Introduction

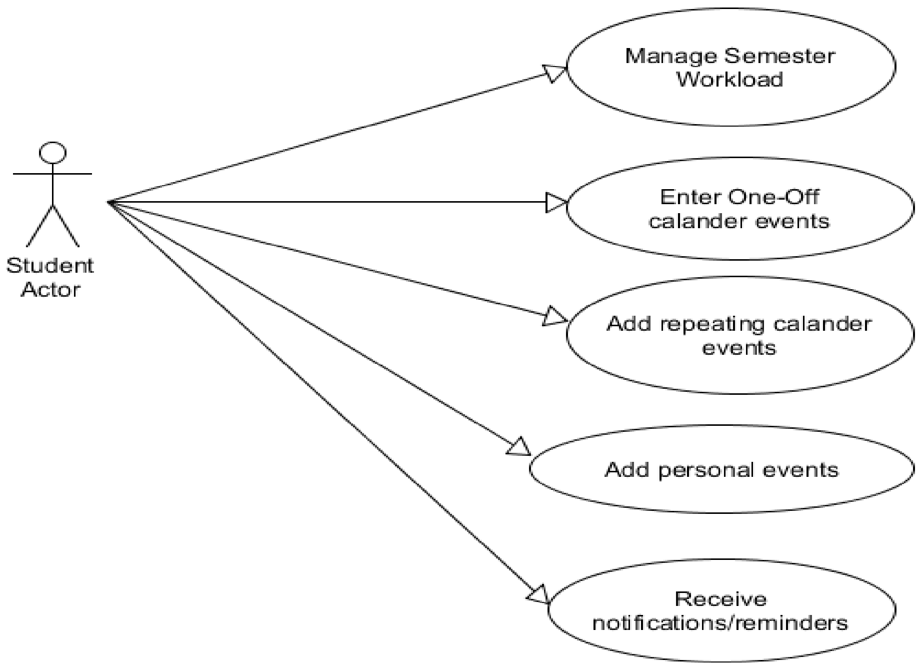
This document is an initial design analysis of a Digital Diary software for use within a college or school. In this, any possible users and use cases will be examined for implementation in the final system. This will involve brainstorming possible uses for the system as well. To do this, a UML diagram will be composed with users, interactions and use cases. From this, simplified class diagrams can be designed. Once this has been done, a proper understanding of what the problem statement expects of us will be clearer and thus a better system will be implemented in the end. This process enables the designer to understand exactly what is being expected so as to not spend time or money on implementing counterproductive software or aspects of the system.

Problem Statement

You are required to develop a C++ based academic digital diary application to aid students manage their semester workload in ITB. The digital diary should allow users to enter one-off events or repeating events. An event can be a timetabled class or assignment etc. The application should provide reminders as well as possible clashes with events. The digital diary should allow lecturer users to add common events to all diaries in the system as well as student users to add personal events. Real-time notifications of class changes, assignment extensions and individualized notes attached to events should also be possible.

Use cases

Use Case - Model 1


Manage Semester Workload
Enter One-Off calander events
Add repeating calander events
Add personal events
Receive notifications/reminders
Student Actor

From analysing of the use cases, it could be seen that a student actor's main goals of using the system would be to manage their workload, enter one off, personal and repeating calendar events and receive notifications from the diary once the time for these events have come.

Fully Dressed Use Case - Model 1

Primary actor: Student user

Goal in context: User wishes to manage workload, enter one off and repeating and personal calendar events and receive notifications from the software.

Level: User Level

Stakeholders and Interests:

User: Wants to keep up to date with college events and workload

Preconditions: System is correctly keeping track of time and date and awaiting inputs from the user

Minimum Guarantee: Failure message stating event could not be created.

Success Guarantees: User created event is stored within the calendar and a timer has been started for notification upon completion

Trigger: User starts interaction by opening software and logging in.

Main Success Scenario:

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to create an event
4. System prepares itself for event creation
5. User adds time, date etc. To the event creation
6. System notifies user of event creation
7. System adds event to calendar and begins timer countdown

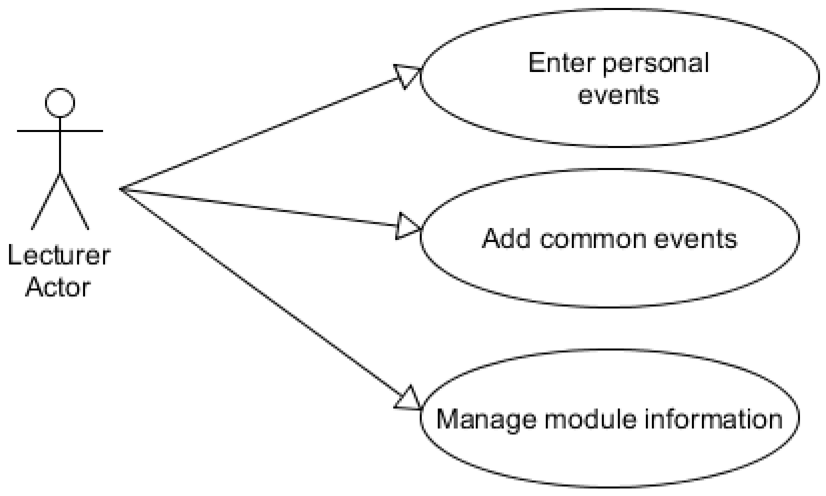
Extensions:

