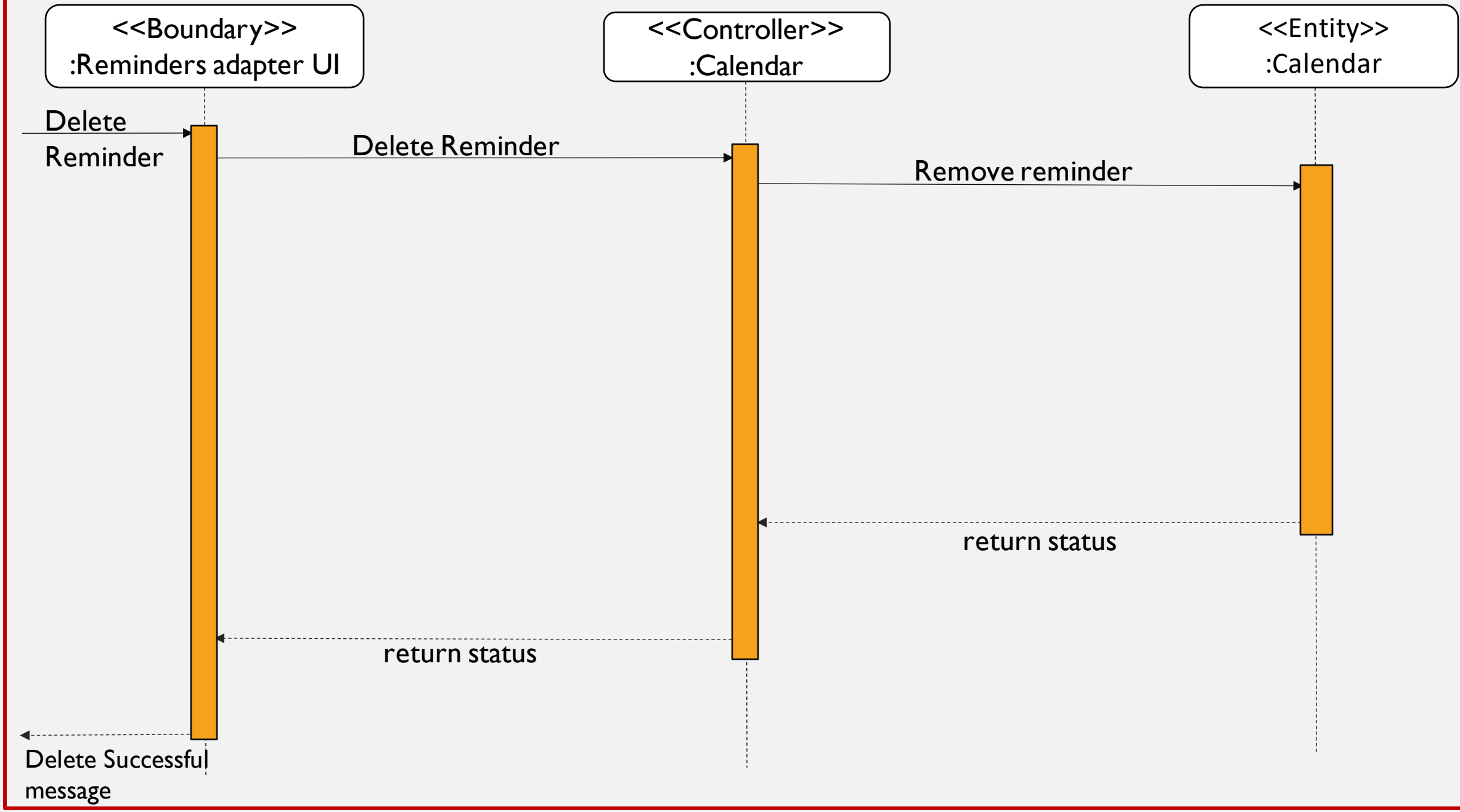
CREATE REMINDER



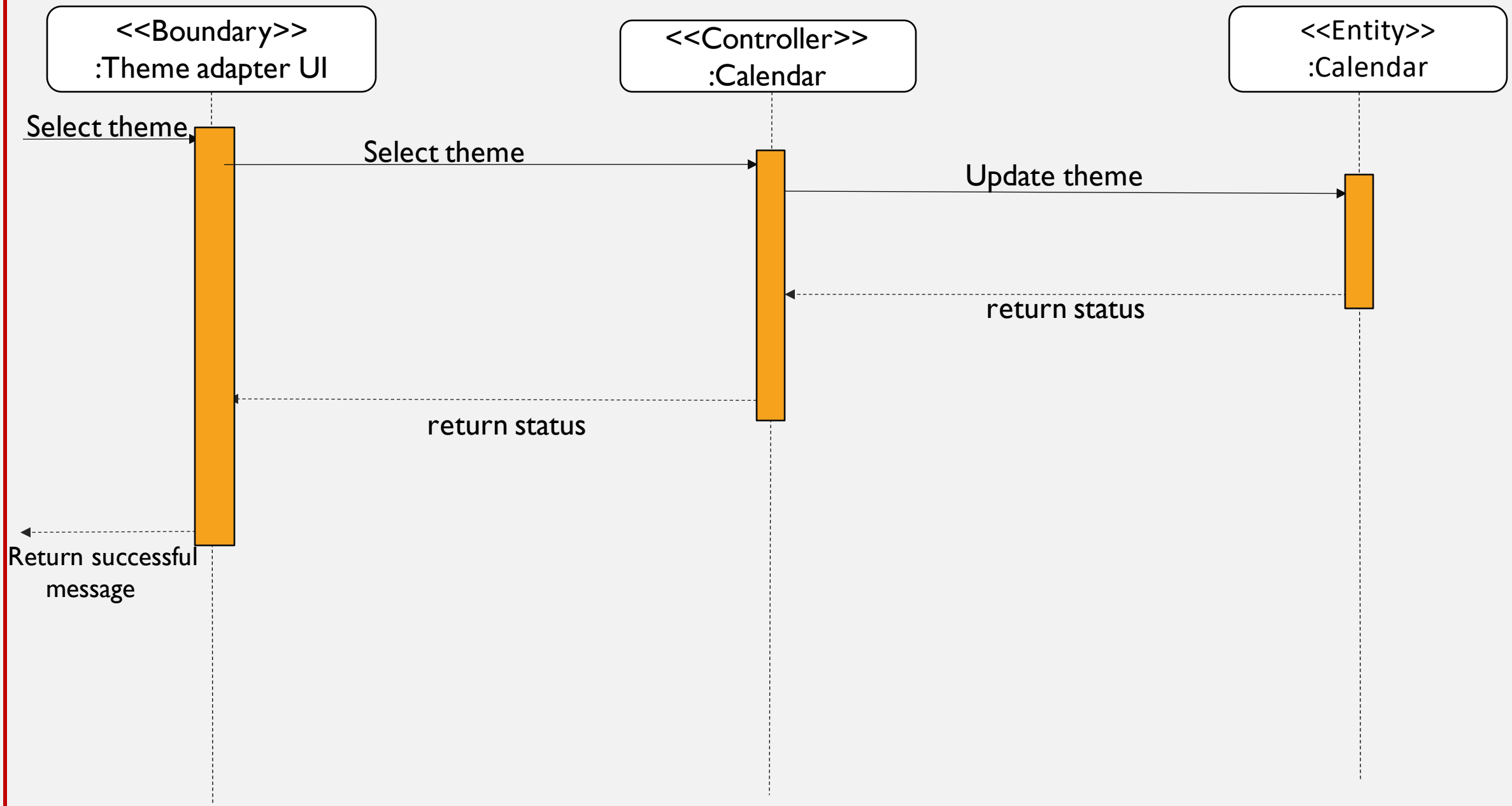
EDIT REMINDER



