CP317 Assignment 2

Portable Calendar

Hongshan Shang 163042150 shan2150@mylaurier.ca

Ziyan Yang 183077870 yang7787@mylaurier.ca

Shuangqian Lu 176801800 luxx1800@mylaurier.ca

Dekai Meng 186800570 meng0570@mylaurier.ca

Zhuoran Yang 186800610 yang0610@mylaurier.ca

Cong Qi 173224830 qixx4830@mylaurier.ca

Haopeng zhu 186804500 zhux4500@mylaurier.ca

Yurong Jia 170375100 jiax5100@mylaurier.ca

Yulin Xue 156801230 xuex1230@mylaurier.ca

Object-oriented analysis:

1. use-case models

Use-case description	Ziyan Yang
Use-case diagram	Hongshan Shang

2. class models Cong Qi
3. dynamic models Shuangqian Lu

An object-oriented design:

4. interaction diagrams Dekai Meng
5. detailed class diagrams Hongshan Shang
6. client design Yurong Jia
7. detailed design Haopeng Zhu

Metrics for design and analysis Zhuoran Yang

Pseudocode Yulin Xue

Survey 3 All of us already finished the survey

1. Use-case modeling

Scenarios of use case : View upcoming events

The user wishes to see what are the most recent events and the details of those events in time order in one week.

1. The user goes to the home page and sees the time for the upcoming events in a week.
2. The user can tap the events for details including starting time, ending time, title and location.

Possible alternatives:

There are no events in the following week, then the homepage will display "No upcoming events in one week".

Scenarios of use case : Create Event

Normal scenario: The user wishes to create an event

1. The user determines the title and location of the event.
2. The user sets the starting time and ending time of the event.
3. The user chooses whether an email notification is needed while the default setting of email notification is on.
4. The system updates the blank event with information the user provided.
5. The system finds no event with the same starting and ending time. It displays a toast message and adds the event into the calendar page.
6. If the event is within one week, it will also be added to the "View upcoming events".

Exception Scenario: The user wishes to create an event

1. The user determines the title and location of the event.
2. The user sets the starting time and ending time of the event.
3. The user chooses whether an email notification is needed while the default setting of email notification is on.
4. The system updates the blank event with information the user provided.
5. The system searches the database and finds an event with the same starting time and ending time.
6. The system asks the user whether he/she wants to replace the old event or re-edit this event.
7. If the user chooses to replace the old event, the system displays a toast message and adds the new event into the calendar page.
8. If the user chooses to re-edit this event, the system clears the information entered and the user can begin with step 1.

Possible alternatives:

1. The user fails to provide certain necessary information, including one of title, location, starting and ending time. The system will generate a reminder and indicate what is missing.
2. The ending time the user enters comes before the starting time. The system will ask the user to reenter the starting time and ending time and remind the user of the order.

Scenarios of use case : Set Alarm Preference

Users can choose between three kinds of alarms including Mute, Vibrate and Sound.

1. The user chooses the alarm settings between Mute, Vibrate and Sound.
2. The system updates the alarm setting based on the user's choice

Scenarios of use case : Search Event

The user wishes to search a specific event.

1. The user enters information about title, location and time. At least one of the information needs to be provided.
2. The system searches the database for the event that satisfies the conditions. Either a corresponding event will show up or a "failure to find this event" message will be displayed.

Possible alternative:

The user enters no information and the system will remind the user that he needs to provide at least one thread.

Scenarios of use case : Choose Language

The user wishes to choose a preferred language for the app.

1. The user choose the language from the language list
2. The system will update the entire app with the chosen language.

Scenarios of use case : E-mail Setting

The user will enter its email for notifications.