1. User enters incorrect login details. Proceeding is held until correct info is entered
2. User creates and clashing event with event already in the system

Frequency of Use: Several times per day

Priority: 1

Use Case - Model 2



From analysing of the use cases, it could be seen that a student actors main goals of using the system would be to manage the module information, enter one off, personal and repeating calendar events and receive notifications from the diary once the time for these events have come. Another main goal of the lecturer is to add common events for all students in a given module that they are delivering.

Fully Dressed Use Case - Model 2

Primary actor: Lecturer user

Goal in context: User wishes to enter one off and repeating and personal calendar and public events, module information and receive notifications from the software.

Level: User Level

Stakeholders and Interests:

User: Wants to manage module info for students, add public events for students of a certain module and add personal events

Preconditions: System is working sufficiently and is awaiting input to display data

Minimum Guarantee: Failure message stating event could not be created or module info could not be added.

Success Guarantees: User created event is stored within the calendar and a timer has been started for notification upon completion

Trigger: User starts interaction by opening software and logging in.

Main Success Scenario 1:

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to create an event

4. System prepares itself for event creation
5. User adds time, date etc. To the event creation
6. System notifies user of event creation
7. System adds event to calendar and begins timer countdown

Main Success Scenario 2:

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish manage module information
4. System prepares itself
5. User adds info to the module of their choice
6. System notifies user successful changes made

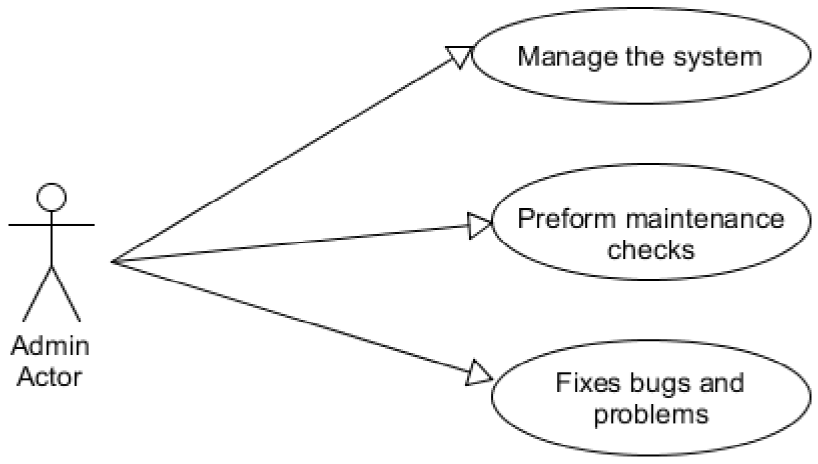
Extensions:

1. User enters incorrect login details. Proceeding is held until correct info is entered
2. User creates and clashing event with event already in the system
3. User tries to change module info that they do not have access to

Frequency of Use: Several times per day

Priority: 1

Use Case - Model 3



From analysing of the use cases, it could be seen that a admin actors main goals of using the system would be to manage the systems integrity for the students and lecturers using it. To do this, he my have to perform maintenance checks and tests on the system. For this, the system may have to be brought offline while the tests are being implemented.

Fully Dressed Use Case - Model 3

Primary actor: Admin user

Goal in context: User wishes to maintain the software.

Level: User Level

Stakeholders and Interests:

User: Wants to maintain the software for reliability for use by students and lecturers throughout the semester

Preconditions: System is waiting for admin inputs

Minimum Guarantee: Failure message stating that system is not functioning properly and needs maintenance

Success Guarantees: Admin preforms tests on the system that return successful

Trigger: User starts interaction by opening software and logging in.

