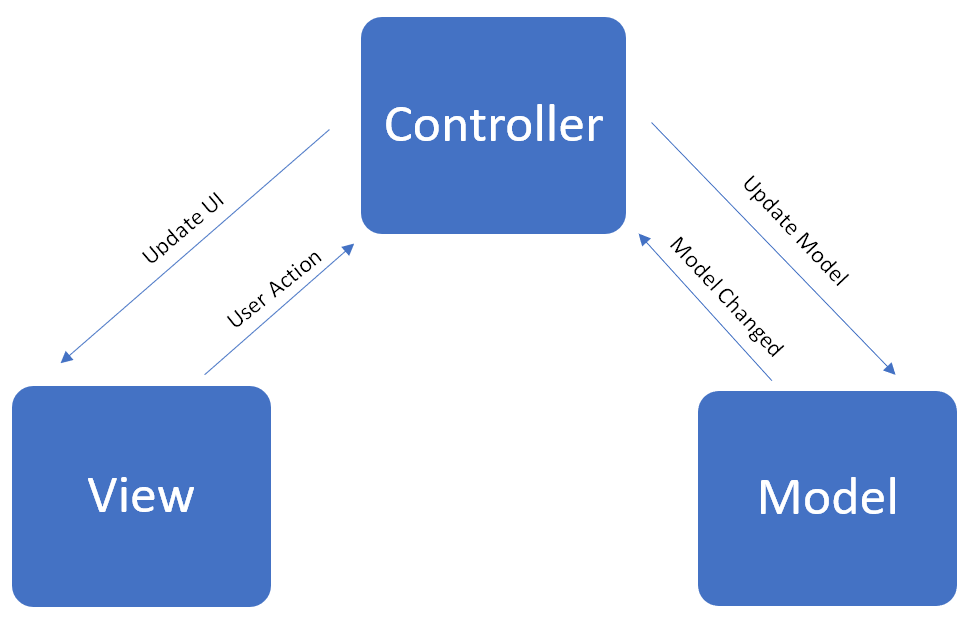
# System design:

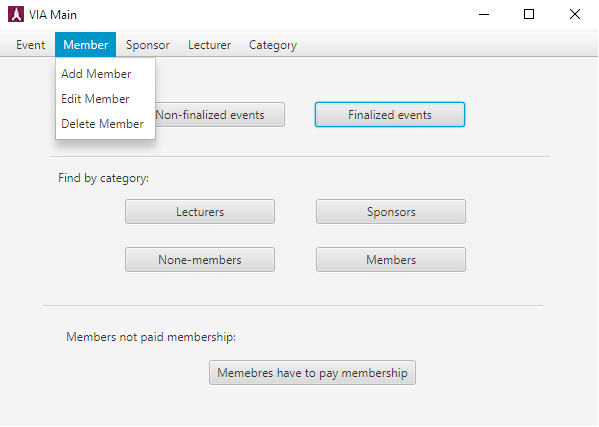
## System Architect:

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



## 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.



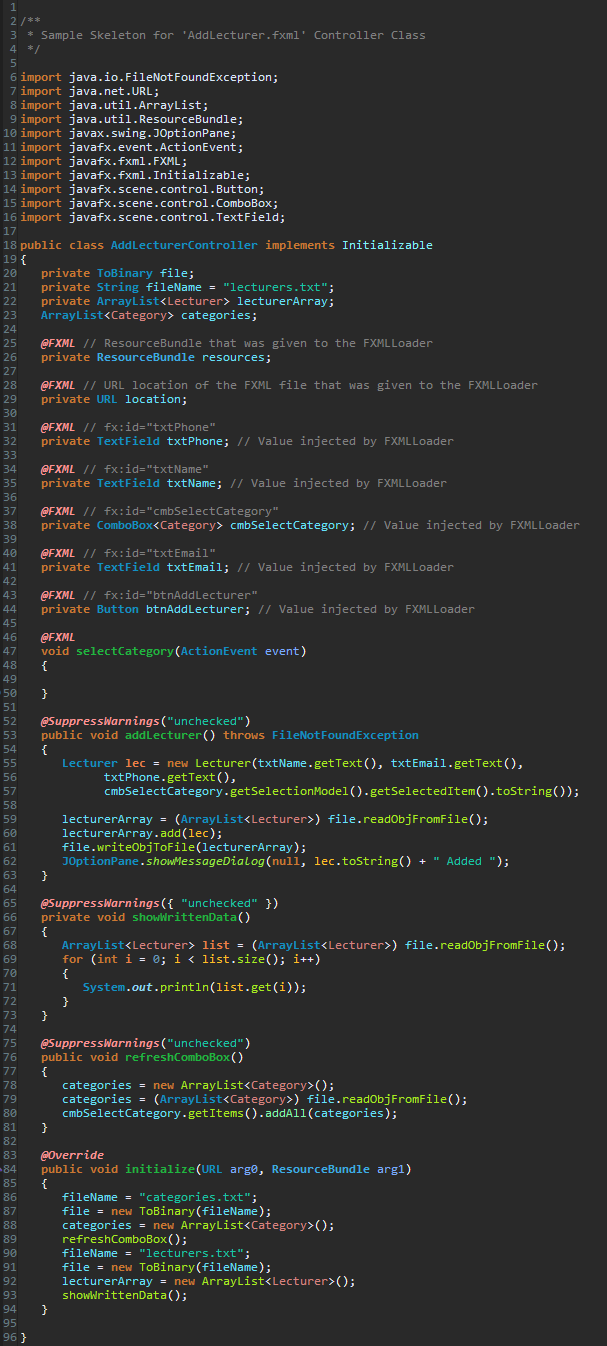
## Model Classes:

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.



## 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.



**Analysis**

In the Analysis stage of the proces, it’s necessery

In the Analysis stage of the project report, it’s necessary to understand the client’s wishes (needs) to meet their requirements

This section has the following sub-sections:

* Requirements
* Use Case modelling
* Activity diagrams
* Analysis class diagrams

Requirements

During the interview the client informed of its wishes for the system.

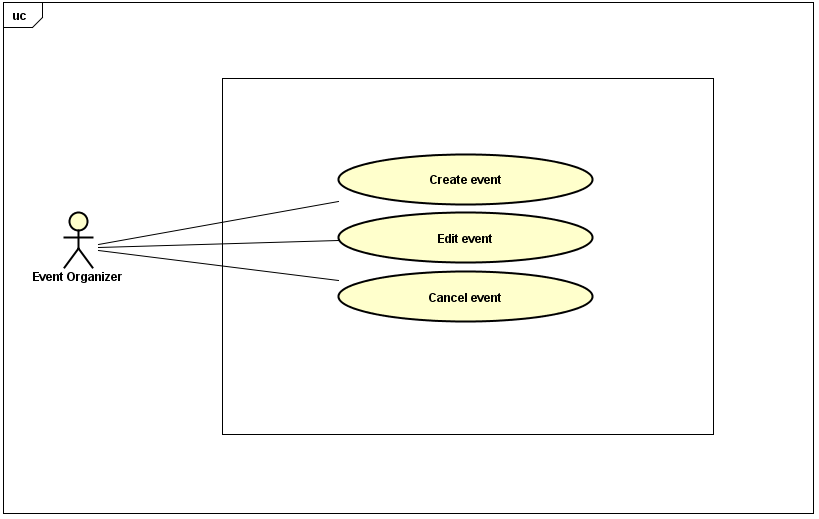
Functional Requirements:

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.

The system must allow user to find the members those not paid their membership.

Non-functional Requirements:

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



The picture represents how the user is able to add, edit and cancel an event using the newly designed system for events.

For full Use Case diagrams see Appendix

**Use Case 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.

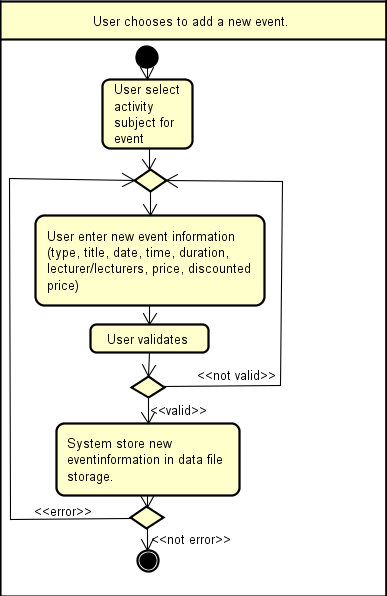
|  |  |
| --- | --- |
| ID: | #7 |
| Title: | Add an event |
| Description: | The user will add an event |
| Primary Actor: | User |
| Preconditions: | At least, one lecturer exists, and one event activity subject exist. |
| Postconditions: | An event is added. |
| Main  Success Scenario: | 1. User chooses 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 |

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

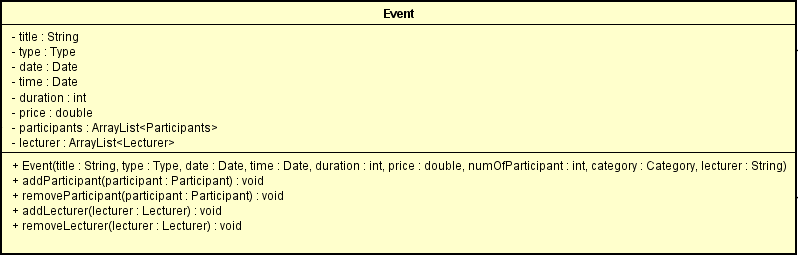
All information, except for “Subject” and “Lecturer” are unique for all events. Meaning that the system doesn’t keep list of possible times or possible prices for events.