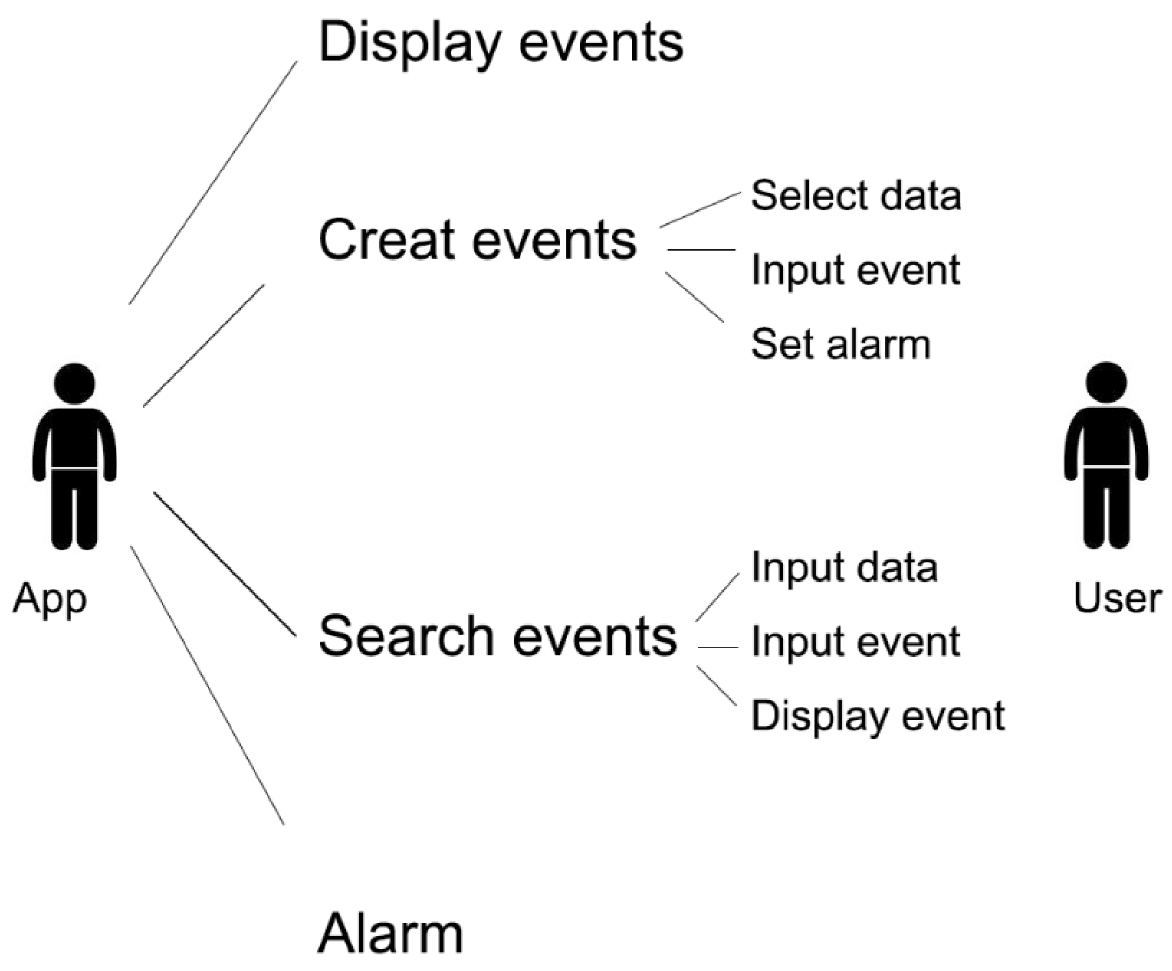
1. The user enters the email he wants to receive notifications.

