1.0.0

|  |
| --- |
| **Vipassanā ‐ Insight Awareness (VIA) Event Management System** |

**Version history**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Version** | **Init** | **Revised** | **Approved** | **Date** |
| Start of document | 0.1.0 | IHA | <INIT> | <INIT> | 2017-01-31 |
| Small corrections | 0.2.0 | IHA | MWA |  | 2017-02-01 |
| Revisions made in workshop meeting | 1.0.0 | MWA/IHA | All | ICT | 2017-08-19 |

**Group: XXXX**

|  |  |
| --- | --- |
| **Student Name** | **Student No** |
| Dragos Chirtoaca | 253742 |
| Pascari Liviu | 266094 |
| Oskars Arajs | 266534 |
| Taha Mohamed Alzein | 269055 |

# Contents

[1 Introduction 2](#_Toc490902148)

[2 Requirements 2](#_Toc490902149)

[2.1 Functional Requirements 2](#_Toc490902150)

[2.2 Non-Functional Requirements 2](#_Toc490902151)

[3 Analysis 2](#_Toc490902152)

[4 Design 2](#_Toc490902153)

[5 Implementation 2](#_Toc490902154)

[6 Test 2](#_Toc490902155)

[6.1 Test Specifications 2](#_Toc490902156)

[7 Results and Discussion 2](#_Toc490902157)

[8 Conclusion 2](#_Toc490902158)

[9 Project future 2](#_Toc490902159)

[10 References 2](#_Toc490902160)

[11 List of Appendixes 2](#_Toc490902161)

[Appendix A Project Description 2](#_Toc490902162)

**Abstract**

*The main objective of the project is to develop an event handler(event management) system for Vipassanā ‐ Insight Awareness (VIA), in order to help the company to solve their current problems related to management of the events and their members. This system should be capable to make the work easier for the event manager of their company. For the event manager it is important to see the status of their events, is this an event in the past or in the future, to create an event, to edit, or to delete. On the other hand the event manager need to be able to manage the participants of the events such as members and non-members. For the members event manager should be able to add a member, to see the status of members membership, and to be able to send a reminder outside of the system to this members that has to pay, so for this it is necessary to identify other contact information of this members. It is very important to handle a list of contacts, is it a member or non-member, in order to send manually newsletter about current events based on the preferences of the event participants.*

*In order to satisfy the customer needs and to develop an end use product, it is needed to divide the project into objectives that should be reached, that are:*

* *based on interview we identified the main requirements of the customer;*
* *design the diagrams that are describing the main problems that the system should solve;*
* *create the interface of the system that will make the work of event manager easier;*
* *develop the system functionality using Java;*

*All the information that is related to the events, members, non-members, sponsors and lecturers, is stored using binary files, on the computer on which is running.*

*As a result, the project has an event handling system related to members, non-members, sponsors and lecturers management, which meets-up the customer’s requirements. This system is easy to modify and add to it more features if this will appear in customer’s needs.*

# Introduction

For any organisation that is organising events it is very hard to manage the events, especially if there are many and especially if you are non-profit organisation. Nowadays people are very opened in studying new things, and as a result of increasing the number of customers that are interested in Buddhism, meditation, alternative self healing, there is very hard to keep track of all the events. For keeping the track of events, members, lecturers, sponsors, participants needs there is a need for a very good event manager, and there is very hard if this event manager decides to leave the company, but by using a system that is holding all this information in digital format and is making creation of an event with almost just a click, it will save not just the time, but it will make the company more secure and safe in the event handling.

In the traditional way of handling events, there were a pencil and a piece of paper (notebook) and the event manager has to search through all of that paper in order to find the information of any specific event. In our fast developing word and in digitalized era this is possible by writing everything in excel, access or other software that are helping a little bit the user. This is still not enough because this does not fully satisfy our needs and the users have to learn how to use this software that are doing too many things that we don’t need. That’s why as a professionals companies need to create something more user-friendly and more suitable for their needs.

VIA is a centre for spiritual events originally with a base in the Buddhist principles of meditation as an insight with awareness of what happens when it happens. Today events at VIA also include spiritual practices not directly related to any religion like dream interpretations, healing, astrology, reincarnation, karma, alternative health care and similar events. VIA helps organize these events and journeys for people with interest in different categories starting from exploring one’s mind and learning how to deal with inner conflicts. Up until now, they used pen and paper to keep track of their lectures, lecturers and upcoming events. But that system is only viable if the number of lectures and events are low. Otherwise, the system gets overwhelmed and it gets hard to keep track of all of the data and needed resources to keep the organization going. That’s why they decided to implement a software in their company that will make their work easier and it will help them to not disappoint the current customers.

The developed event handling software system allows the event manager to: create, edit, delete an event, by entering the category, type, name, date, time, duration, lecturer and price, on the other hand it will help to keep track of events, is this a future event, or this is an event in the past. The event manager will be able to add, edit, delete a member, by entering name, address, phone number, email and preferences of the member, on the other hand the event manager will be able to see the members and non-members in a specific category, and to see which members has to pay their membership by choosing the interested date. Every time when the event manager is adding a member or a non-member the system is generating automatically an ID for each of them, that is starting with “M” if it is a member and with “N” if it is non-member. The event manager will be able to add needed member or non-member to the needed event only by finding the specific member or non-member by their ID, or to create new ones if they are not in their lists. The system is not able to calculate the total income of an event and it is not able to send automatically newsletters to the potential participants.

The system purpose is to help non-profit organization with their hard lifetime in managing events, and on the other hand to manage their sponsors, lecturers, and of course members.

**Remember:** You must ensure a clear connection between sections in the project report, from Project Description, Requirements, Analysis, Design, Implementation to Test. This means that everything that is implemented can be found in design, everything that is designed is based on the analysis, and anything that is found in analysis has a clear link to requirements, etc.

# Requirements

During the interview with the representatives of VIA center there are identified some Functional and Non-functional requirements. This requirements serve as boundaries and checkpoints for the future implemented system.

## Functional Requirements

Functional requirements represent the major functionality of the software that is expected from the customer, and there are the following:

1. The system must allow user to add a member.
2. The system must allow user to edit a member.
3. The system must allow user to delete a member.
4. The system must allow user to add category.
5. The system must allow user to delete category.
6. The system must allow user to add an event.
7. The system must allow user to edit an event.
8. The system must allow user to delete an event.
9. The system must allow user to add a sponsor.
10. The system must allow user to edit a sponsor.
11. The system must allow user to delete a sponsor.
12. The system must allow user to add a lecturer.
13. The system must allow user to edit a lecturer.
14. The system must allow user to delete a lecturer.
15. The system must allow user to add a none-member.
16. The system must allow user to add participant (member/non-member) to event.
17. The system must allow user to delete participant (member/non-member) from event.
18. The system must allow user to add lecturer to event.
19. The system must allow user to delete lecturer from event.
20. The system must allow user to search for finalized events.
21. The system must allow user to search for and non-finalized events.
22. The system must allow user to search for sponsor in specific category.
23. The system must allow user to search for lecturers in specific category.
24. The system must allow user to search for members’ preference those interested in specific category.
25. The system must allow user to find the members those not paid their membership.

## Non-Functional Requirements

The non-functional requirements represent the requirements that are not related to the main functionality and does not affect the usage of the system, and this are:

1. The system must use files as data storage
2. The system must be implemented in Java
3. The system needs to answer within 2 seconds 95% of the time
4. The system must be usability tested by end users

# Analysis

During the analysis phase, there is a need of understanding the main requirements of the customer, and in order to understand their needs we have to implement the following steps:

* + - * use case model;
      * description of the use case;
      * activity diagrams;
      * class diagram;

## Use case diagram

The following diagram shows what the event manager must be able to do, in order to satisfy the client needs based on the interview. Each use case that you can see in the diagram is an action that can be performed by the event manager.

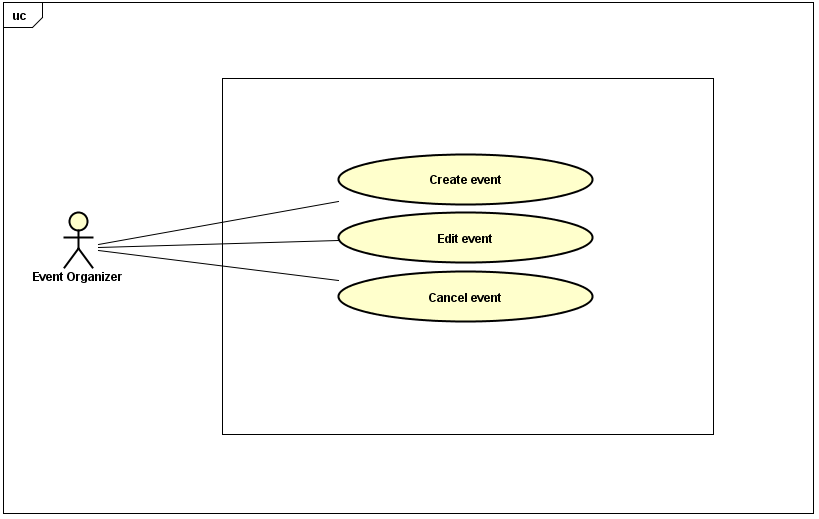


Figure 1, Use case diagram

In the figure above, we have a short version of the use case diagram, which is showing what the user will be able to do by using the developed system.

See the full version of the use case diagram in the Appendix B

## Use case diagram description

A Use Case diagram contains the following information:

* + - * ID, for system design.
      * Title to describe the action taken in the use case.
      * Description of the title in more detail.
      * Primary actor shows who is going to use the system.
      * Preconditions show what is needed before the action is initiated.
      * Post conditions shows what is the end result of the action performed.
      * Main scenario shows the steps taken to achieve the end result from the initiation of the action.
      * Extension shows what external interference may occur.

