Lab 2 – Object Oriented Conceptual Modelling

**Student Name: simon**

**Last Name: Convery**

**Student Number: B00041726**

***Department of Engineering***

***School of Informatics & Engineering***

***Institute of Technology, Blanchardstown***

***Dublin 15.***

**Module Title:** Software Design & Quality

**Submission date:** 17 December 2017

# **Table of content**

[Table of content 2](#_Toc496192879)

[1.Introduction 3](#_Toc496192880)

[2. Problem Statement 3](#_Toc496192881)

[3. Use Case Analysis 3](#_Toc496192882)

[Fig.3.1 4](#_Toc496192883)

[3.1 Use Case Model 1 4](#_Toc496192884)

[3.2 Use Case Model 2 4](#_Toc496192885)

[3.3 Use Case Model 3 5](#_Toc496192886)

[4. Conceptual Model 5](#_Toc496192887)

[Fig.4.1 6](#_Toc496192888)

[5. Modelling Assumptions 6](#_Toc496192889)

[6. System Design Observations and Recommendations 6](#_Toc496192890)

[7. Fully dressed use case 7](#_Toc496192891)

[7.1 Use case 1: turn on display 7](#_Toc496192892)

[7.2 Use case 2: change display 7](#_Toc496192893)

[7.3 Use case 2: admin log on 8](#_Toc496192894)

[8. Sequence Diagrams 9](#_Toc496192895)

[8.1 Sequence Diagram case 1 9](#_Toc496192896)

[8.2 Sequence Diagram case 2 10](#_Toc496192897)

[8.3 Sequence Diagram 3 11](#_Toc496192898)

1. Introduction

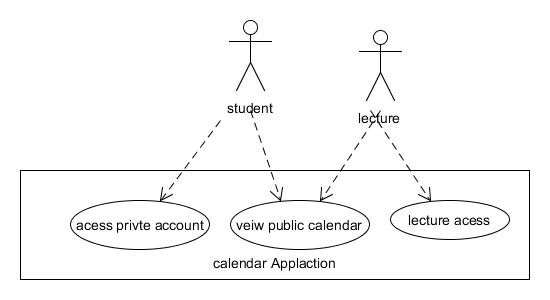
For this aissment I will practice creating visual models of a problem domain to better understand the problem using use cases and Conceptual Modelling.

1. Problem Statement:

This assessment is to design and developed an academic digital diary with the following functions:

* Allow you to create one off and repeating events.
* Give reminders.
* Give Warnings of clashing events.
* Lectures can set events to their students or school wide.
* Add personal events.
* Real time notifications of class changes, assignment extensions and individualized notes attached to events should also be possible.

# 3. Use Case Analysis



## Figure.3.1

## 3.1 Use Case Model 1

Use case = view calendrer

Description= view public calendrer

Pre-Conditions= student can access the public calendar

Post-Conditions= student views the public calendar

Main flow= Access calendar –view calendar

## 3.2 Use Case Model 2

Use case = access private account

Description= access your private account

Pre-Conditions= student try’s to access their account

Post-Conditions= student is given access to their account

Main flow=access private account –enter password-logged into your account

Alternate Flow= if wrong password is entered access denied

## 3.3 Use Case Model 3

Use case=lecture access

Description= lecture access lecture account

Pre-Conditions= lecture has required access to his lecture account

Post-Conditions= lecture is signed in and accessing his account

Main flow=access lecture account-enter lecture password-signed in to account

Alternate Flow= if wrong password is entered access denied

# 4. Conceptual Model

# 5. Modelling Assumptions

* That each student has their own account
* That there would be a public calendar
* That only the student has access to their privet calendar

# 6. System Design Observations and Recommendations

* I gave everyone their account.
* Lectures accounts are different as they are the only ones that can set class events.
* I have reminders set as a sub-class of events because you can set an event as a reminder.
* Class events are a sub-class of lecture events s the user can decide what events are class events to share.
* I have public events separated but maybe they should be one class accessible by student and lecture.
* The reminders are only attached to the calendar and private events it needs access to class events to give a reminder of conflicting events.
* Reminder also need access to public events encase of conflicting events.

# 7. Fully dressed use case

## 7.1 Use case 1: access private account

Primary Actor: student

Goal: access private event

Stake holders and investors:

* Customers: wants access to account.
* Investor: wants customer to access account.

Main success scenario:

* Access calendar App
* Access private account
* Enter password
* Password correct account accessed

Extensions:

* Wrong password access denied.
* Something wrong with App no access.