Main Success Scenario 1:

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to perform maintenance tests on the software
4. System prepares itself for maintenance and removes itself from server until finished
5. User preforms tests
6. System notifies user of test results

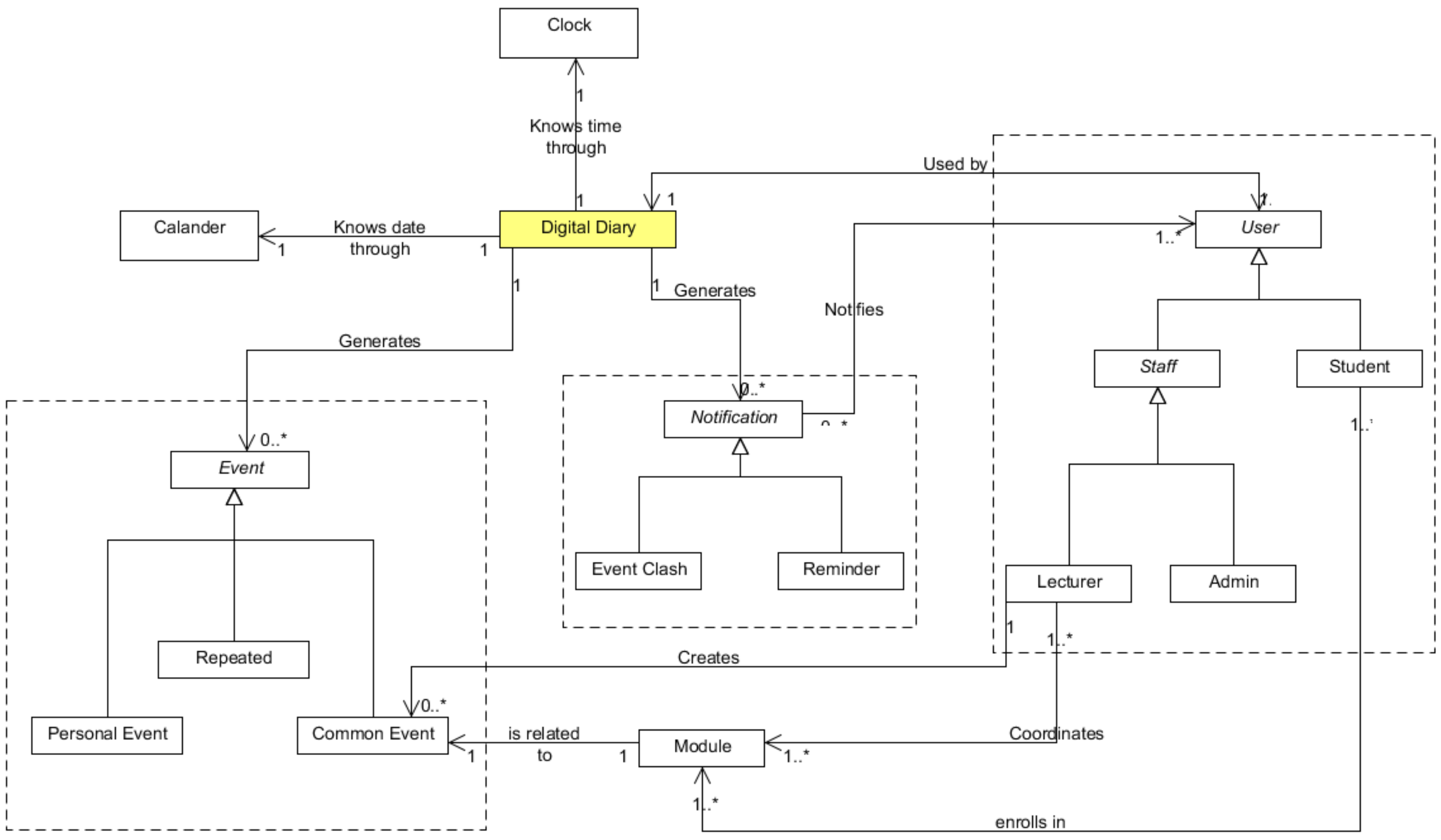
Extensions:

1. User enters incorrect login details. Proceeding is held until correct info is entered
2. Tests return a fail and user will need to perform maintenance on the ststem