Table 1, Add an event USE CASE

|  |  |
| --- | --- |
| ID: | #7 |
| Title: | Add an event |
| Description: | The user will add an event |
| Primary Actor: | User |
| Preconditions: | At least, one lecturer exists, and one event activity subject exist. |
| Postconditions: | An event is added. |
| Main  Success Scenario: | 1. User chooses to add a new event. 2. User select activity subject for event 3. User enter new event information (type, title, date, time, duration, lecturer/lecturers, price, discounted price) 4. User validates 5. System store new event information in data file storage |
| Extensions: | 4., System not able to reach a data file storage shows an error message,  2., If activity subject not found, go to #4 step2  3., if lecturer not found, go to #13 step2 |

See the full list of Use Case description in Appendix C

This Use Case diagram represents the adding of an event.

The user has to choose the subject for the event. If the subject entered is not found then the user has to go to case #4 “Add an activity subject” and add the required subject to the list so that in the future the user can add this subject to the specific event.

Next possible error that might occur is when adding a lecturer. If the Lecturer entered is not found then the user has to go to case #13 “Add a lecturer” and add the required lecturer to the systems

All information, except for “Subject” and “Lecturer” are unique for all events. Meaning that the system does not keep list of possible times or possible prices for events.

## Activity diagram

An activity diagram represent visualization of the actions that are needed in order to implement something in the usage of a system.

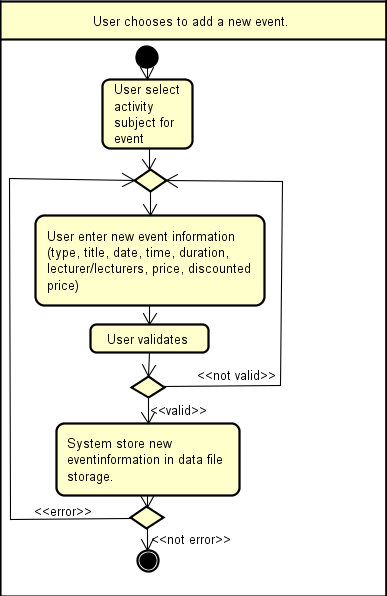


Figure 2, Sequence diagram

The figure above represents an activity diagram showing the steps and actions taken to reach the final node (create an event).

See full list of activity diagrams in the Appendix D;

## Class diagram

A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations. Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.[[1]](#footnote-2) UML Class diagrams help to understand the structure of a system by representing different classes and their relationship with other classes and objects as well as methods, instance fields and the structure of the system.

The above picture is representing a UML diagram of the class “Event”. The importance of this class is that it is responsible for creating events. Other methods that are implemented in this class are adding and deleting participants and lecturers.

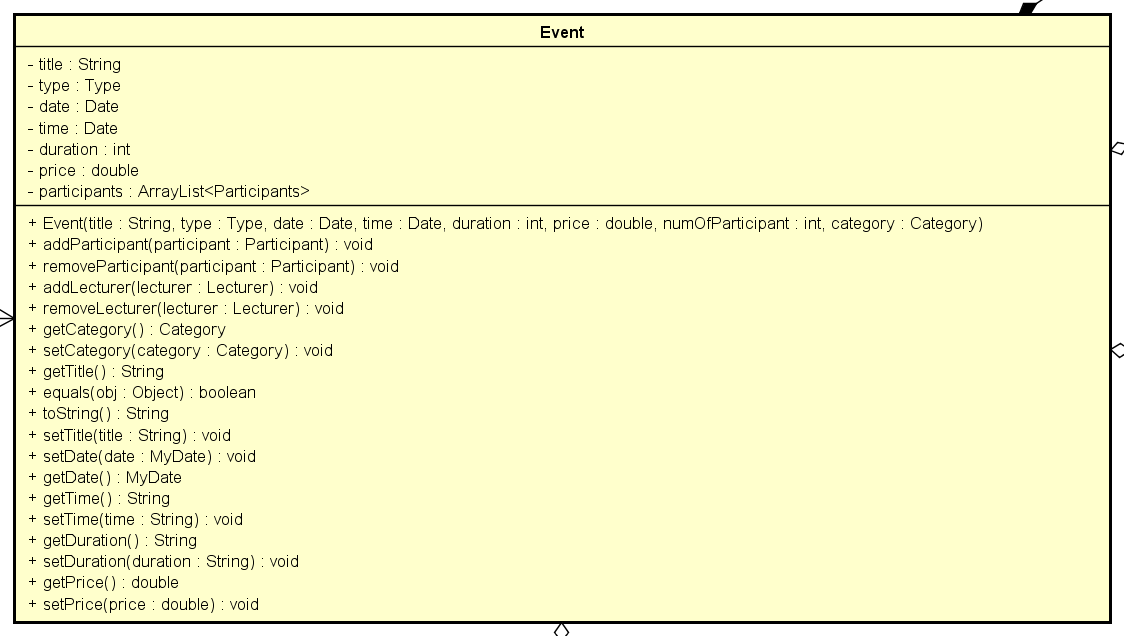


Figure 3, Event class diagram

The figure above is an example of UML class diagram for the Event class.

See full version of class diagram in Appendix E

# Design

## System Architect:

The system uses the MVC architectural pattern, which is divides the application into three interconnected layers these are Model, view, and controller.

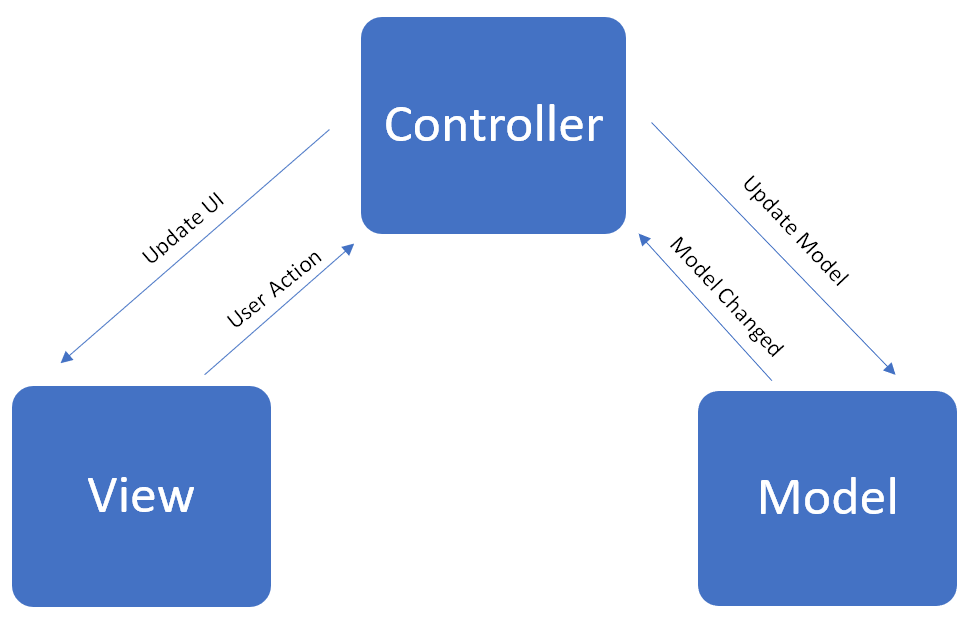


Figure 4, MVC Model

## GUI

The GUI is which technically classes fall in View layer, it has very vital role which is functioning as a mediator between user and other parts of the system, it is consists of menu bar and multiple buttons for the main widow, and for sub windows; the combo box and the table view they have been used significantly.

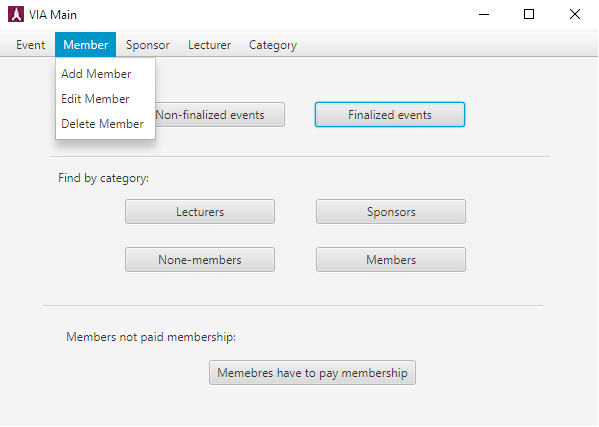


Figure 5, User Interface

## Model Classes:

Model classes those are representing the blueprints of objects, the one class is a single entity which mean single object has its own fields and methods, the class below is a Lecturer class, so Lecturer object can be instantiated based on it.



Figure 6, Lecturer model class

## Controller classes:

Controller classes are those classes interconnecting the model classes to view classes, so they play a mediator role between GUI and actual models, the class below is an AddLecturerController, it is using a lecturer class as a filed with one-to-many relationship, so array list has been used.