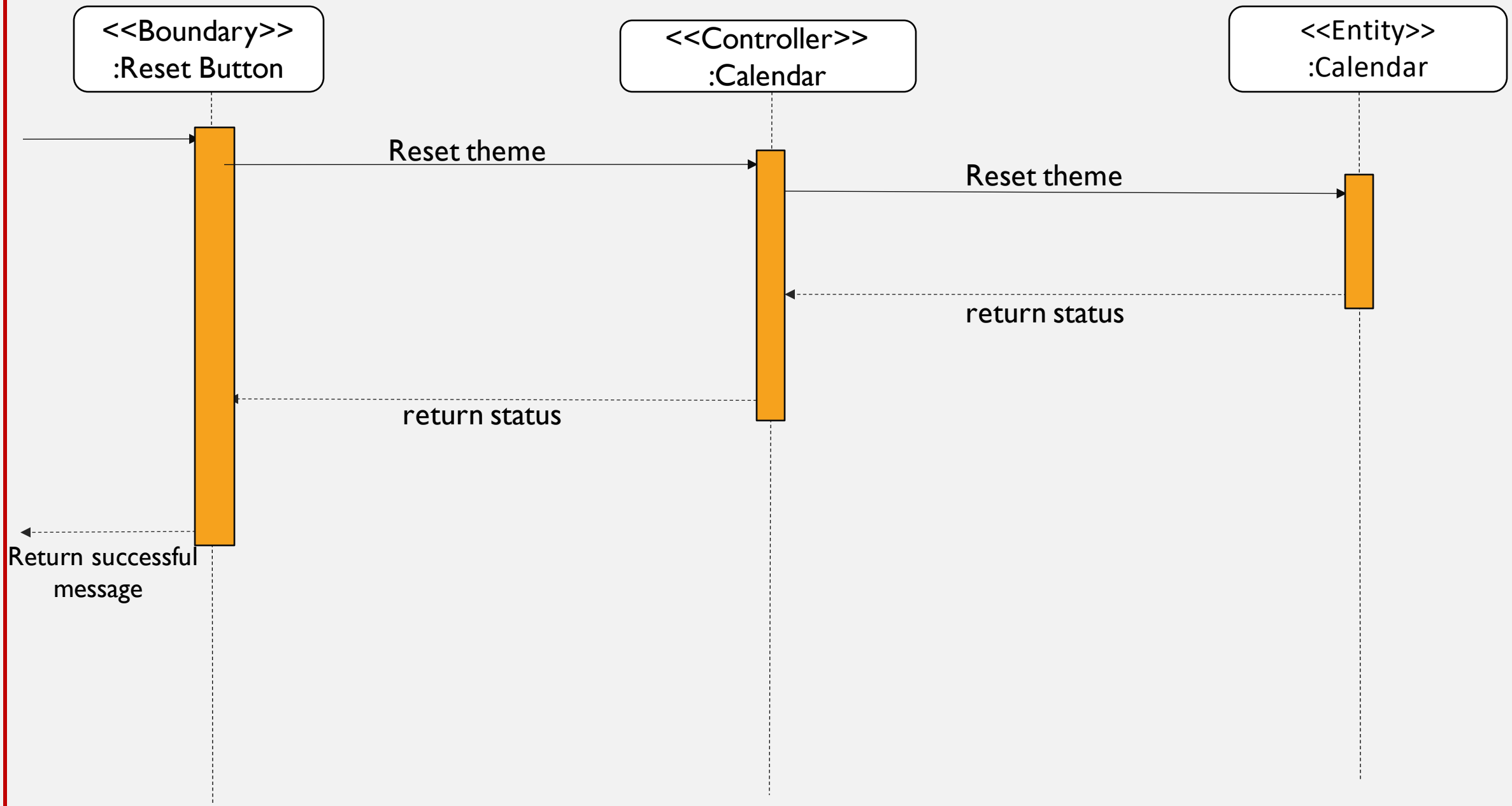
DISPLAY DAY WITH EVENTS



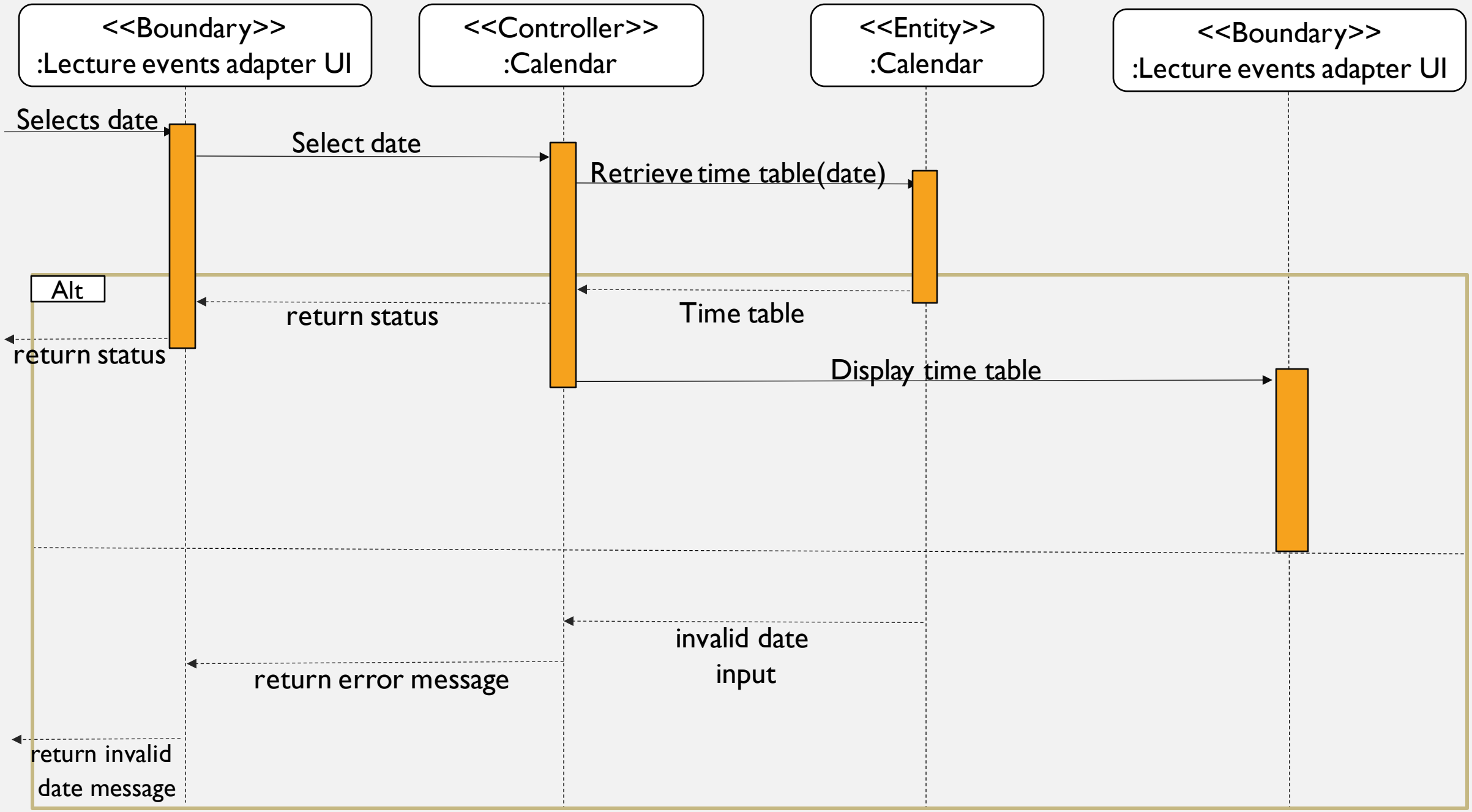
SET THEME


<<Boundary>>
:Theme adapter UI
<<Controller>>
:Calendar
<<Entity>>
:Calendar
Select theme
Select theme
Update theme
return status
return status
Return successful message