2. The system will send a confirmation to that email address.

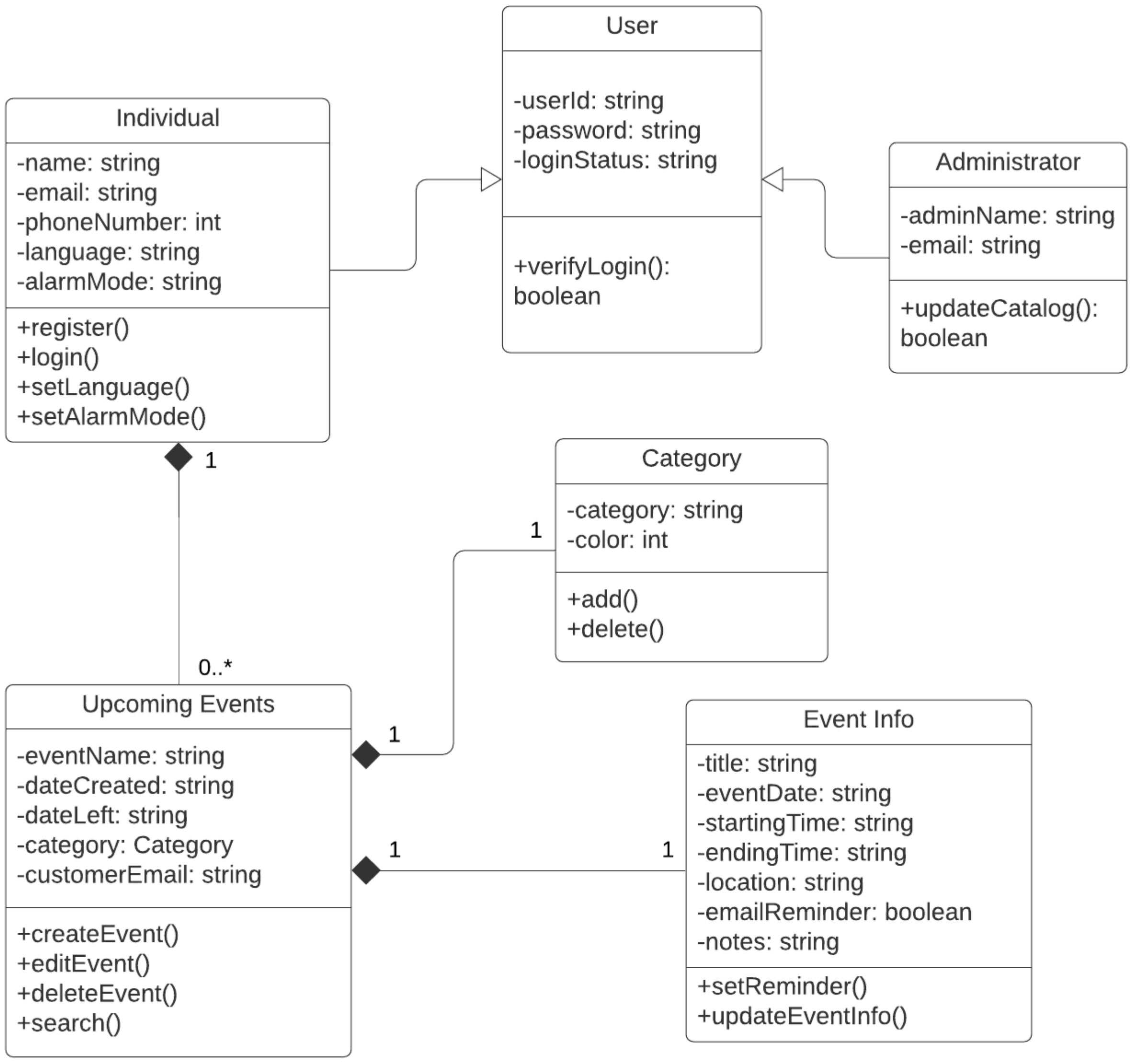
Possible Alternative:

The email the user entered doesn't exist

1.2 Use case diagram

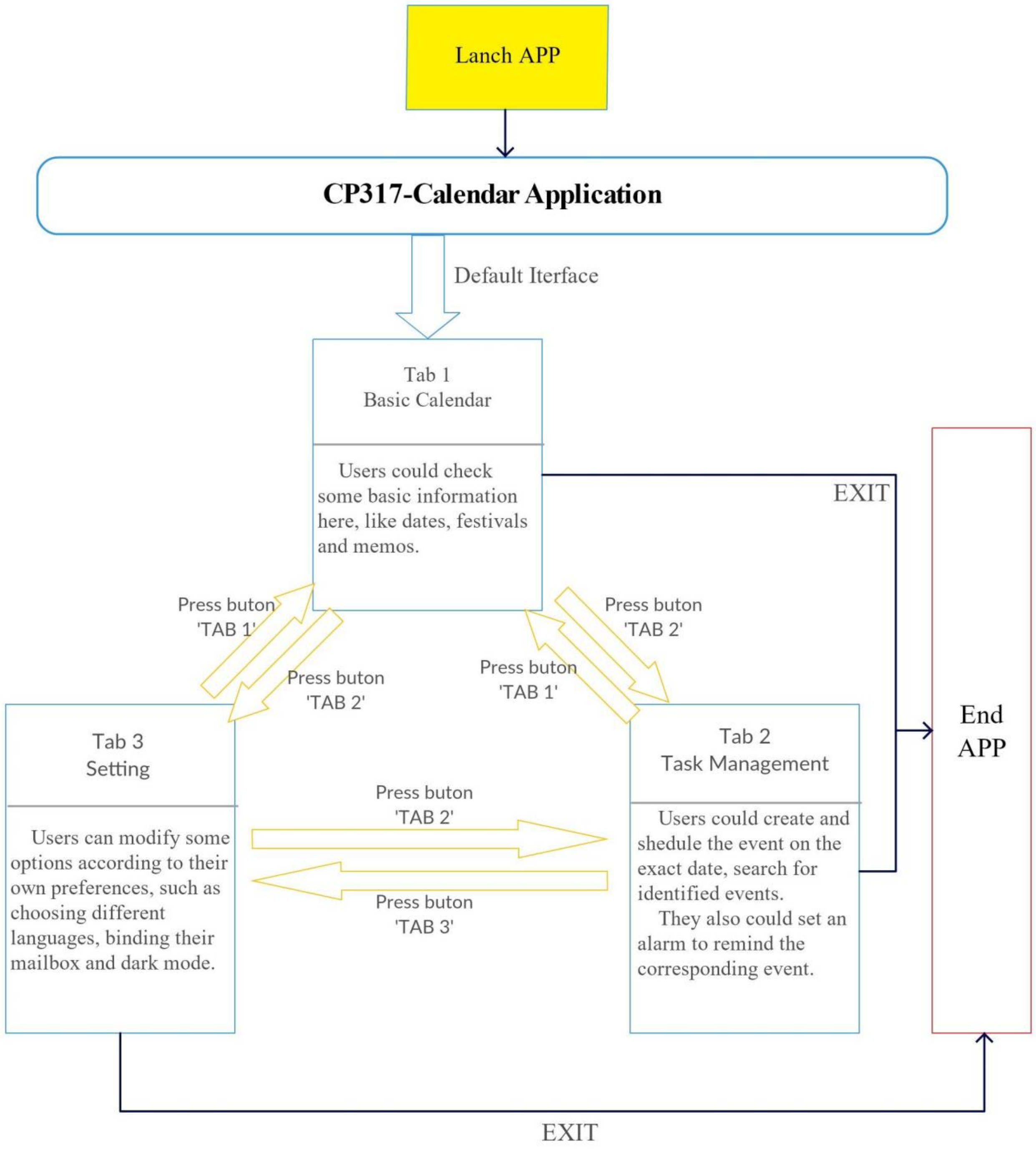


2. Class Models



3. Dynamic Models

After entering the program, the user can switch between the three tabs to complete different goals. And can exit the program in any tab.



Lanch APP
CP317-Calendar Application
Default Iterface
Tab 1
Basic Calendar
Users could check some basic information here, like dates, festivals and memos.
EXIT
End APP
Press buton 'TAB 1'
Press buton 'TAB 2'
Press buton 'TAB 2'
Press buton 'TAB 1'
Tab 3
Setting
Users can modify some options according to their own preferences, such as choosing different languages, binding their mailbox and dark mode.
Press buton 'TAB 2'
Press buton 'TAB 3'
Tab 2
Task Management
Users could create and shedule the event on the exact date, search for identified events.
They also could set an alarm to remind the corresponding event.
EXIT

4, Interaction diagrams

4.1: Introduction for portable calendar:

The UML supports two types of interaction diagrams for our application “Portable Calendar”:

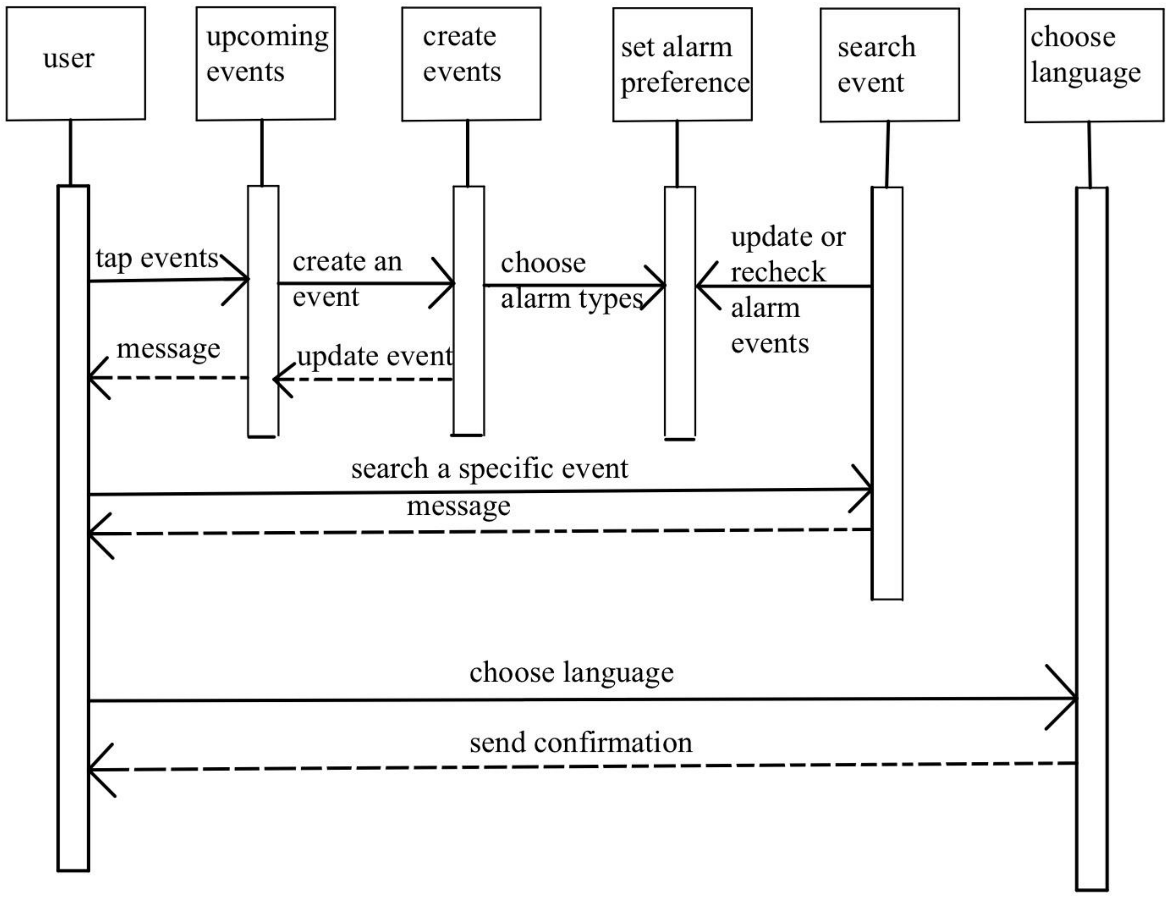
one is Sequence diagrams, another is Collaborations diagrams.

Sequence diagrams show the object interaction with time-based.