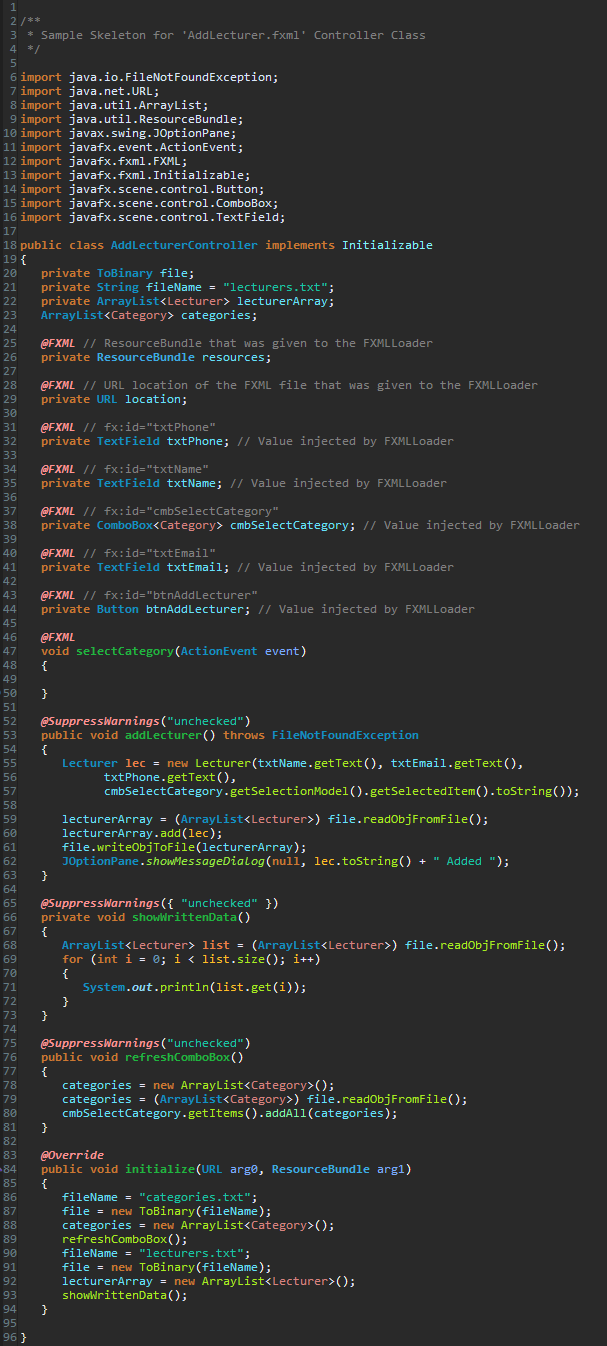


Figure 7, AddLecturerController

# Implementation

Implementation phase was made based on the VIA case requirements, use case description and System Development Life Cycle. Having all the tools, the implementation of the classes started. Late when the classes were done, based on the use case descriptions work on Java FX interfaces has begun. For connecting the Java FX with the Java code some additional classes have been made and tested. For instance, EditEventController class connects the Java FX interface with the Event class where the information about the selected event can be changed. Looping for the desired event is possible through the combo box that displays additional information, in order to make the searching task more accessible. After getting a functional code, some modifications were done in the class diagrams in order to fit the code. The system is using binary files in order to store information.

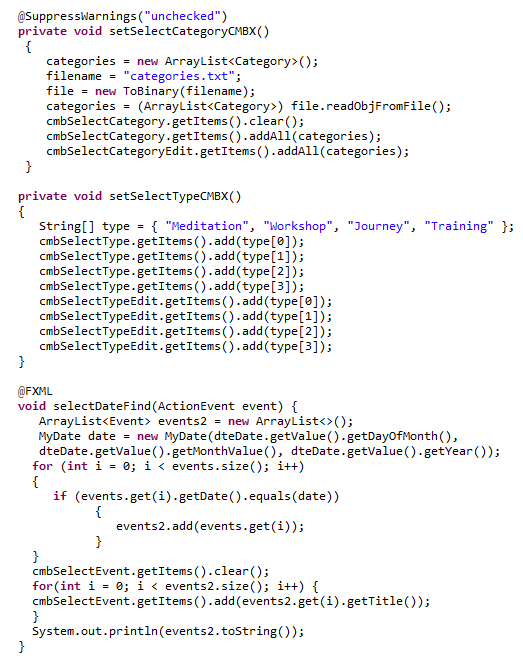


Figure 8, Comboboxes and Binary files reading

# Test

During the testing phase, to test the systems functionality it was decided to use the “Black box” method. The results where satisfactory, meaning, the program worked as it should.

## Test Specifications

Table 2, Test Specifications



As we can see in the table above the system passed all the tested parts.

# Results and Discussion

The single user system created for the VIA company is based on user-friendly system criteria that will definitely make a change on the company’s future.

The implemented system is able to keep track of every member and non-member, create and manage events as well as manage sponsors and lecturers. In addition, it can add, edit and delete participant from an event and also to keep track of members that have to pay membership. Furthermore, it also displays a list of finalized and non-finalized events, members, non-members, lecturers and sponsors given category and a list of members that need to pay membership. The system was created in order to be maintained and improved afterwards with new innovative features.

There are also some components that are not implemented yet, such as a method that controls if there are two or more participants with the same name.

# Conclusion

As a result, of the following project description, the group members have concluded that the requirements given from Vipassana Insight Awareness were successfully fulfilled. A single user system has been proposed. This system is able to manage all the functions provided in the code. The group members mutually agreed that the following project needs further development in order to become a real asset.

# Project future

Reflect on your project from a technical viewpoint and describe what you would change if you could.

Suggest how the project could be improved or made ready for production. Discuss scalability, suggest possible spin offs, what is needed, missing, etc.?

# References

**Note: Use the standard reference method: Harvard Anglia. A very good reference tool is Mendeley** (Mendeley.com 2016), **ask VIA Library if you need help.**

Banger, D., 2014. A Basic Non-Functional Requirements Checklist « Thoughts from the Systems front line.... Available at: https://dalbanger.wordpress.com/2014/01/08/a-basic-non-functional-requirements-checklist/ [Accessed January 31, 2017].

Business Analyst Learnings, 2013. MoSCoW : Requirements Prioritization Technique — Business Analyst Learnings. , pp.1–5. Available at: https://businessanalystlearnings.com/ba-techniques/2013/3/5/moscow-technique-requirements-prioritization [Accessed January 31, 2017].

Dawson, C.W., 2009. *Projects in Computing and Information Systems*, Available at: http://www.sentimentaltoday.net/National\_Academy\_Press/0321263553.Addison.Wesley.Publishing.Company.Projects.in.Computing.and.Information.Systems.A.Students.Guide.Jun.2005.pdf.

Gamma, E. et al., 2002. *Design Patterns – Elements of Reusable Object-Oriented Software*, Available at: http://books.google.com/books?id=JPOaP7cyk6wC&pg=PA78&dq=intitle:Design+Patterns+Elements+of+Reusable+Object+Oriented+Software&hl=&cd=3&source=gbs\_api%5Cnpapers2://publication/uuid/944613AA-7124-44A4-B86F-C7B2123344F3.

IEEE Computer Society, 2008. *IEEE Std 829-2008, IEEE Standard for Software and System Test Documentation*,

Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*,

Mendeley.com, 2016. Homepage | Mendeley. Available at: https://www.mendeley.com/ [Accessed February 2, 2017].

YourCoach, S.M.A.R.T. goal setting | SMART | Coaching tools | YourCoach Gent. Available at: http://www.yourcoach.be/en/coaching-tools/smart-goal-setting.php [Accessed August 19, 2017].

# List of Appendixes

Appendix A – Project Description;

Appendix B – Use Case Diagrams;

Appendix C – Use Case Description;

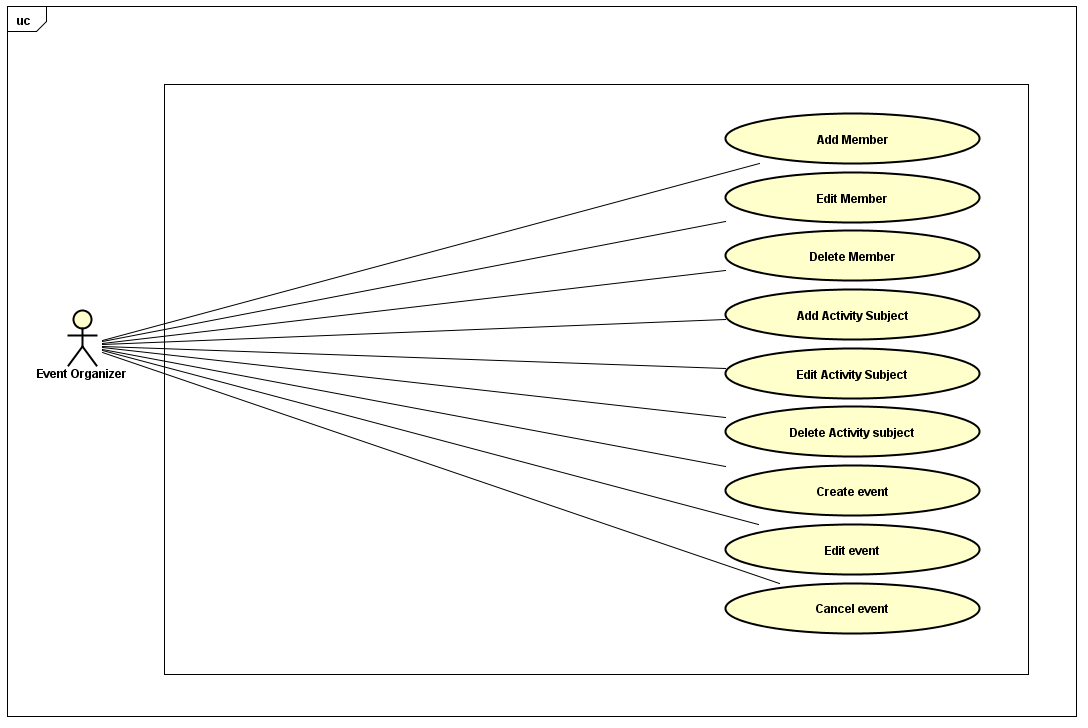
Appendix D – Activity Diagrams;