Frequency of Use: Several times per day

Priority: 1

Class Diagram

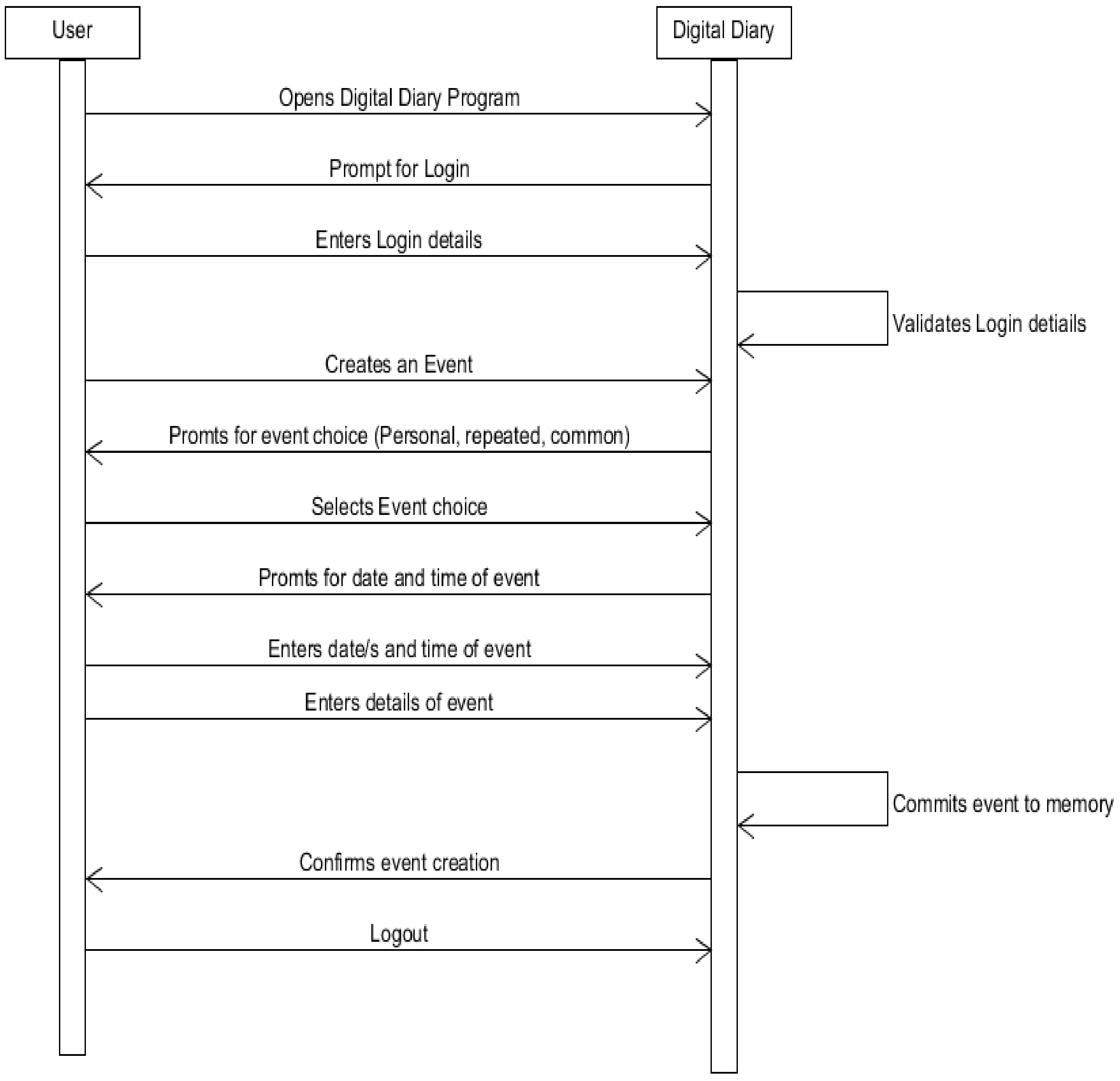


In the diagram above, it can be seen that the Digital Diary system has some main classes associated with it. These classes include *User, Notification* and *Event*. The Open Close and Liskov Substitution Principles have been taken into consideration at this early stage class diagram. Examples of this would be as *User* is an abstract class with derived classes *Student, Lecturer* and *Admin* stemming from it. Similarly with *Notification,* Event Clash and Reminder are two possible instances of notification within the system. There are three possible instances of event that have been teased out of the design so far which include *Personal, repeated and Common*. As Lecturer will be the only user with the ability to create a common event, there will need to be some way of representing this. For this reason, a line has been connected between the two. A common event will be related to 1 module coordinated by a single lecturer, so this has been represented also.