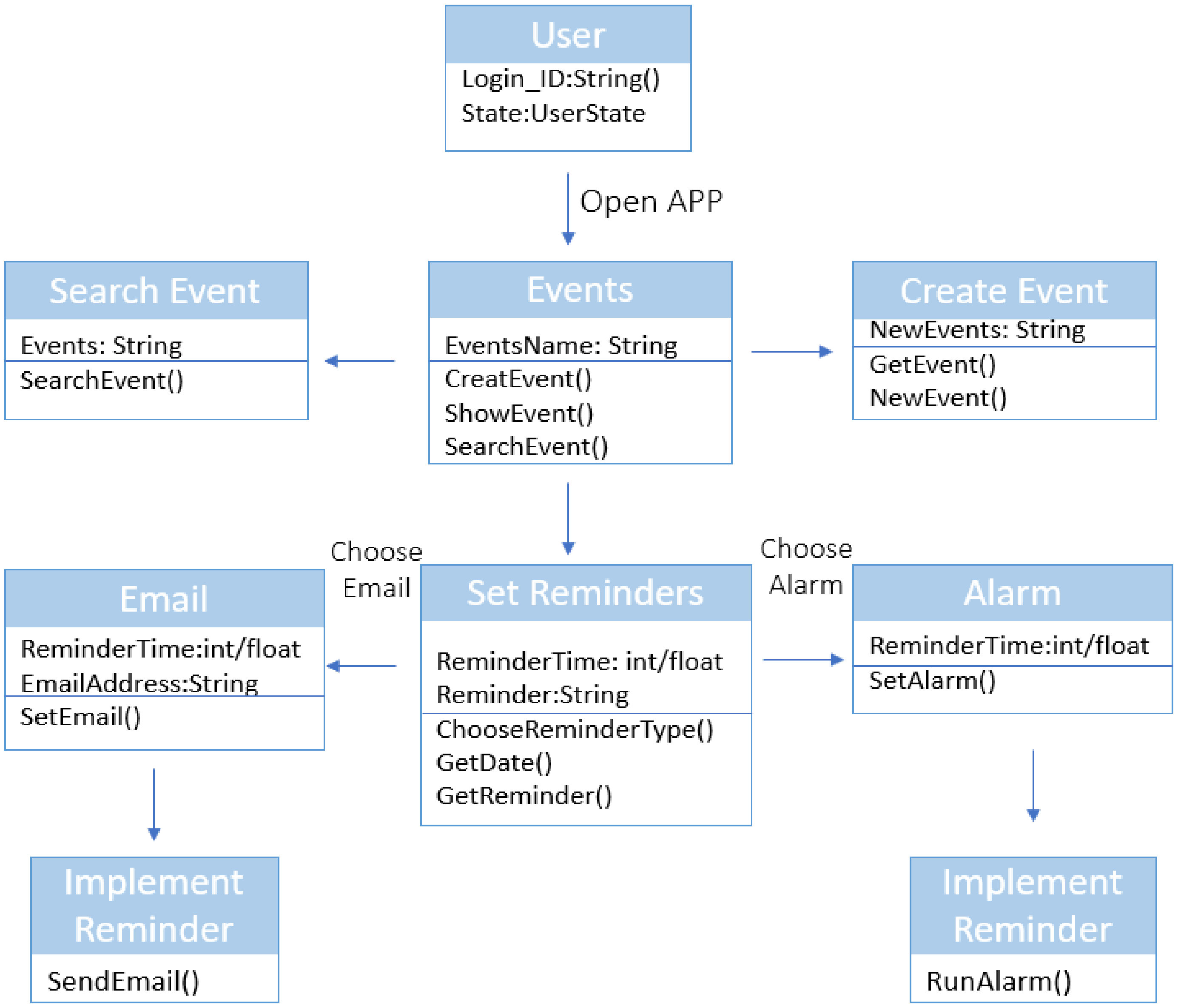
Collaboration diagrams show the way how objects associate with each other.

Both of them are the diagrams which alternate representations of the interaction diagrams.