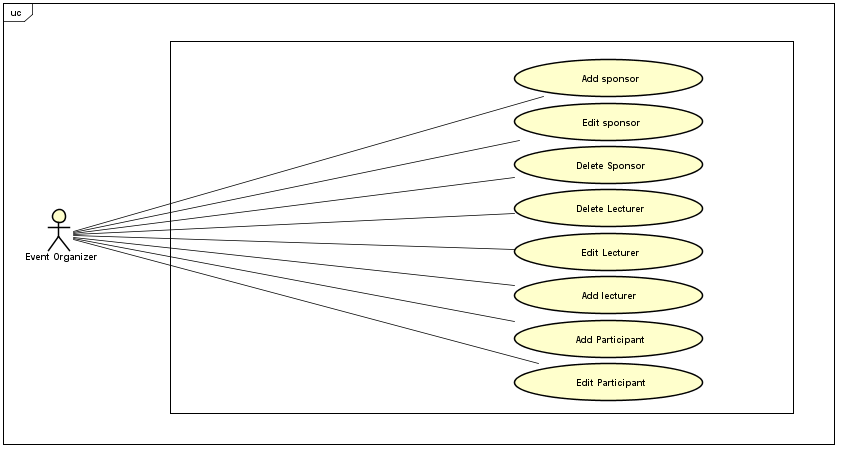
Appendix E – UML Class Diagrams;

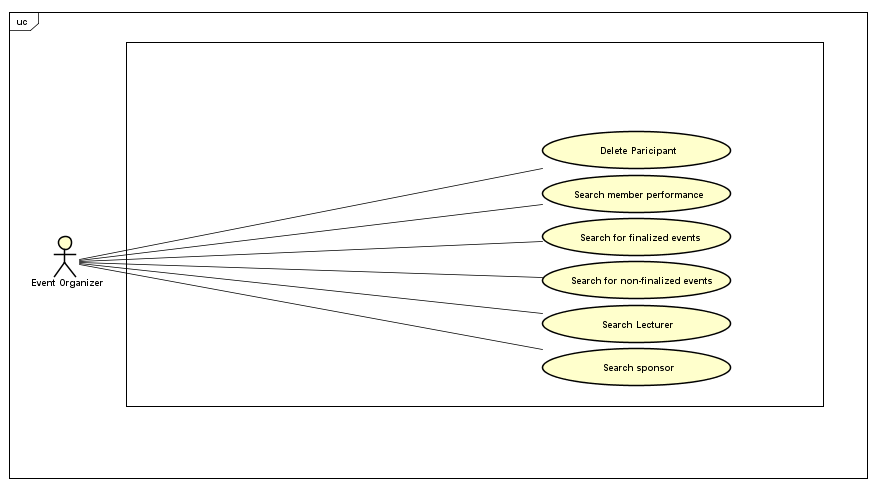
1. Project Description

Insert the original Project Description in here.

1. Use case diagram







1. Use Case Description

Table 3, Add a member

|  |  |
| --- | --- |
| ID: | #1 |
| Title: | Add a member |
| Description: | The user will add a member |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | A is member added |
| Main  Success Scenario: | 1. User chooses to add a new member. 2. User enter new member information (name, address, phone, email, date for payment of membership, date of membership, preferences). 3. User validates. 4. System store new members information in data file storage. |
| Extensions: | 4., System not able to reach a data file storage shows an error message, |

Table 4, Edit a member

|  |  |
| --- | --- |
| ID: | #2 |
| Title: | Edit a member |
| Description: | The user will edit a member |
| Primary Actor: | User |
| Preconditions: | At least one member exists. |
| Postconditions: | A member is edited. |
| Main  Success Scenario: | 1. User chooses to edit a new member. 2. User find member to modify. 3. Modify one or more of (name, address, phone, email, date for payment of membership, date of membership, preferences). 4. User validates. 5. System modify member’s information in data file storage. |
| Extensions: | 5., System not able to reach a data file storage shows an error message. |

Table 5, Delete a member

|  |  |
| --- | --- |
| ID: | #3 |
| Title: | Delete a member |
| Description: | The user will delete a member |
| Primary Actor: | User |
| Preconditions: | At least one member exists. |
| Postconditions: | A member is deleted |
| Main  Success Scenario: | 1. user chooses to delete member. 2. User find a member to delete 3. User delete the member 4. User validates 5. System delete member from data file storage |
| Extensions: | 5., System not able to reach a data file storage shows an error message. |

Table 6, Add an activity subject.

|  |  |
| --- | --- |
| ID: | #4 |
| Title: | Add an activity subject. |
| Description: | The user will add an activity subject. |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | An activity subject is added. |
| Main  Success Scenario: | 1. User chooses to add a new activity subject. 2. User enter new activity subject (Subject name) 3. User validates 4. System store new activity subject information in data file storage |
| Extensions: | 4., 8., 12. System not able to reach a data file storage shows an error message, |

Table 7, Delete an activity subject.

|  |  |
| --- | --- |
| ID: | #5 |
| Title: | Delete an activity subject. |
| Description: | The user will delete an activity subject. |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | An activity subject is deleted. |
| Main  Success Scenario: | 1. User chooses to delete an activity subject. 2. User delete an activity subject 3. User validates 4. System delete activity subject information from data file storage |
| Extensions: | 4., 8., 12. System not able to reach a data file storage shows an error message, |

Table 8, Add an event

|  |  |
| --- | --- |
| ID: | #6 |
| Title: | Add an event |
| Description: | The user will add an event |
| Primary Actor: | User |
| Preconditions: | At least, one lecturer exists, and one event activity subject exist. |
| Postconditions: | An event is added. |
| Main  Success Scenario: | 1. User chooses to add a new event. 2. User select activity subject for event 3. User enter new event information (type, title, date, time, duration, lecturer/lecturers, price, discounted price) 4. User validates 5. System store new event information in data file storage |
| Extensions: | 4., System not able to reach a data file storage shows an error message,  2., If activity subject not found, go to #4 step2  2., if lecturer not found, go to #13 step2 |

Table 9, Edit an event

|  |  |
| --- | --- |
| ID: | #7 |
| Title: | Edit an event |
| Description: | The user will edit an event |
| Primary Actor: | User |
| Preconditions: | At least one event exists. |
| Postconditions: | An event is edited. |
| Main  Success Scenario: | 1. User chooses to edit an event. 2. User find event to modify 3. User change activity subject for event 4. User modify one or more of (type, title, date, time, duration, lecturer/lecturers, price, discounted price). 5. User validates. 6. System modify event’s information in data file storage. |
| Extensions: | 5., System not able to reach a data file storage shows an error message, |

Table 10, Delete an event

|  |  |
| --- | --- |
| ID: | #8 |
| Title: | Delete an event |
| Description: | The user will delete an event |
| Primary Actor: | User |
| Preconditions: | At least, one event exists. |
| Postconditions: | An event is deleted. |
| Main  Success Scenario: | 1. User chooses to delete an event. 2. User find event to delete 3. User delete the event 4. User validates 5. System delete the event from data file storage |
| Extensions: | 5., System not able to reach a data file storage shows an error message. |

Table 11, Add a sponsor

|  |  |
| --- | --- |
| ID: | #9 |
| Title: | Add a sponsor |
| Description: | The user will add a sponsor |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | A sponsor is added. |
| Main  Success Scenario: | 1. User chooses to add a new sponsor. 2. User enter new sponsor information (name, address, phone, email, activity subject) 3. User validates 4. System store new sponsor information in data file storage. |
| Extensions: | 4., System not able to reach a data file storage shows an error message, |

Table 12, Edit a sponsor

|  |  |
| --- | --- |
| ID: | #10 |
| Title: | Edit a sponsor |
| Description: | The user will edit a sponsor |
| Primary Actor: | User |
| Preconditions: | At least, one sponsor exists. |
| Postconditions: | A sponsor is edited. |
| Main  Success Scenario: | 1. User chooses to edit a new sponsor. 2. User find sponsor to modify. 3. Modify one or more of (name, address, phone, email, activity subject) 4. User validates 5. System modify sponsor’s information in data file storage |
| Extensions: | 5., System not able to reach a data file storage shows an error message, |

Table 13, Delete a sponsor

|  |  |
| --- | --- |
| ID: | #11 |
| Title: | Delete a sponsor |
| Description: | The user will delete a sponsor |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | At least, one sponsor exists. |
| Main  Success Scenario: | 1. user chooses to delete a sponsor. 2. User find a sponsor to delete. 3. User delete the sponsor. 4. User validates. 5. System delete the sponsor from data file storage. |
| Extensions: | 5., System not able to reach a data file storage shows an error message, |

Table 14, Add a lecturer

|  |  |
| --- | --- |
| ID: | #12 |
| Title: | Add a lecturer |
| Description: | The user will add a lecturer |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | A lecturer is added. |
| Main  Success Scenario: | 1. User chooses to add a new lecturer. 2. If the lecturer is a sponsor go to 6 3. User enter new lecturer information (name, address, phone, email, activity subject.) 4. User validates. 5. System store new lecturer information in data file storage. 6. User select a sponsor to add as a lecturer. 7. User validates. 8. System store new lecturer information in data file storage. |
| Extensions: | 5., 8., System not able to reach a data file storage shows an error message,  3., If activity subject not found, go to #4 step2  6., if no sponsor available, the system shows a message stating no there is sponsors available |

Table 15, Edit a lecturer

|  |  |
| --- | --- |
| ID: | #13 |
| Title: | Edit a lecturer |
| Description: | The user will edit a lecturer |
| Primary Actor: | User |
| Preconditions: | At least, one lecturer exists. |
| Postconditions: | A lecturer is edited. |
| Main  Success Scenario: | 1. User chooses to edit a lecturer. 2. User find a lecturer to modify 3. User modify one or more of (name, address, phone, email, activity subject) 4. User validates. 5. System modify lecturer’s information in data file storage |
| Extensions: | 5., System not able to reach a data file storage shows an error message. |