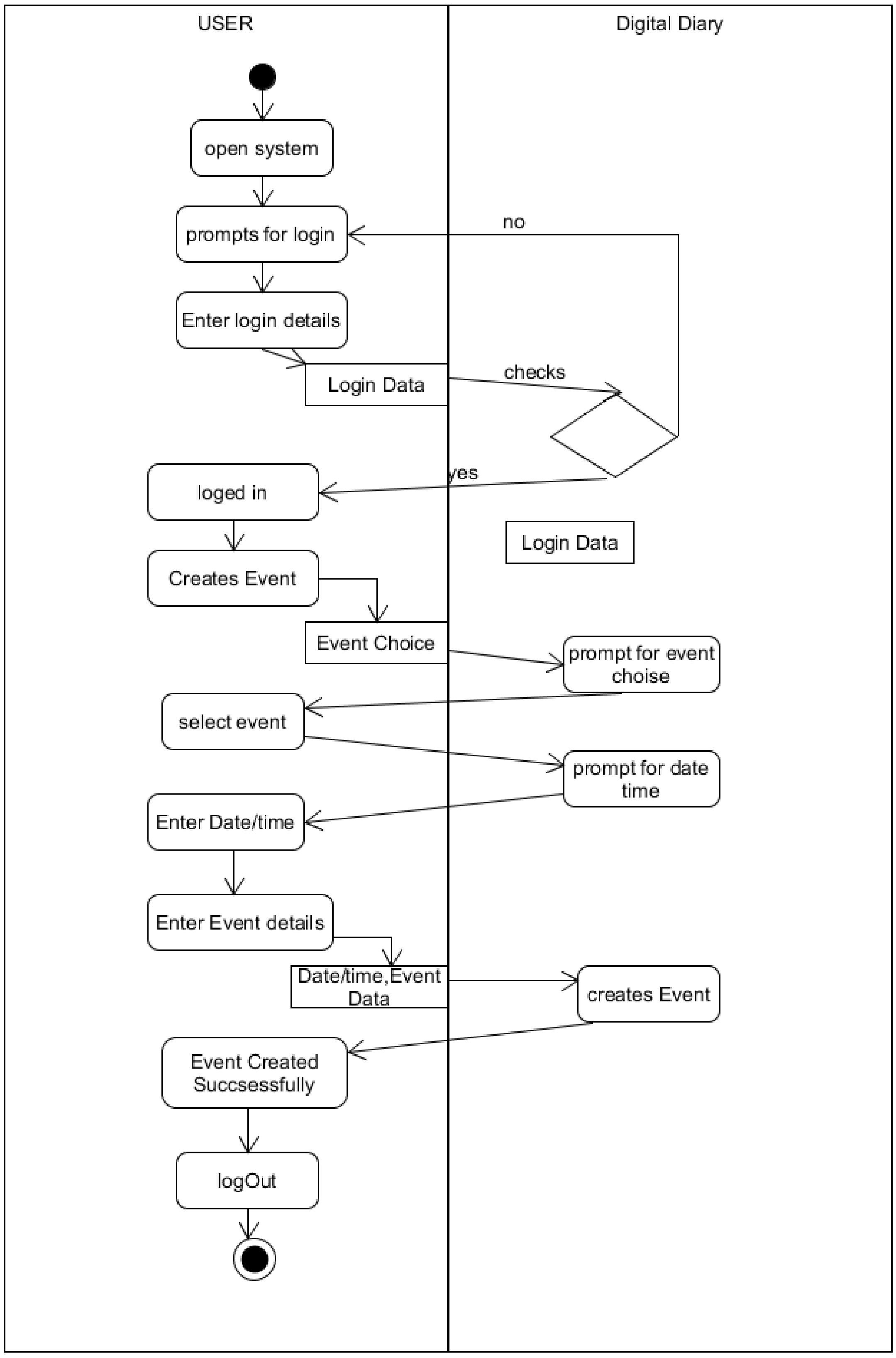
The system also has a *Calendar* which will act as a memory of all the event stored within the system. When an event has been created, a timer until that specific time and date will be started through the *Clock.*