Activity diagram

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

**Activity diagrams** are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.



**UML Class diagram**

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.

To view all UML’s check Appendix

**RESULTS**

The single user system created for the VIA company is based on the 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 and also 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. In addition 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.

**IMPLEMENTATION**

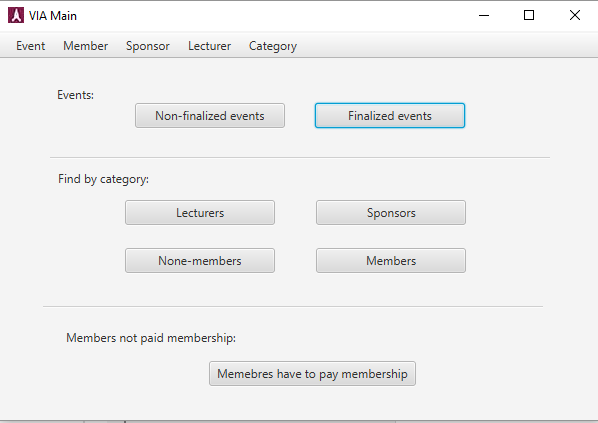
Implementation phase was made based on the VIA case requirements, use case description and waterfall model. Having all the tools, the implementation of the classes started. Late when the classes were done, work on Java FX interfaces has begun based on the use case descriptions. For connecting the Java FX with the Java code some additional classes have been made and tested. 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.

**Testing**

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.



**User guide**



This is the main frame for VIA. Here it is shown a menu bar that is located in the top horizontal part of the main window. Below the menu bar there are multiple buttons for multiple actions.

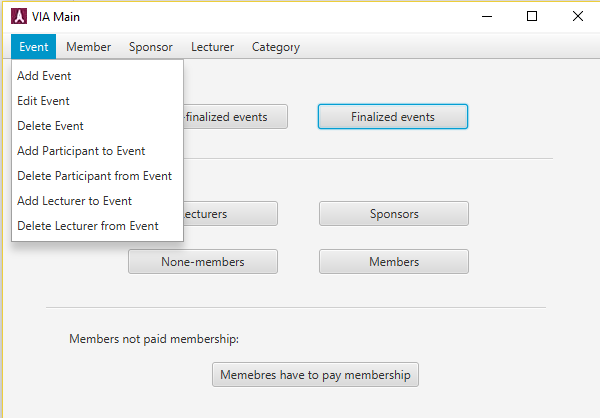
From this window it is possible to access every necessary action the user choses to initiate.

**Using the menu bar**



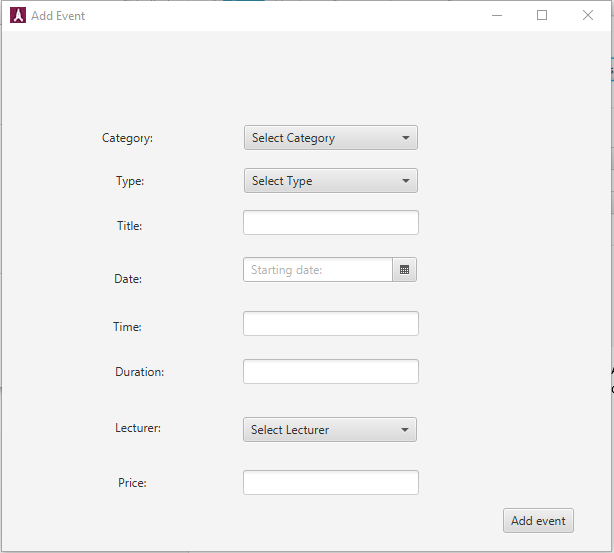
From the menu bar it is possible to access “Event”, “Member”, ”Sponsor”, ”Lecturer” and ”Category”. Clicking on any of these menu bar elements, a drop down list of menu items will appear.

**Using the menu items (Event)**



Clicking on the menu item element named “Event” it is possible to Add, edit and delete a event. In addition it is possible to add participants and lecturers to certain events and delete them.