Frequency of use: several times per day

Priority: 1

## 7.2 Use case 2: view private calendar

Primary Actor: student

Goal: view private calendar

Stake holders and investors:

* Student: wants access to view private calendar.
* Investor: wants student to view private calendar.

Main success scenario:

* Access calendar App
* Access private account
* Enter password
* Password correct account accessed
* Access private calendar

Extensions:

* Wrong password access denied.
* App having problems accessing account in no access.
* App having problems with private calendar can’t see private calendar.

Frequency of use: several times per day.

Priority: 2

## 7.3 Use case 3: access lecture account

Primary Actor: lecture

Goal: access lecture account

Stake holders and investors:

* Lecture: wants to access their account.
* Investor: wants lecture to get access.

Main success scenario:

* Access calendar App
* Access lecture account
* Enter password
* Password correct account accessed

Extensions:

* Wrong password access denied.
* App having problems accessing account in no access.

Frequency of use: most days.

Priority: 2

## 7.4 Use case 4: view lecture calendar

Primary Actor: lecture

Goal: view lecture calendar

Stake holders and investors:

* Lecture: wants access to view private calendar.
* Investor: wants lecture to view private calendar.

Main success scenario:

* Access calendar App
* Access lecture account
* Enter password
* Password correct account accessed
* Access lecture calendar

Extensions:

* Wrong password access denied.
* App having problems accessing account in no access.
* App having problems with lecture calendar can’t see private calendar.

Frequency of use: most days.

Priority: 3

## 7.5 Use case 5: view public calendar

Primary Actor: student

Goal: view public calendar

Stake holders and investors:

* Student: wants access to view private calendar.
* Investor: wants student to view private calendar.

Main success scenario:

* Access calendar App
* Access public calendar
* view public calendar

Extensions:

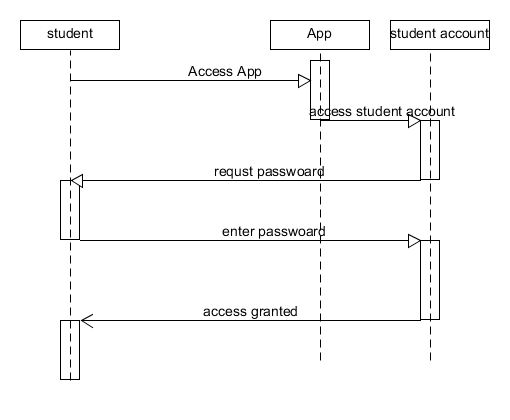
* App having problems no access.
* App having problems with public calendar can’t see public calendar.

Frequency of use: every couple of days.