4.2: Sequence diagrams:



5.detailed class diagrams

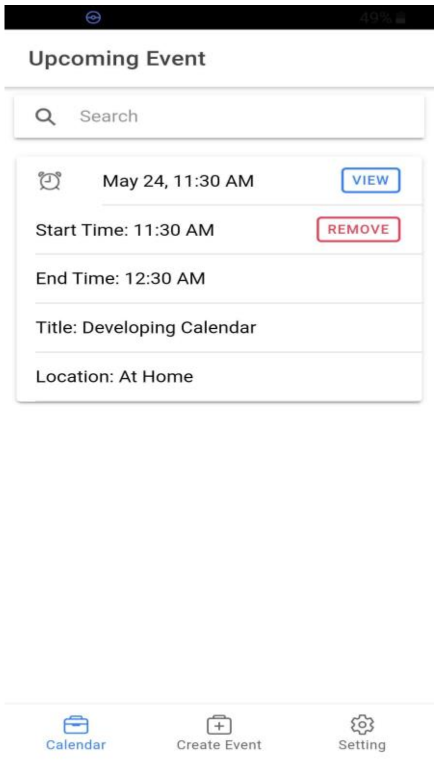


6. Client Design

The client of “Portable Calendar” application is provided in both Android and IOS systems.

Users can download the client from the app store. The client design consists of three interfaces which are “Calendar”, “Create Event” and “Setting”.

1. In the “Calendar” Page, users can look through the overview of upcoming events. There is a search bar on the top of the page to allow users to find a particular event. By clicking the “view” button and “remove” button, users can view the details or remove the event.



Upcoming Event

Search

May 24, 11:30 AM

Start Time: 11:30 AM

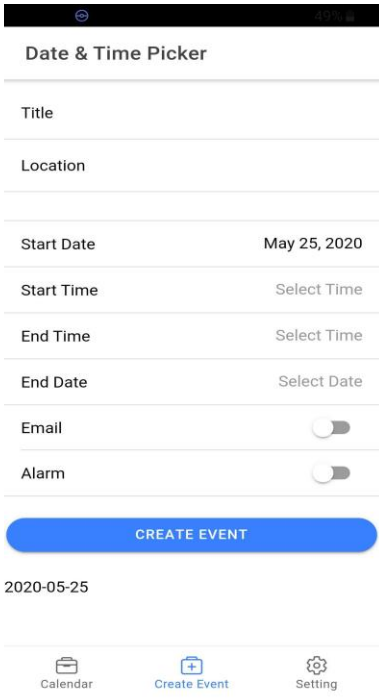
End Time: 12:30 AM

Title: Developing Calendar

Location: At Home

2. In the “Create Event” Page, users can create event details. There are several instructions in front of each user input box which are events “Title” and “Location”, time “Start Date” “Start Time” “End Time” and “End Date”. Users can choose whether to get an alarm and receive email when the event comes. At the bottom of this page, users can

click “Create Event” button to successfully create the event if all the information is filled correctly.



Date & Time Picker

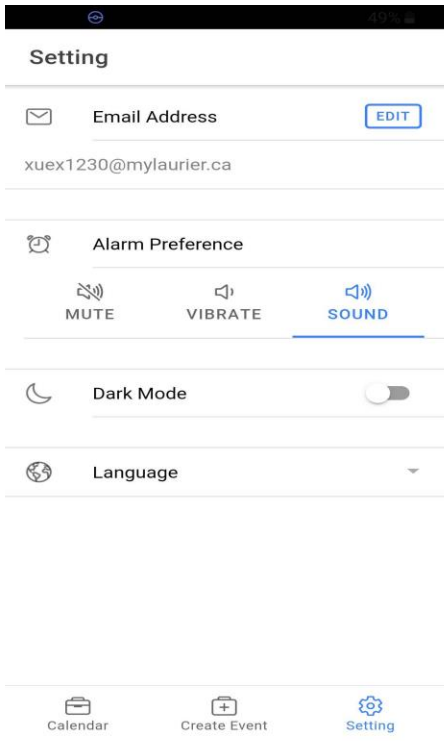
Title

Location

Start Date	May 25, 2020
Start Time	Select Time
End Time	Select Time
End Date	Select Date
Email	
Alarm	

2020-05-25

3. For the “Setting” Page, this page allows users to turn on/off some functions. There are four function settings which are “Email Address”, “Alarm Preference” “Dark Mode” and “Language”. Users can decide whether they want to get event notifications, and model and language preferences.