Table 16, Delete a lecturer

|  |  |
| --- | --- |
| ID: | #14 |
| Title: | Delete a lecturer |
| Description: | The user will delete a lecturer |
| Primary Actor: | User |
| Preconditions: | At least, one lecturer exists. |
| Postconditions: | A lecturer is deleted. |
| Main  Success Scenario: | 1. User chooses to delete a lecturer. 2. User find a lecturer to delete 3. User delete lecturer 4. User validates. 5. System delete lecturer’s information in data file storage |
| Extensions: | 5., System not able to reach a data file storage shows an error message. |

Table 17, Add participant to an event

|  |  |
| --- | --- |
| ID: | #15 |
| Title: | Add participant to an event |
| Description: | The user will add participant to an event |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | Participant is added to an event |
| Main  Success Scenario: | 1. User chooses to add a new participant to event 2. User find event to add participant to. 3. If participant is member go to 8 4. User enter participant information (name, address, phone, email, preferences). 5. User validates 6. System make participant type false (is it member?) 7. System store participant’s information in data file storage 8. User select participant from members. 9. User add that member (name) 10. User validates. 11. System make participant type true (is it member) 12. System store participant’s information in data file storage |
| Extensions: | 7., 12. System not able to reach a data file storage shows an error message, |

Table 18, Delete participant from an event

|  |  |
| --- | --- |
| ID: | #16 |
| Title: | Delete participant from an event |
| Description: | The user will delete participant from an event |
| Primary Actor: | User |
| Preconditions: | At least, one participant exists. |
| Postconditions: | Participant is deleted from an event |
| Main  Success Scenario: | 1. User chooses to delete a participant from an event. 2. User find event to delete participant from. 3. User find a participant to delete. 4. User delete a participant 5. User validates. 6. System delete participant’s information from data file storage. |
| Extensions: | 6., System not able to reach a data file storage shows an error message, |

Table 19, search for finalized events

|  |  |
| --- | --- |
| ID: | #17 |
| Title: | search for finalized events |
| Description: | The user will search for finalized events |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | finalized events is shown |
| Main  Success Scenario: | 1. User chooses to find finalized events. 2. User search for finalized events 3. list of finalized events is shown |
| Extensions: | 3., if no finalized events available, no events to show |

Table 20, search for non-finalized events

|  |  |
| --- | --- |
| ID: | #18 |
| Title: | search for non-finalized events |
| Description: | The user will search for non-finalized events |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | non-finalized events are shown |
| Main  Success Scenario: | 1. User chooses to find non-finalized events. 2. User search for non-finalized events 3. list of non-finalized events is shown |
| Extensions: | 3., if no events available, no events to show |

Table 21, search for sponsor

|  |  |
| --- | --- |
| ID: | #19 |
| Title: | search for sponsor |
| Description: | The user will search for sponsor in specific category |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | sponsors in specific category is shown |
| Main  Success Scenario: | 1. user chooses to find sponsors in specific category 2. user select the desired category 3. user find sponsors in this category 4. list of sponsors is shown |
| Extensions: | 4., if no sponsor available, no sponsor to show |

Table 22, search for lecturer

|  |  |
| --- | --- |
| ID: | #20 |
| Title: | search for lecturer |
| Description: | The user will search for lecturers in specific category |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | lecturers in specific category is shown |
| Main  Success Scenario: | 1. user chooses to find lecturers in specific category 2. user select the desired category 3. user find lecturers in this category 4. list of lecturers is shown |
| Extensions: | 4., if no lecturers available, no lecturers to show |

Table 23, search for members’ preference

|  |  |
| --- | --- |
| ID: | #21 |
| Title: | search for members’ preference |
| Description: | The user will search for members’ preference in specific category |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | members’ preference in specific category is shown |
| Main  Success Scenario: | 1. user chooses to find members’ preference in specific category 2. user select the desired category 3. user find members’ preference in this category 4. list of members is shown |
| Extensions: | 4., if no members available, no members’ preference to show |

Table 24, find the members those not paid their membership

|  |  |
| --- | --- |
| ID: | #22 |
| Title: | find the members those not paid their membership |
| Description: | The user will search for the members those not paid their membership |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | members those not paid their membership is shown |
| Main  Success Scenario: | 1. user chooses to find the members those not paid their membership 2. user select the desired category 3. list of members is shown |
| Extensions: | 4., if all members paid, no members to show |

Table 25, Use files as data storage.

|  |  |
| --- | --- |
| ID: | #23 |
| Title: | Use files as data storage. |
| Description: | The system must use files as data storage. |
| Primary Actor: | System |
| Preconditions: | None |
| Postconditions: | Data is stored in data file storage. |
| Main  Success Scenario: | 1. The system stores the data in a data file storage. |
| Extensions: | 1. System not able to reach a data file storage shows an error message, |

Table 26, System developed in Java

|  |  |
| --- | --- |
| ID: | #24 |
| Title: | System developed in Java |
| Description: | The system must be developed using Java |
| Primary Actor: | System |
| Preconditions: | None |
| Postconditions: | System is implemented in Java |
| Main  Success Scenario: | 1. Using java to implement the system |
| Extensions: | none |

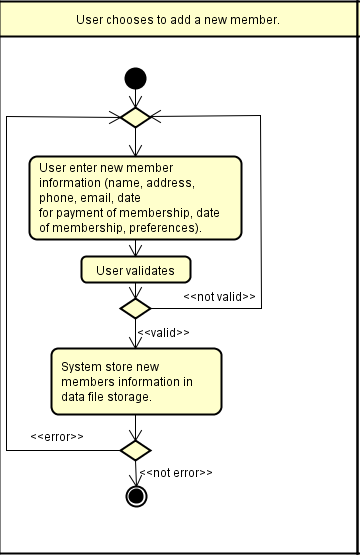
Table 27, Search response time within 2 seconds 95% of the time

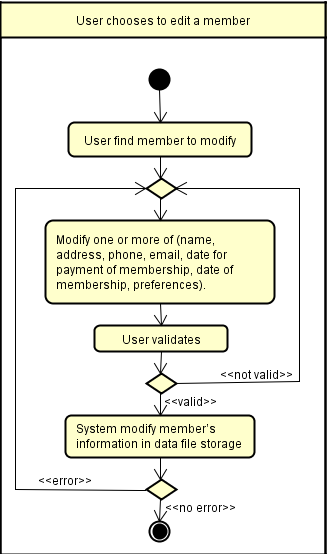
|  |  |
| --- | --- |
| ID: | #25 |
| Title: | Search response time within 2 seconds 95% of the time |
| Description: | The system needs to answer within 2 seconds 95% of the time |
| Primary Actor: | System |
| Preconditions: | None |
| Postconditions: | Response time no longer than 2 seconds 95% of the time |
| Main  Success Scenario: | 1. Using local storage data file on user’s machine |
| Extensions: | 1., Bad performance of user’s machine |

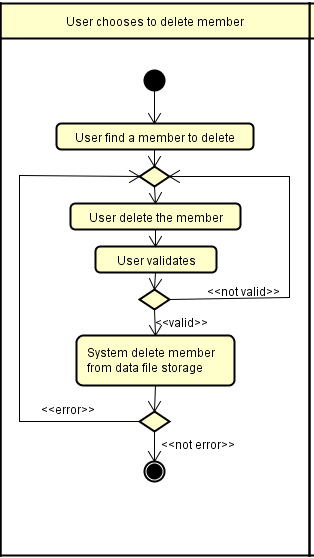
Table 28, The system usability by end users

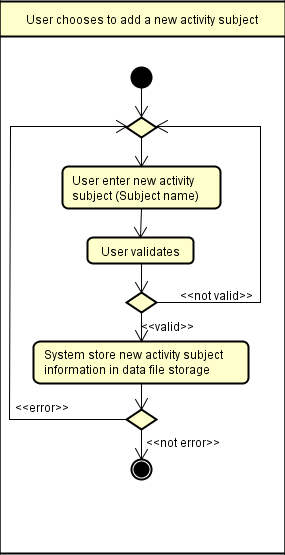
|  |  |
| --- | --- |
| ID: | #26 |
| Title: | The system usability by end users |
| Description: | The system must be usability tested by end users |
| Primary Actor: | System |
| Preconditions: | None |
| Postconditions: | System usability is tested by end user |
| Main  Success Scenario: | 1. User to test the system before deployment |
| Extensions: | None |