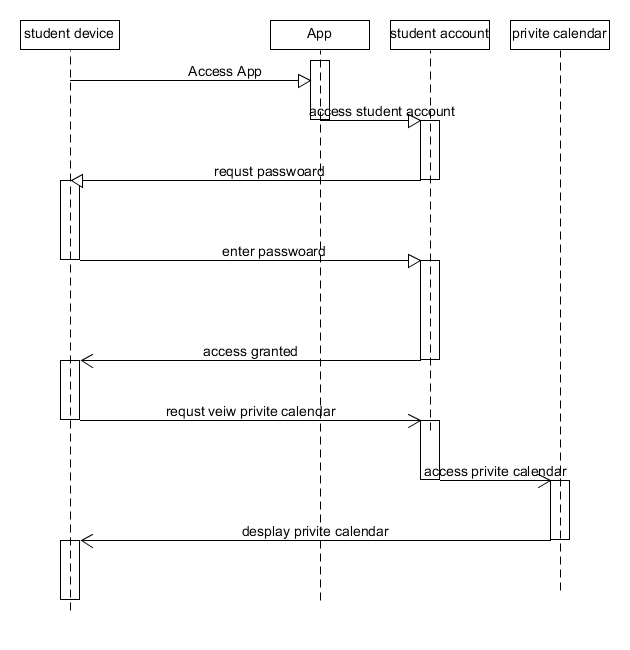
Priority: 4

# 8. Sequence Diagrams

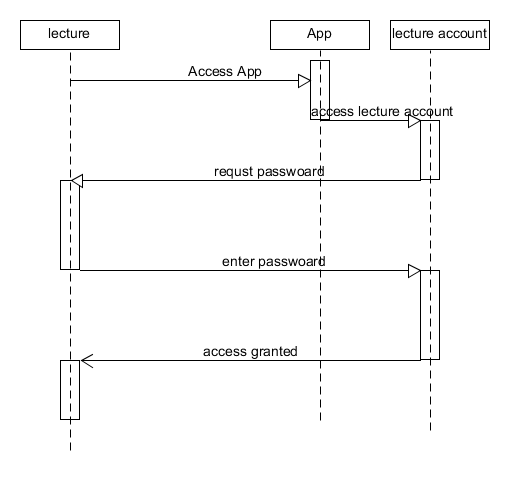
## 8.1 Sequence Diagram case 1



## 8.2 Sequence Diagram case 2



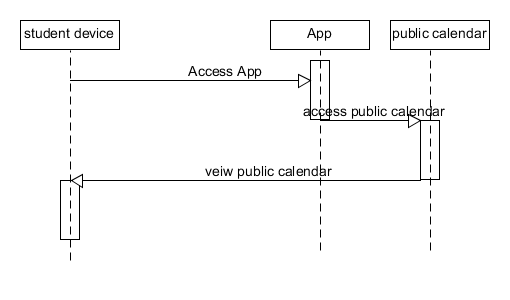
## 8.3 Sequence Diagram 3



## 8.3 Sequence Diagram 4

# 

## 8.3 Sequence Diagram 5



# 9. Refined Conceptual Model

# 10. Code frame work

#include <vector>

#include <iostream>

using namespace std;

class studentAccount{

public:

const string password;

int choies;

void enterpassword()

{

// enter password and checked hear

}

void Desplaychoies()

{

//desplay options menu for account

}

};

class LectureAccount{

public:

const string password;

int choies;

void enterpassword()

{

// enter password and checked hear

}

void Desplaychoies()

{

//desplay options menu for account

}

};

class StudentsEvents {

private:

string EventType;

string Massage;

float Time;

float Date;

public:

float EnterEvent()

{

//Enter your event hear

}

};

class PublicEvents {

private:

string Massage;

float Time;

float Date;

public:

float EnterEvent()

{

//Enter your event hear

}

};

class LectureEvents {

private:

string EventType;

string Massage;

float Time;

float Date;

public:

float EnterEvent()

{

//Enter your event hear

}

};

class ClassWideEvents {

private:

string ClassWideEvent;

string Massage;

float Time;

float Date;

public:

float Setreminder()

{

//set a reminder in private student calendar

}

};

class Reminder {

public:

float time;

float date;

string massage;

int SortEvent(){

//sort events by time and date

};

float EventConfiict{

//detect event conflicts on calendar

};

};

class StudentCalendar{

public :

void DesplayCalendar()

{

//Desplay blank calendar

}

void SortStudentEvents()

{

//sort student events into desplayed calendar

}

void SortReminders()

{

//sort reminders in to desplaed calendar

}

void SortClssWideEvents()

{

//sort class wide events in to calendar

}

};

class LectureCalendar{

public :

void DesplayCalendar()

{

//Desplay blank calendar

}

void SortStudentEvents()

{

//sort student events into desplayed calendar

}

void SortReminders()

{

//sort reminders in to desplaed calendar

}

void SortClssWideEvents()

{

//sort class wide events in to calendar

}

};

class Calendar {

public:

float time;

float date;

string massage;

};

int main()

{

} 11. Requirement

|  |  |
| --- | --- |
| Requirement ID | SRS-C++-001 |
| TITLE | Access App |
| DESCRIPTION | Use a device to access the app |
| VERSION | 1.0 |

|  |  |
| --- | --- |
| Requirement ID | SCR-C++-002 |
| TITLE | Access private account |
| DESCRIPTION | Student will have entered there password and gained access to their accounts |
| VERSION | 1.0 |

|  |  |
| --- | --- |
| Requirement ID | SCR-C++-003 |
| TITLE | Access private calendar |
| DESCRIPTION | Allow student to access their provide calendar |
| VERSION | 1.0 |

|  |  |
| --- | --- |
| Requirement ID | SCR-C++-004 |
| TITLE | Access lecture account |
| DESCRIPTION | Lecture enters password and gains access to his account |
| VERSION | 1.0 |

|  |  |
| --- | --- |
| Requirement ID | SCR-C++-005 |
| TITLE | Lecture access lecture calendar |
| DESCRIPTION | After accessing their account the lecture access’s their lectures calendar |
| VERSION | 1.0 |

|  |  |
| --- | --- |
| Requirement ID | SCR-C++-006 |
| TITLE | Student logging out of their account |
| DESCRIPTION | Student exits their account |
| VERSION | 1.0 |