Behavioural Analysis

High Level Sequence Diagram

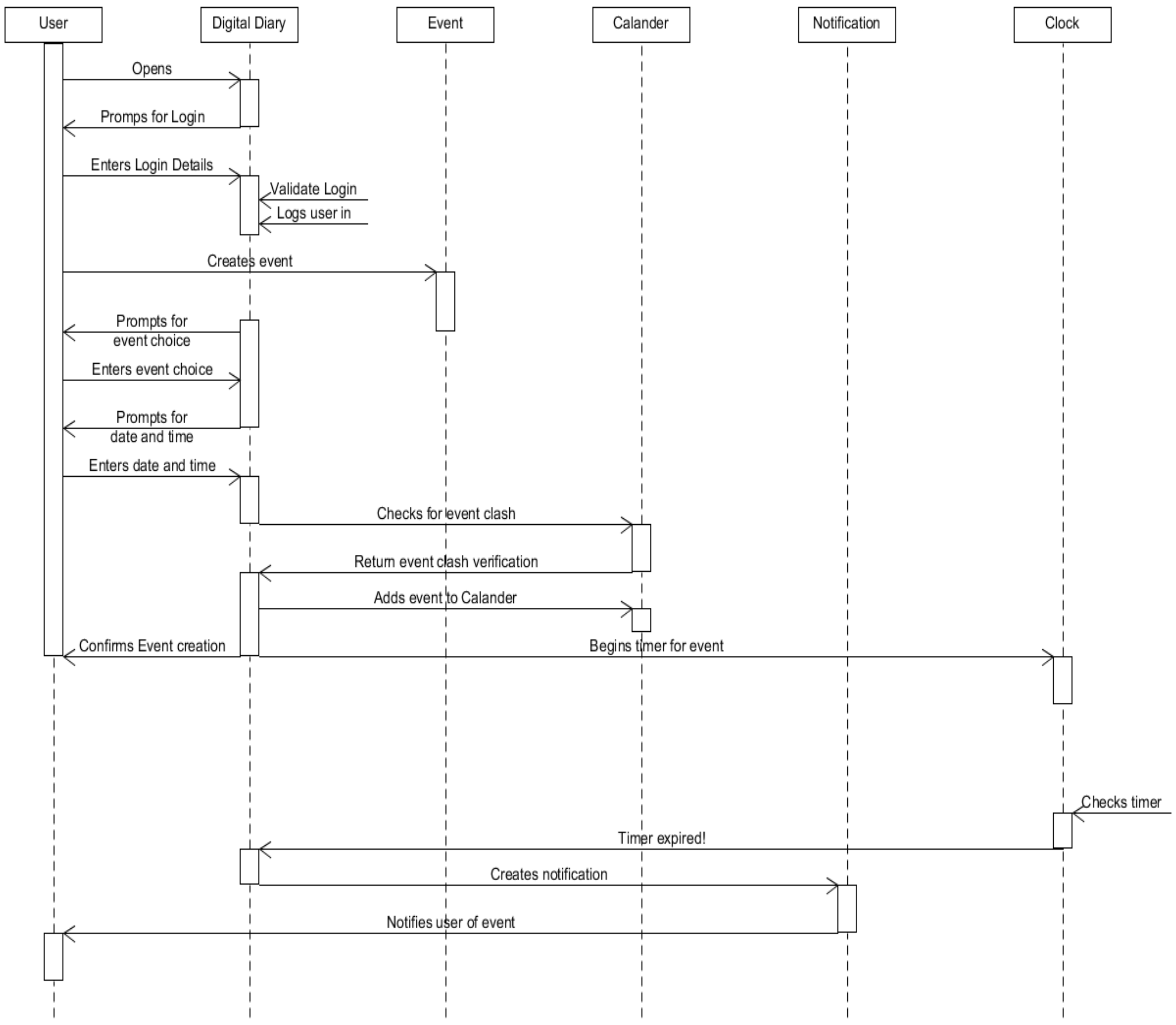


High Level Activity Diagram