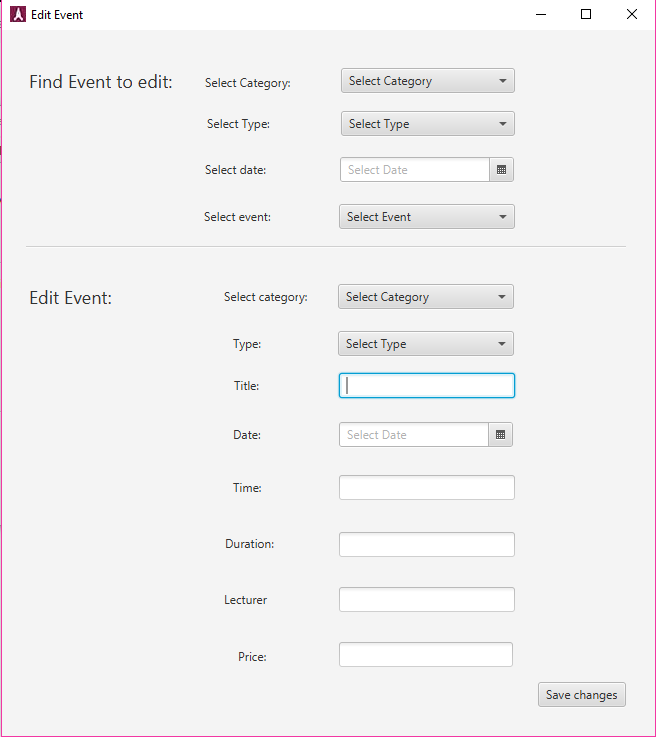
**Add Event**



The menu item “Add Event” is meant to create events.

Next to items (i.e. Category, Type, Title, etc.) there are fields that need to be completed in order to create an event. If one of these fields is not completed it is still possible to create an event but it will be categorized as a, ”Non-Finalized event”.

**Edit Event**

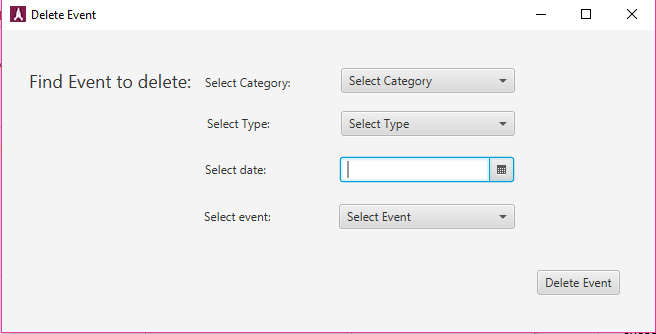


The menu item “Edit Event” is meant to edit events.

In this section the user can find the respective event they wish to edit.

At the top half of the interface it is possible to search for an existing event. When the Event is chosen, all of its corresponding information will be showed in the “Edit Event” section of this interface. From there it is possible to add missing information and to finalize the event. When all of the needed changes are made then it is necessary to press the button “Save changes” to store the changes in the system file for future use.

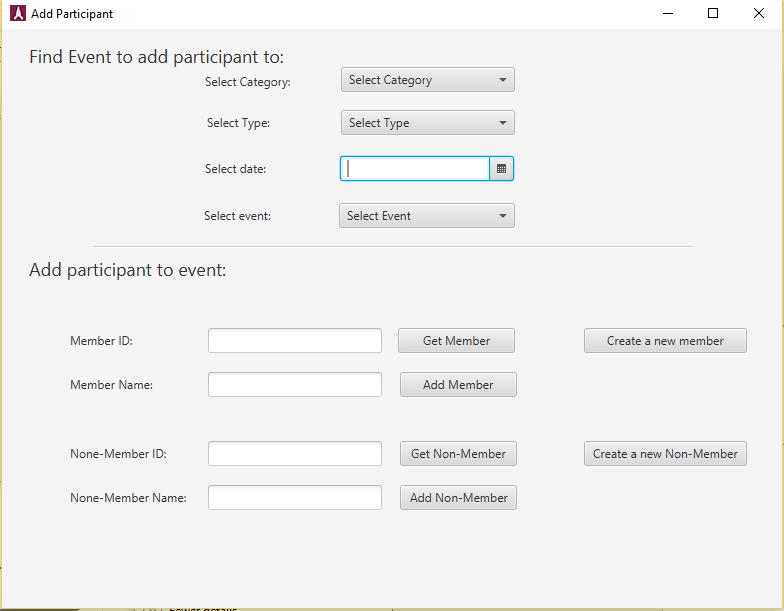
**Delete Event**



In this interface it is possible to delete an existing event.