Setting

Email Address EDIT

xuex1230@mylaurier.ca

Alarm Preference

MUTE VIBRATE [✓ SOUND]

Dark Mode

Language

7. Detail Design

Module name : read_file_name

Module type: Function

Return type : String

Input arguments: None

Output arguments: None

Error message : None

Files accessed : None

File changed: None

Modules called : None

Narrative : The product is invoked by the user by means of the command string word_count<file_name>

Using an operating system call, this module accesses the contents of the command string input by the users extracts<file_name>, and return it as the valueof date, festival, weather in the module,

that the user checked

Metrics for design and analysis

- Measures of design quality

1. Cohesion metrics

   Split up function: class Event; createEvent; class EventList; EventList search; remove

2. Coupling metrics

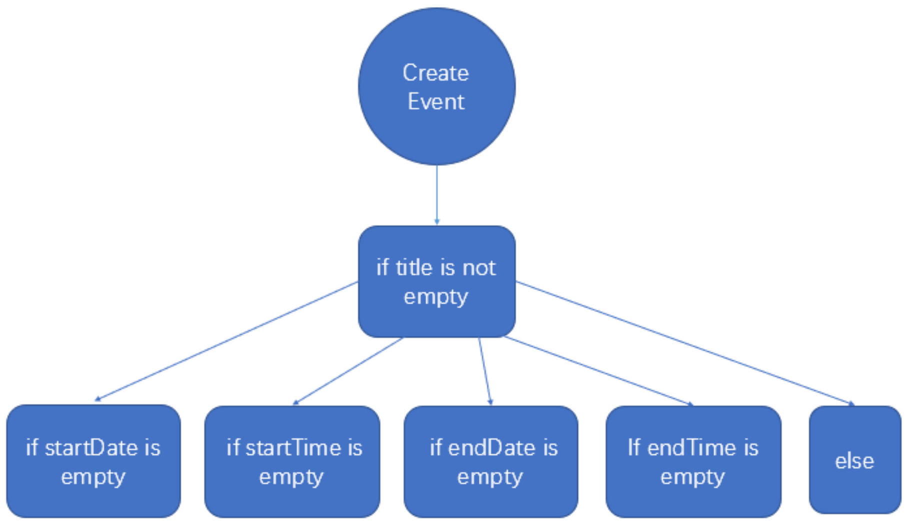
   Compute object-oriented coupling, control coupling and unit coupling

3. Fault statistics metrics

- Cyclomatic Complexity

  $V(G)=e-n+2p$

Example : Program control flowchart of CreatEvent


Create Event
if title is not empty
if startDate is empty
if startTime is empty
if endDate is empty
If endTime is empty
else

The Cyclomatic Complexity of CreatEvent is *V(G) = 7-7+2 = 2*

Pseudocode

```
public class Event {
    String title, location, startDate, startTime, endTime, endDate;
    Int id;
    public void createEvent(String title, String location, String startDate, String ….){
        // title: CAN NOT be empty
        // location: can be empty
        // startDate, endDate: can be empty, by default is today
        // startTime: can be empty, by default is right now
        // endTime: can be empty, by default is the end of today
        if title is not empty
            this.title = title;
            this.location=location;
            this.startDate=startDate; if startDate is empty, then = today;
            this.startTime=startTime; if startTime is empty, then = now;
            this.endDate=endDate; if endDate is empty, then =today;
            this.endTime=endTime; if endTime is empty, then = the end of today;
            toastGenerator.print("Successfully create Event");
        else
            toastGenerator.print("Fail to create Event, require title");
            Make the input of title red;
        }
}
public class EventList {
        // Add Event to the list, based on the time sequence
        public void addEvent(Event calendarEvent) {
                EventList.append(calendarEvent)
                EventList.startDate.sort()
        }
        public EventList search(String keyword){
                New EventList() = copy of eventlist;
                for (int i=0; i<length; i++)
                        if EventList(i).not contains keyword
                                EventList.remove(i)
        return EventList
        }
        public void remove(ActionListener event, int id){
                EventList.remove(i)
        }
}
```