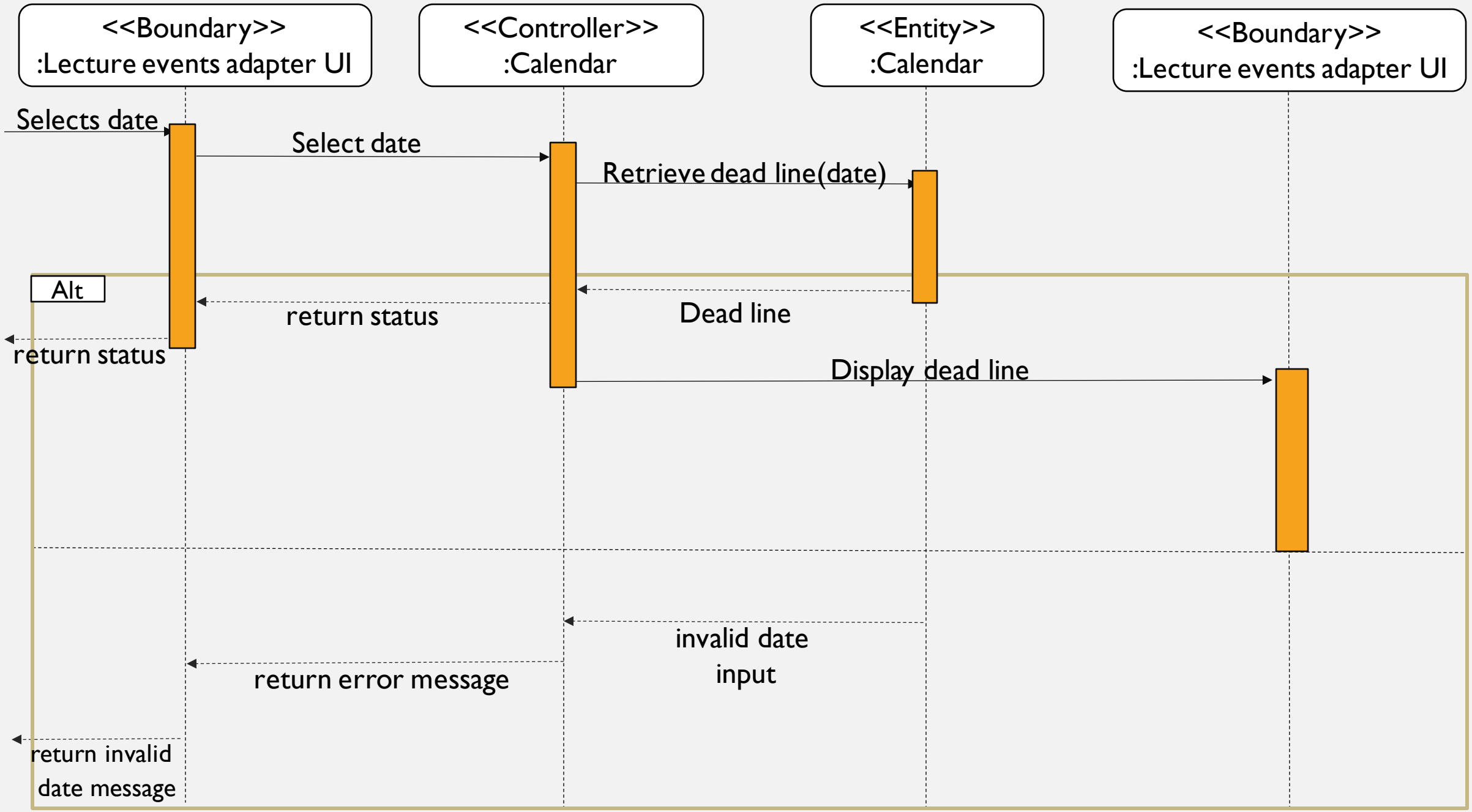
RESET THEME


<<Boundary>>
:Reset Button
<<Controller>>
:Calendar
<<Entity>>
:Calendar
Reset theme
Reset theme
return status
return status
Return successful
message