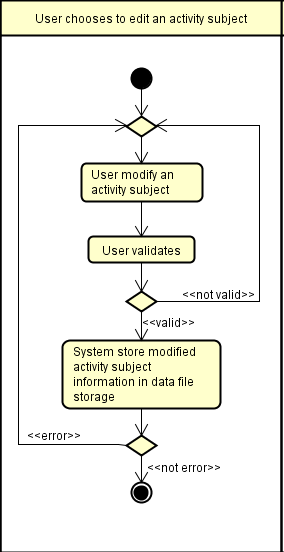
1. Activity Diagrams

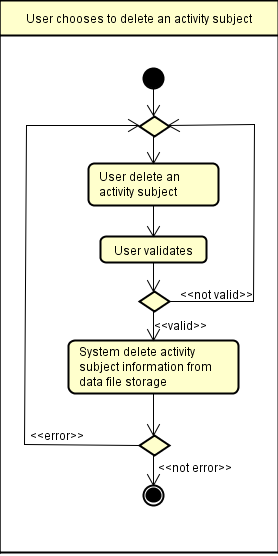


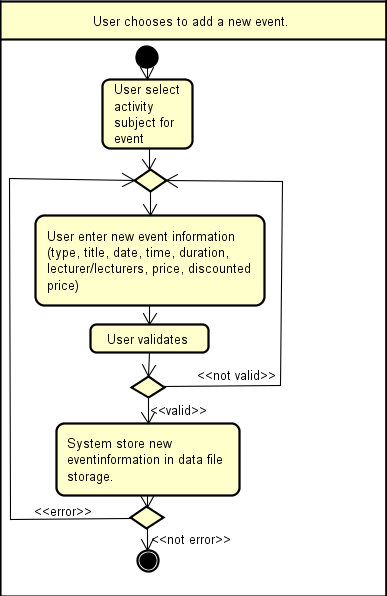


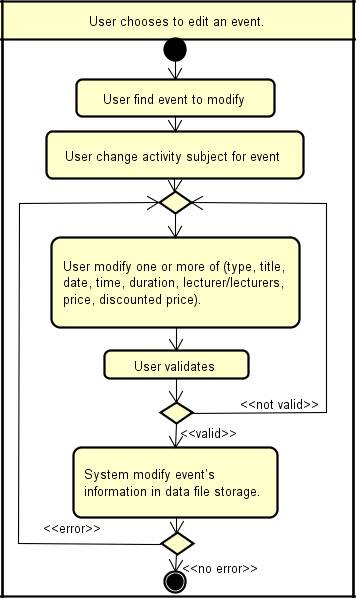


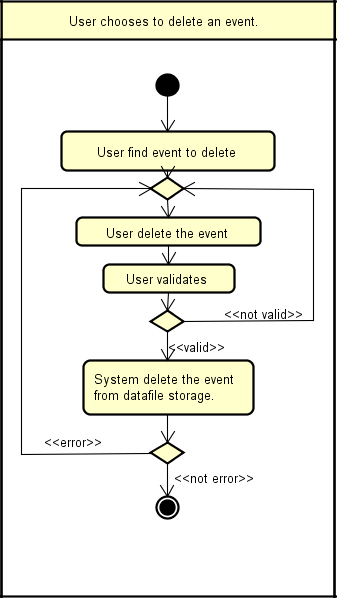


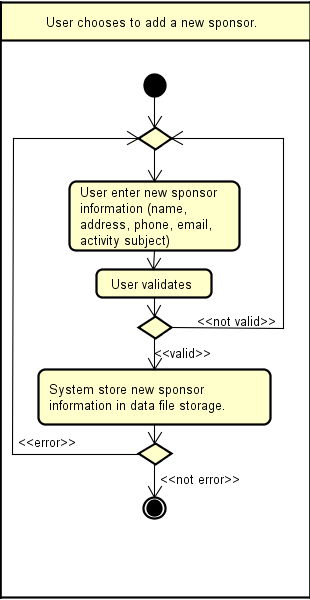


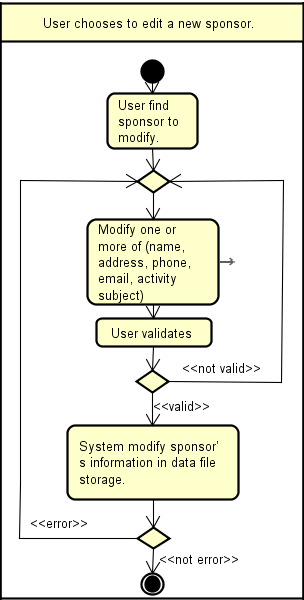


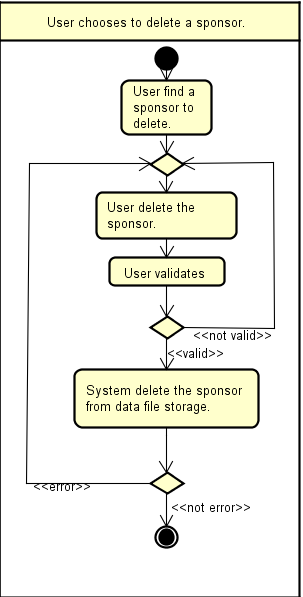