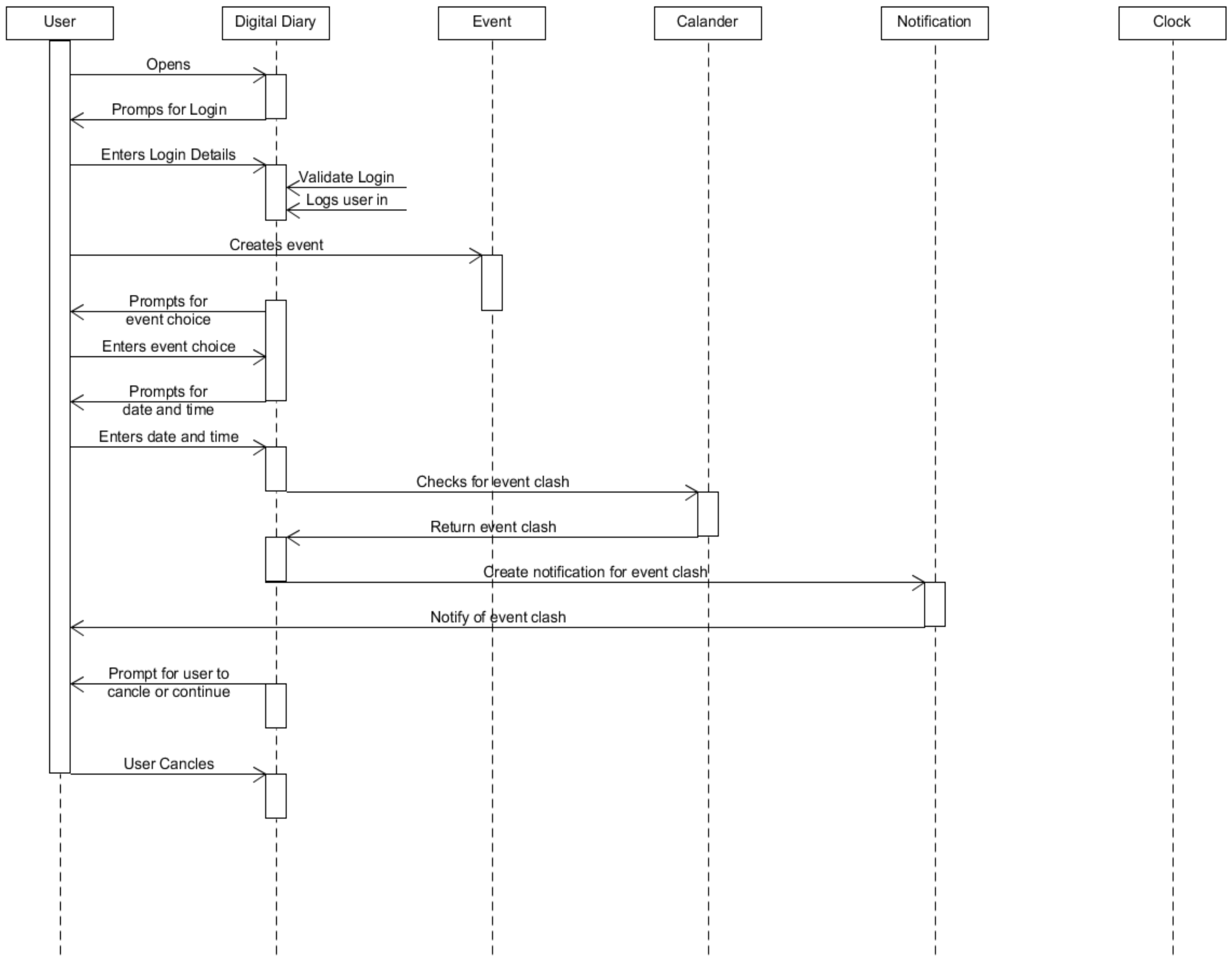


Software Design

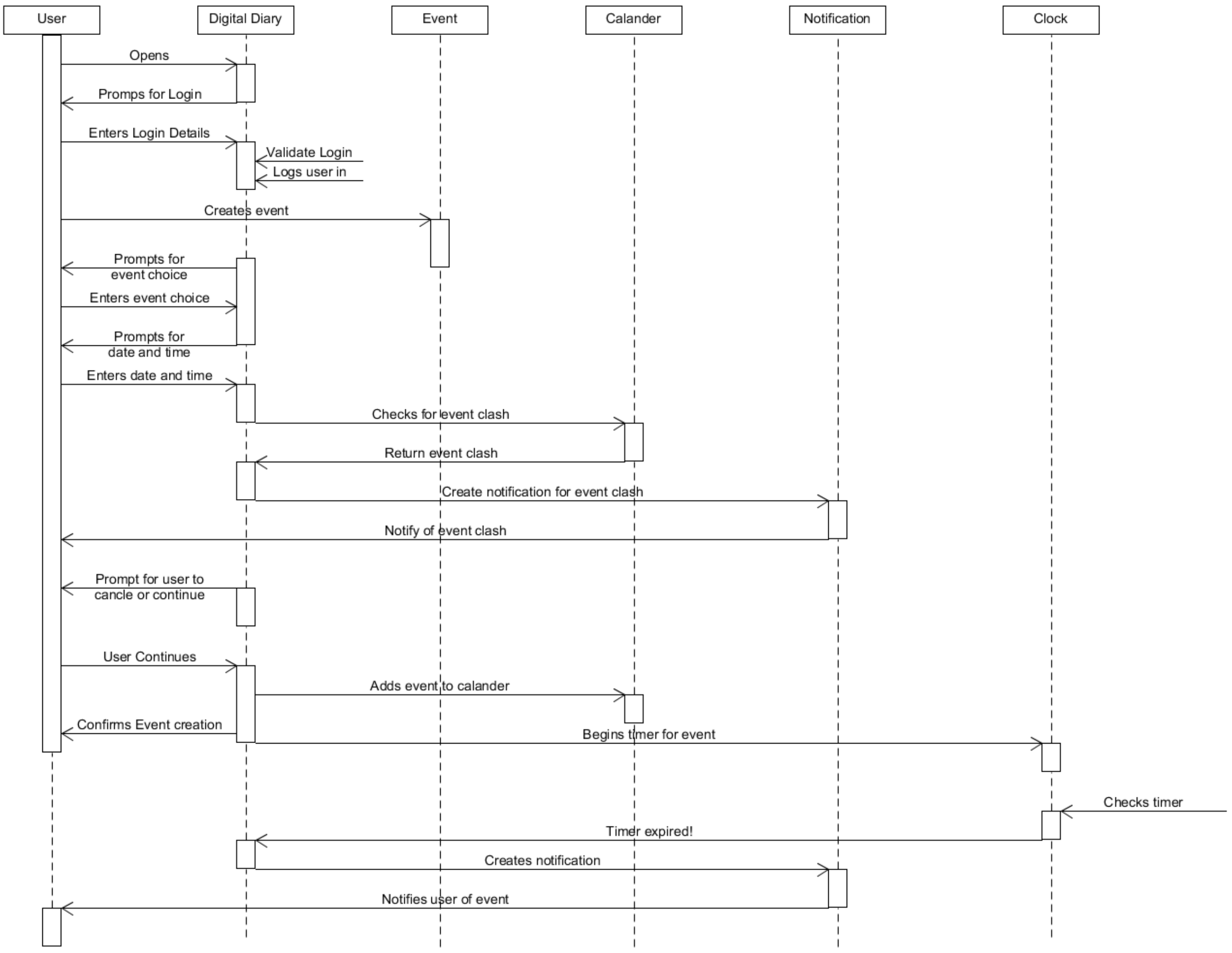
Low Level Sequence Diagram 1 – Event Creation (No Clash)