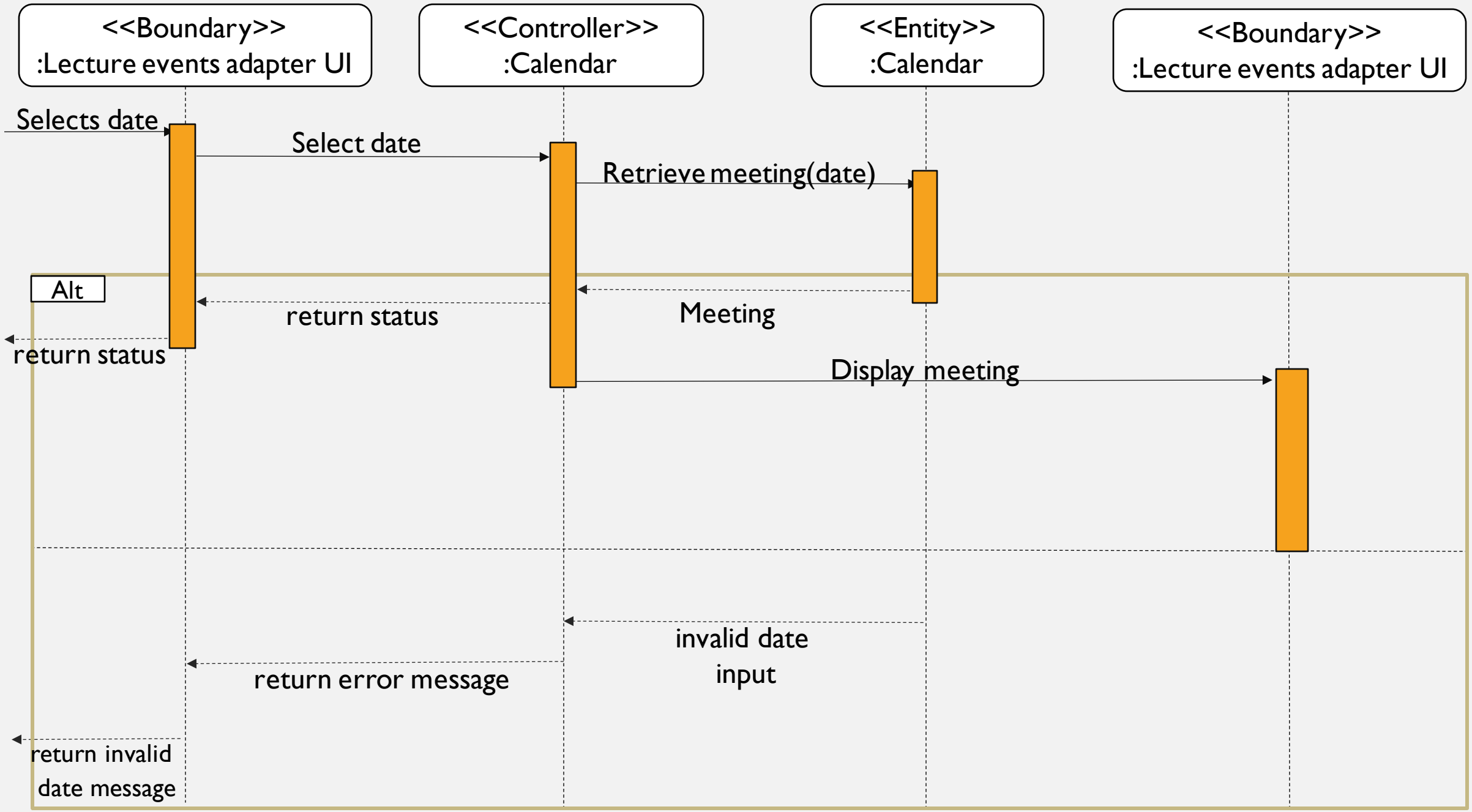
TIME TABLE VIEW



DEADLINE VIEW



MEETING VIEW



Calendar

Methods:

1. Displaycalendar():
- Return value: Returns months in current year
- Description: this method is used to display months in current year

2. Reminders():
- Return value: Returns Reminders list
- Description: this method is used to display all the reminders set by the user

3.Views():
- Return value: Returns three types of views
- Description: this method is used to display different types of views

Year

Methods:

1. DisplayMonths():
- Parameters: year
- Return value: Return the months in that year
- Description: this method is used to display months in selected year

2. Calendar():
- Return value: Returns calendar home class
- Description: this method is used to navigate back to calendar Home page

Month

Methods:

1. DisplayDaysWithEvents():
- Parameters: month
- Return value: Returns Days list with events
- Description: this method is used to display days with events in a month

2. DisplayCalendar():
- Return value: Returns Calendar class
- Description: this method is used to navigate back to Calendar page

Day

Methods:

1. DisplayDateView():
- Parameters: Date
- Return value: Returns events list
- Description: this method is used to display days with events in a year

2. DisplayCalendar():
- Return value: Returns Calendar class
- Description: this method is used to navigate back to Months page

Event

Methods:

1. DisplayDaysWithEvents():
- Return value: Returns Days list with events
- Description: this method is used to navigate back to days page

Reminders

Methods:

1. CreateReminder():
- Parameters: Date,Time,Frequency
- Return value: Returns success/error message
- Description: this method is used to create new reminder and then display Reminders list

2. EditReminder():
- Parameters: Date,Time,Frequency
- Return value: Returns success/error message
- Description: this method is used to edit reminder and then display Reminders list