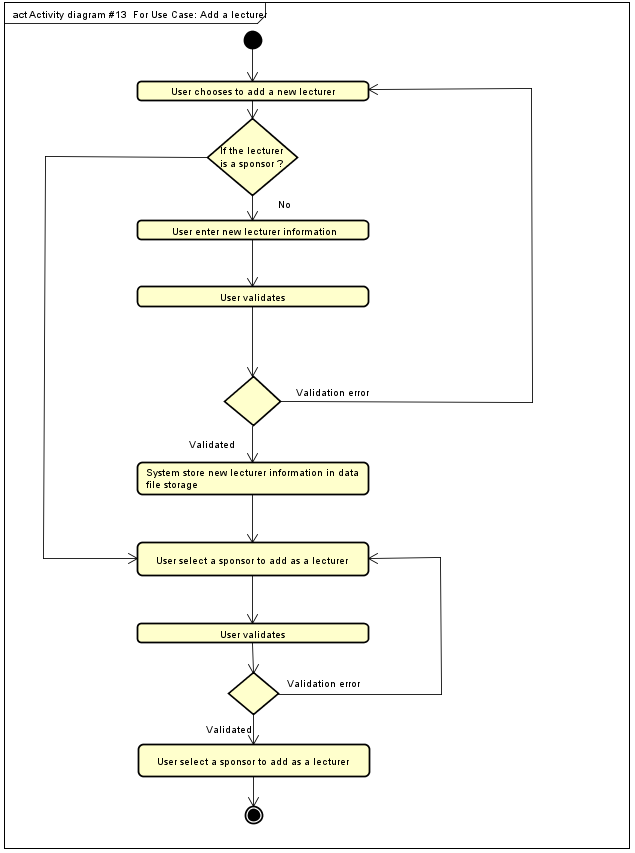


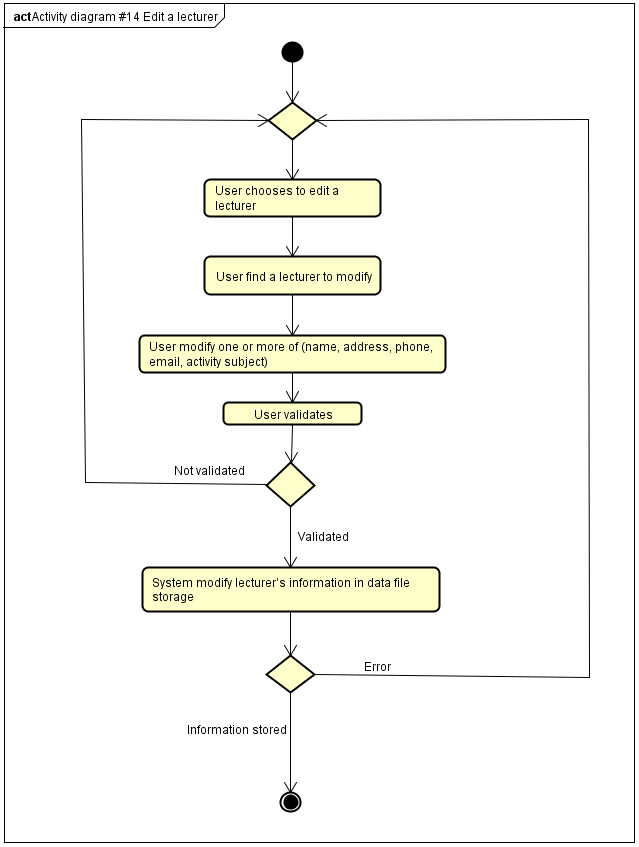


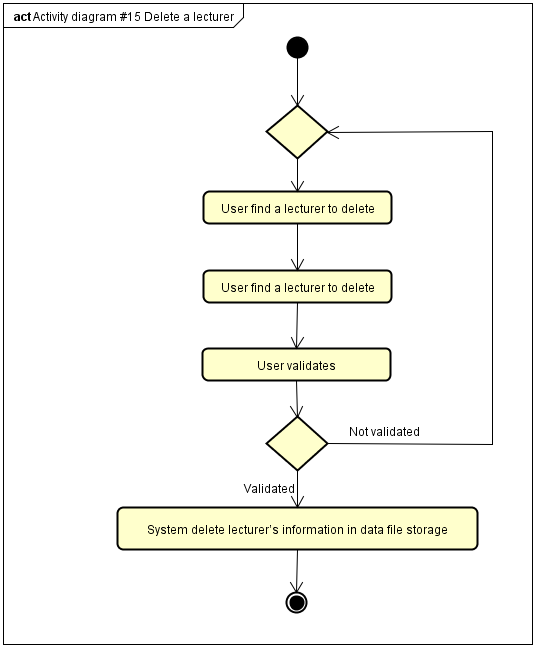


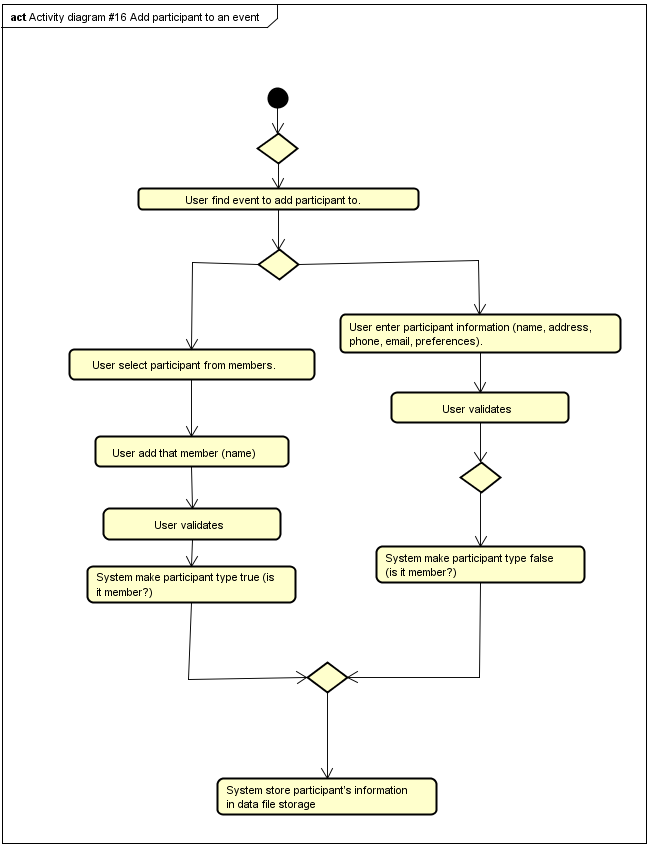


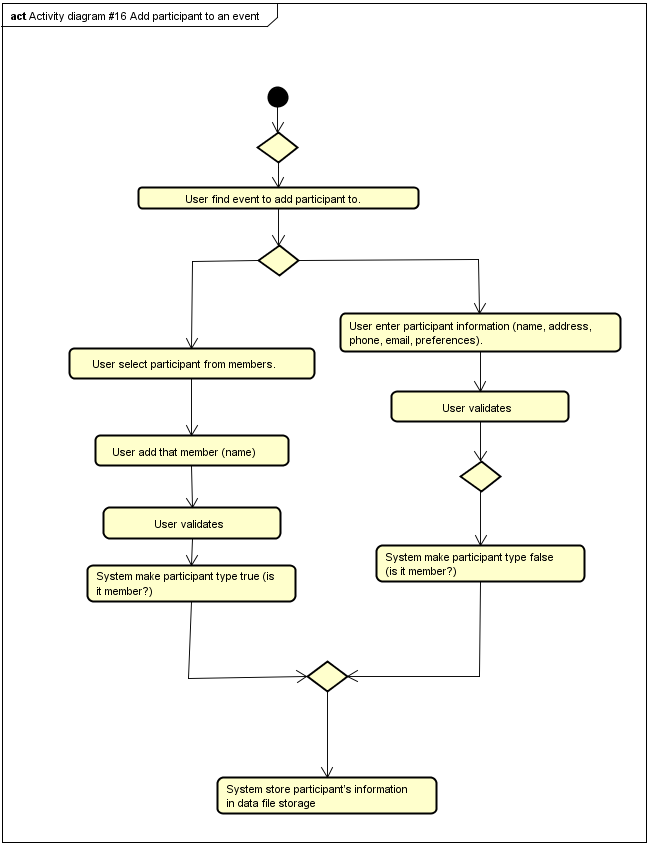


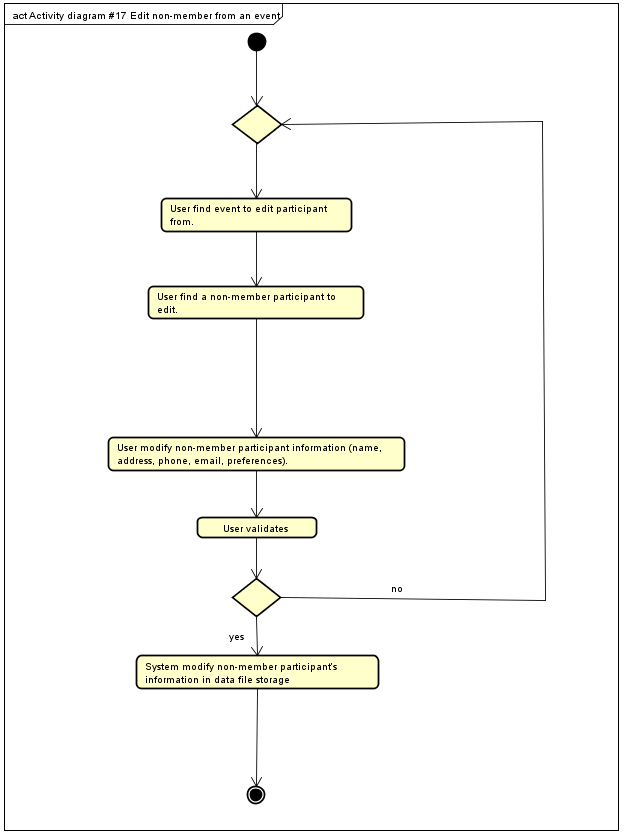


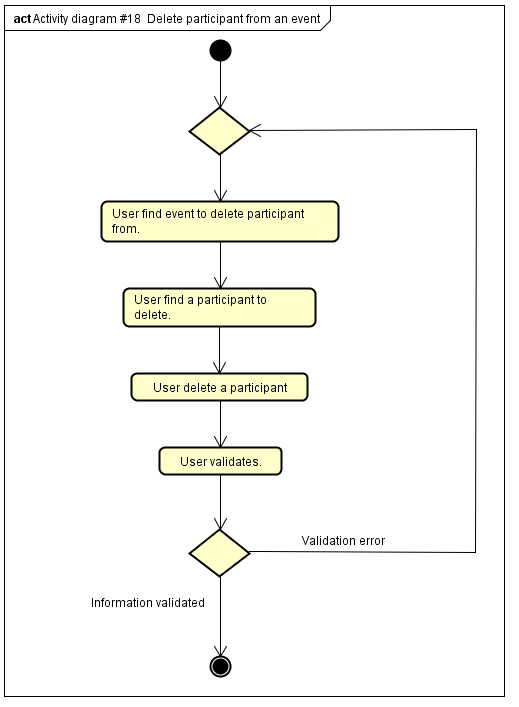


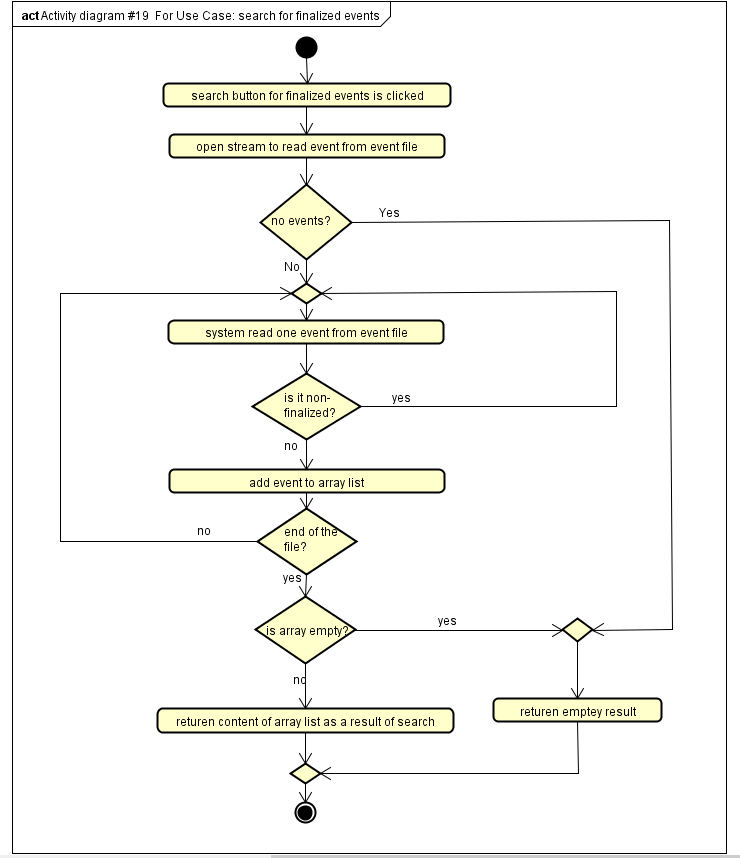


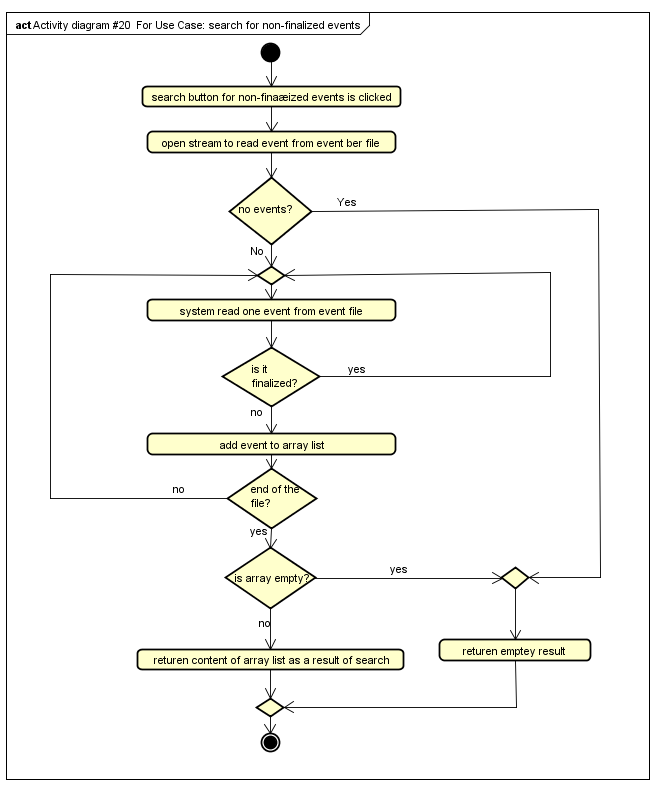