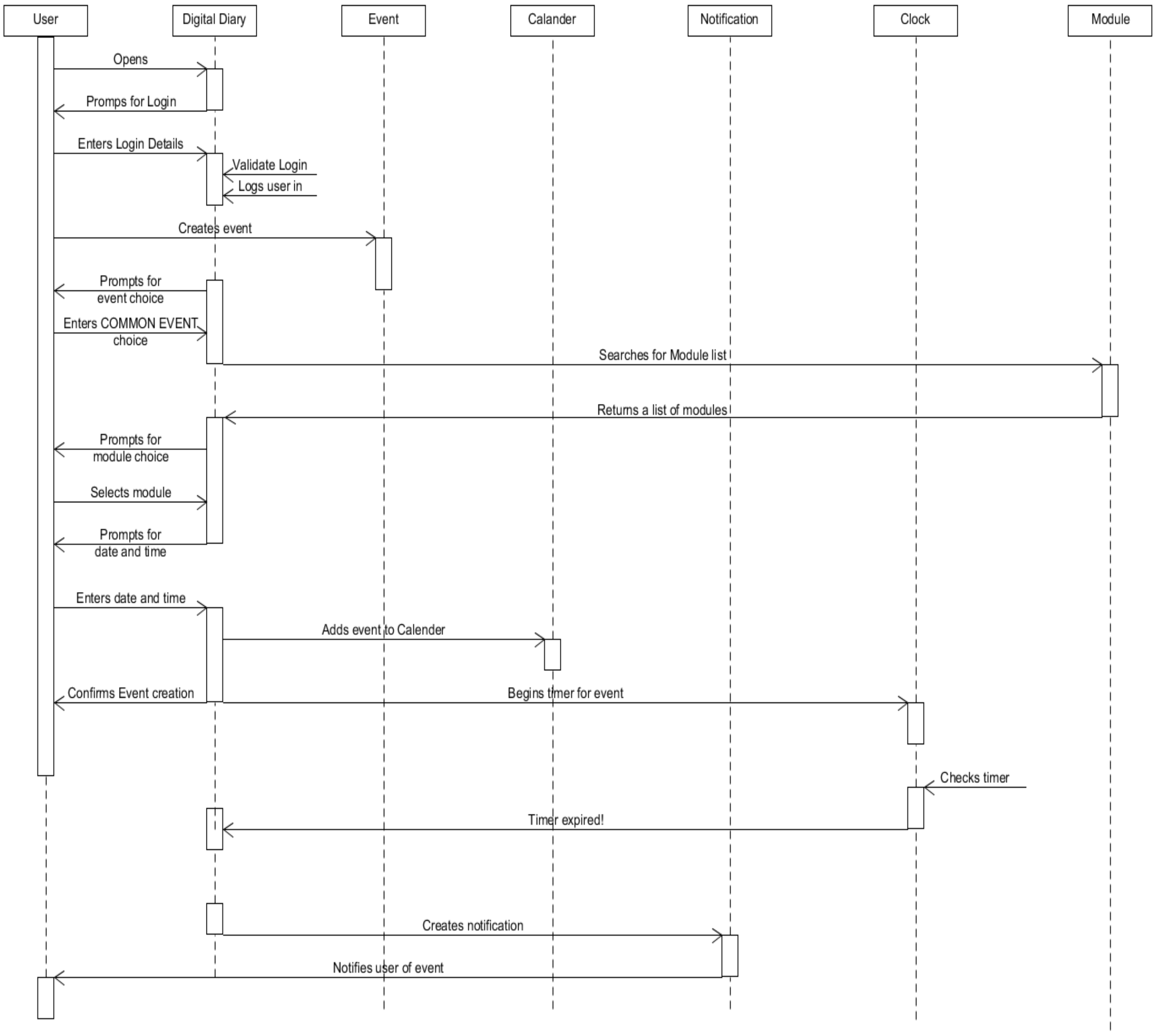
Low Level Sequence Diagram 2 – Event Creation (With Clash – User Cancel)



Low Level Sequence Diagram 3 – Event Creation (With Clash – User Continue)



Low Level Sequence Diagram 4 – Event Creation (Lecturer – Common Event)


User
Digital Diary
Event
Calander
Notification
Clock
Module
Opens
Promps for Login
Enters Login Details
Validate Login
Logs user in
Creates event
Prompts for event choice
Enters COMMON EVENT choice
Searches for Module list
Returns a list of modules
Prompts for module choice
Selects module
Prompts for date and time
Enters date and time
Adds event to Calender
Confirms Event creation
Begins timer for event
Checks timer
Timer expired!
Creates notification
Notifies user of event