3. DeleteReminder():
- Return value: Returns success message
- Description: this method is used to delete selected reminder and then display Reminders list

4. Calendar():
- Return value: Returns calendar home class
- Description: this method is used to navigate back to calendar Home page

Backgrounds

Methods:

1. SetTheme():
- Parameters: Theme
- Return value: Returns success message
- Description: this method is used to set new theme

2. ResetTheme():
- Return value: Returns success/error message
- Description: this method is used to reset theme

3. Calendar():
- Return value: Returns calendar home class
- Description: this method is used to navigate back to calendar Home page

Views

Methods:

1. DisplayTimetable():
- Return value: Returns success/error message
- Description: this method is used to display all lecture events on the current date

2.DisplayDeadline():
- Return value: Returns success/error message
- Description:this method is used to display all deadlines events on the current date

3. DisplayMeetings():
- Return value: Returns success message
- Description: this method is used to display all meetings events on the current date

4. Calendar():
- Return value: Returns calendar home class
- Description: this method is used to navigate back to calendar Home page

Timetable

Methods:

1. DisplayTimetable():
- Parameters: Date
- Return value: Returns success/error message
- Description: this method is used to display all lecture events on the current date

2. views():
- Return value: Returns view home class
- Description: this method is used to navigate back to views page

Deadlines

Methods:

1. DisplayDeadlines():
- Parameters: Date
- Return value: Returns success/error message
- Description: this method is used to display all deadlines on the current date

2. views():
- Return value: Returns view home class
- Description: this method is used to navigate back to views page