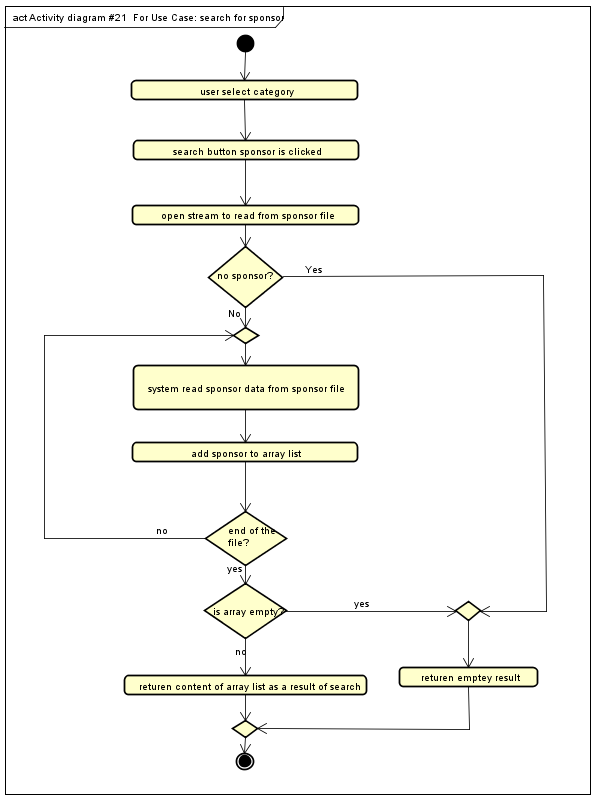


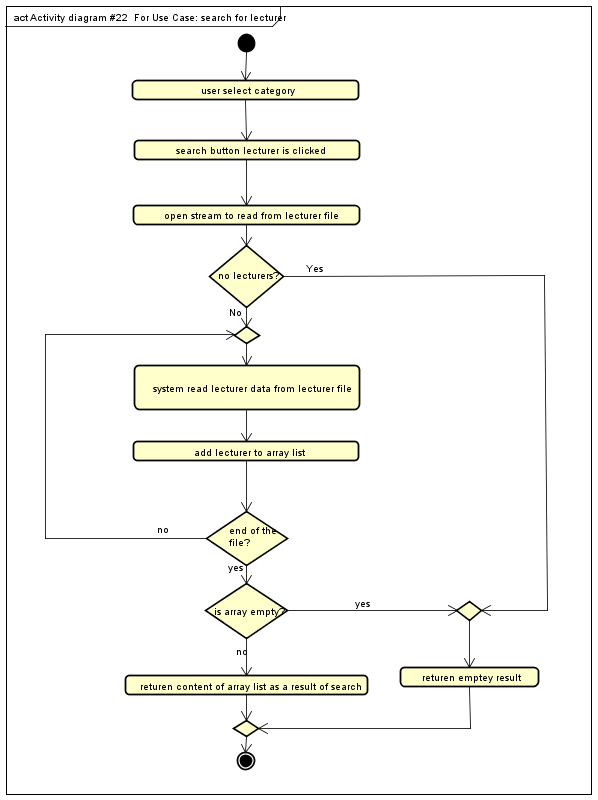


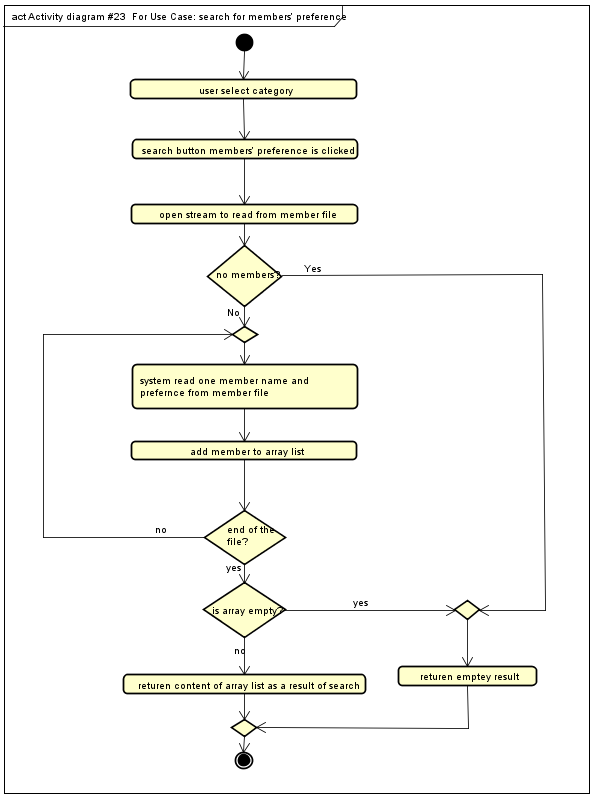


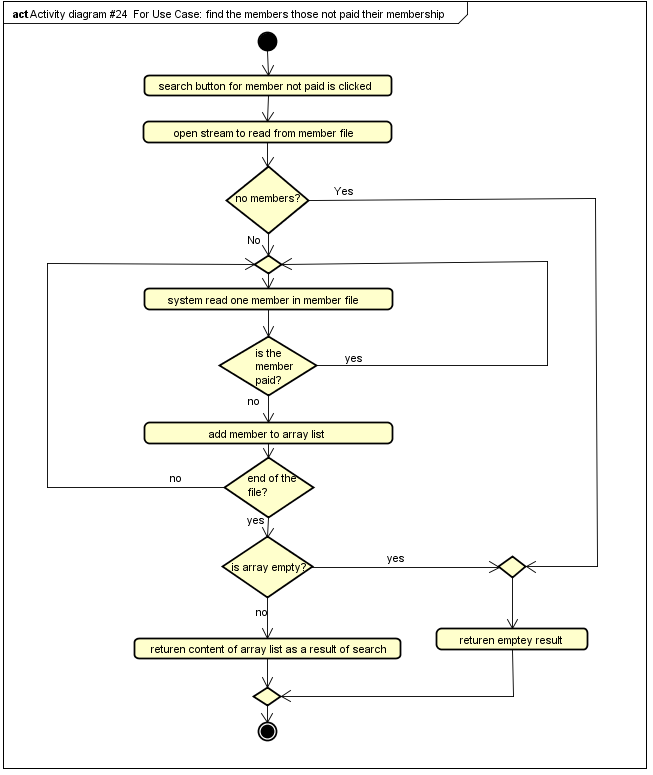




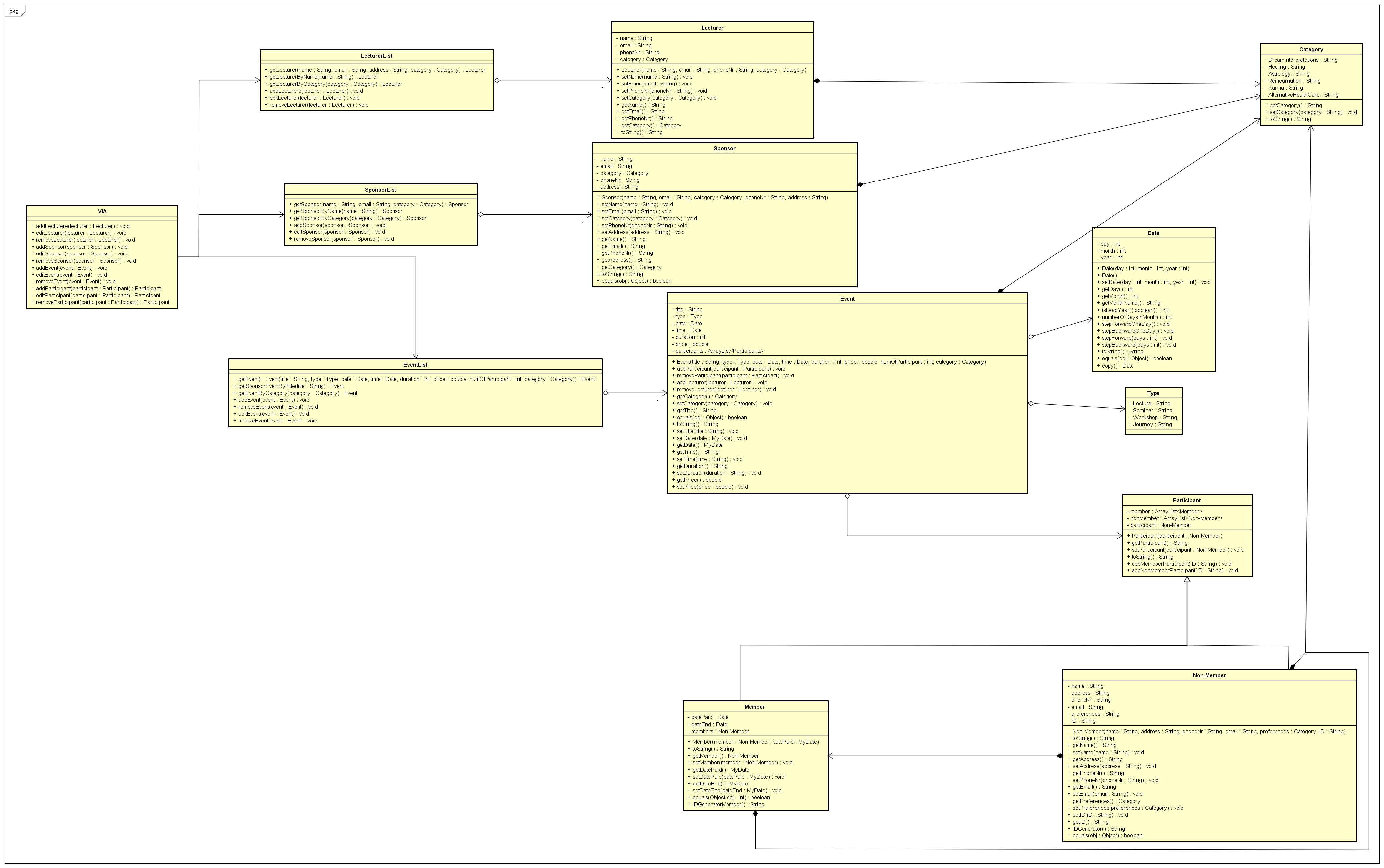








1. UML Class Diagram



1. SmartDraw, LLC, definition of class diagram, <https://www.smartdraw.com/class-diagram/>, visited 18/12/2017 [↑](#footnote-ref-2)