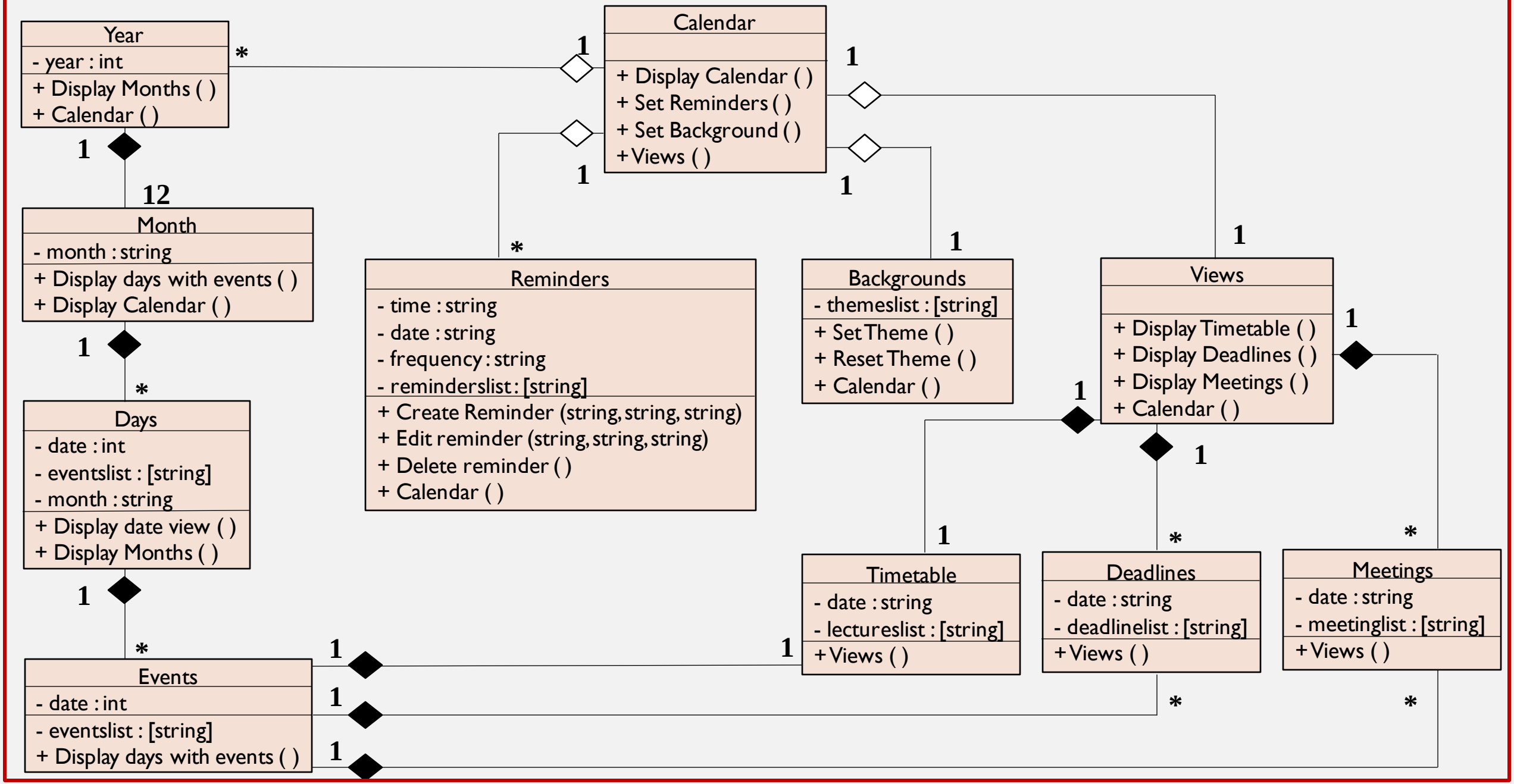
Meetings

Methods:

1. DisplayMeetings():
- Parameters: Date
- Return value: Returns success/error message
- Description: this method is used to display all Meetings on the current date

2. views():
- Return value: Returns view home class
- Description: this method is used to navigate back to views page

CLASS DIAGRAM



USABILITY DOCUMENT

We have highlighted the usability of our Invoice Application using the eight golden rules of Shneiderman.

1. Strive for Consistency :
People spend most of their time using other similar products and their experiences with those products set their expectations. It can frustrate our user if our design is inconsistent and not familiar to users.

a. Internal Consistency: We must maintain Consistency with inside of our software.
--- Icons : windows,mac,linux make use of the same button icon to perform the same action across all the windows. In our application we maintained the same consistency with buttons, icons. For example back button anywhere in our application means navigating back to the back page.
---Color : In Our proposed application design we maintained uniform color consistency so that it is easier for users to keep track. Using different colors for different buttons makes user feel unpleasent. In our applicatiion we have chosen basic color theme to be purple,black,ash and all the application is designed with the same set of colours to make the application design look pleasant.

USABILITY DOCUMENT

b. External/Environmental Consistency: Follow established industry conventions. To facilitate user interaction, all functionalities of our system will cater to the established notion in the society. For example,
---Icons: The add button has a symbol of plus which can be understandable by the user.
The Delete icon has a symbol of dustbin which helps the user to identify its operation just by looking at it.

2. Design for universal usability:
Recognize the needs of diverse users and design, facilitating the transformation of content.
- For first time novice users our application will showcase a step by step approach on how to navigate between tabs and how to create and delete reminder were shown.
- Hence our application is designed for all kind of users keeping in mind universal usability requirements.

USABILITY DOCUMENT

3. Offer Informative Feedback:
Informing the users all the time about whats going on in the application
Human Understandable Reactions-
when the user clicked on any button the button will be hovered and highlighted so that user can keep track where exactly hie is clicking and where exactly the pointer of mouse is.

4. Design dialogues to yield closure:
a. when the user create a reminder the application delivers a success message if the reminder is successfully created and it shows unsuccessful message if the reminder inputs were wrong.

USABILITY DOCUMENT

5. Offer error prevention and simple error handling:

When error occurs, users should be provided with simple, step-by-step instructions to solve the problem.

a. conflicting functionalities are not kept together inorder to prevent the wrong clicks from the user.

For example, delete button and create button are kept farther.

b. When the user fails to create reminders or fails to esit reminders the system intimate the user about where the inputs were given wrong so that user tries again correcting his inputs.

6. Permit easy reversal of actions:

permit the reversal of actions relieves anxiety since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options.

a. Back Button: This option provides user freedom to reverse the changes made by selecting the wrong option. Example: Suppose a user wants to see the months list but mistakenly opened days list now the user can go back to home page by clicking back button.

USABILITY DOCUMENT

7. Keep users in control:

Making the users the initiators of actions rather than the responders to actions. users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions.

a. Confirm or Delete Message:

When the user wants to delete a reminder he gets the confirmation message if he really wants the item to be deleted.

b. save button:

When the user creates reminder he have to click on save button to create new reminder.

8. Reduce short term memory load:

limitation of human information processing in short term memory requires that display be kept simple. Keeping our interface consistent will help us to make our design more intuitive so our user doesn't have to call every time he/she uses the product. It's simpler for us to recognize information rather than recall it.

a. add button have plus symbol and delete have dustbin symbol .

All these visual elements are easy to recognize because they resemble real-world things that serve the same purpose.

COHESION

UML design should strive to achieve high cohesion and low coupling among its classes. That is what we have achieved in this application design too. The following are the occurrences of different types of cohesion in our design followed by the justification of how few classes in our design are tightly coupled and mostly loose coupling is seen.

Cohesion: Encapsulation enables a class to be highly cohesive i.e. their purpose is very clear. Put another way, the class does one thing, only one thing, and it does that only one thing well. Most of the classes we made have a single purpose.

1. Logical cohesion: A class exhibits logical cohesion if the tasks its methods perform are conceptually related which can be seen In reminder, editreminder(), deletereminder(), createReminder() are logically related to reminder module.

COHESION

1. Temporal cohesion: Temporal cohesion is present in a class if some of its methods are executed in the same time span . This can be seen in the list of invoices class , where both deleteReinder() and ReminderList() are performed together for deleting a Reminder , and also in saveReminder() and closeReminder() when the user closes theReminder , saving unfinished Reminder as a draft.

2. Communicational cohesion: A communicationally cohesive module is one which performs several functions on the same input or output data. The classes, Timetable, Deadlines and Meetings works at the same time when events class calls these classes for calculating Timetable list,Deadlines list, Meeting list.

3. Functional cohesion: Functional cohesion can be seen in Reminders,Views module. The goal of the Reminder module is to provide for functions that allow the user to organize his/her Reminders according to their liking . For example making new Reminders . Similarly, All the functions related to creation an modification of Reminder(create Reminder, edit Reminder, delete Reminder to name a few) are grouped under the Reminder module.

COUPLING

Coupling: Most Classes are lightly coupled. But tight coupling can be seen between invoice and payments , expenses, and sales , reports, where the creation of latter's object is dependent on the initial one. For example createReminder() in Reminder class is the method that is going to create a Reminder Object.

1. Data Coupling: There is minimal data coupling between modules. The data is only passed while creation of an object or while passing messages to perform a subroutine. One of the few examples of this that can be seen is between Reminers list and create Reminder, where the create Reminder class is going to provide the data to Reminder list class for displaying contents on the screen.

2. Control Coupling: Modules do not pass control information to each other. Therefore, control coupling is almost non-existent.

3. Content Coupling: This type of coupling can be seen across create Reminder, edit Reminer, delete Reminder classes as all of them contain the listRemineder() method , the code of which is going to be the same for all of them.