First the user has to match the corresponding information regarding to the event in mind. When the event is found the user has to select it form the combo box and press the button “Delete” to delete the event.

**Add participant** (same steps as for Add lecturer to event)

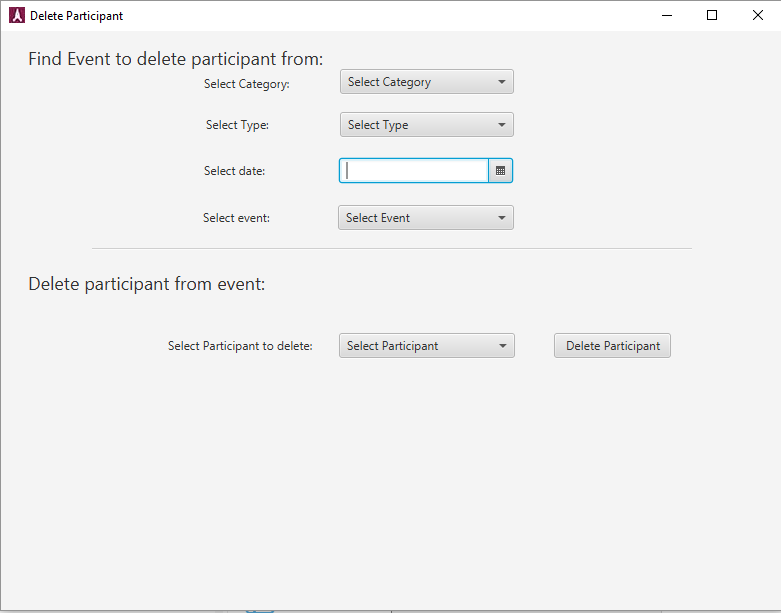


This interface is meant to add a participant to an existing event.

First the user has to fill out the necessary search criteria and select the event from the combo box. After the event is selected the user can start to add participants to the event.

The lower part of the interface is meant to add participants to the chosen event. To add a participant to an event the user first has to input the ID of the participant it wishes to add. After the user has entered the ID, the next step is to press the button “Get member”, after that the members name will be displayed in the text field below and the by pressing the button “Add member” the participant will be added to the event.

**Delete Participant** (same steps as for Delete lecturers from event)

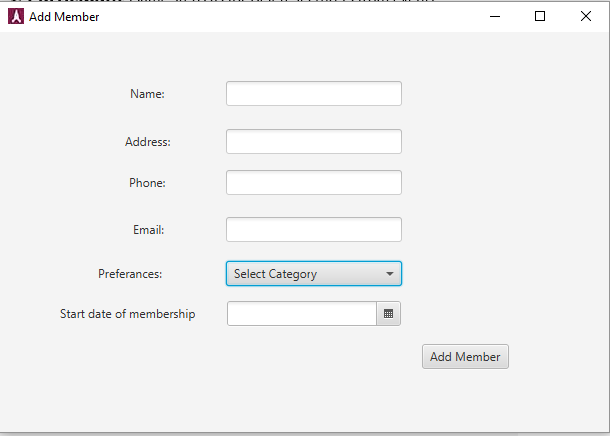


In this interface it is possible to delete a participant from a certain event.

By filling out the fields at the top part of the interface the user select the event from which to delete a participant from.

When the event is chosen the next step is to select the participant from the event. After the participant is selected the user has to press the button “Delete Participant” to remove the participant from the event.

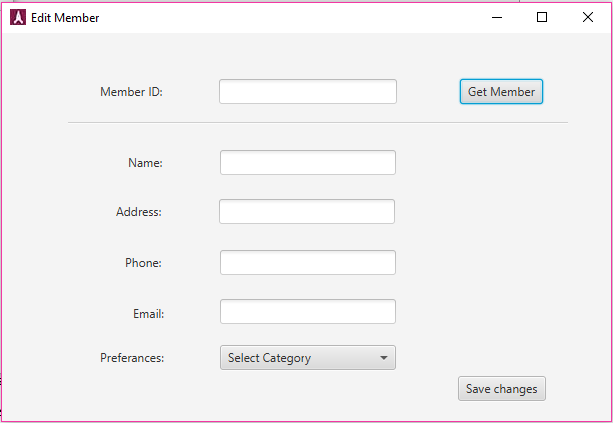
**Add a member** (Same steps for “Add sponsor” and “Add lecturer”)



In this interface it is possible to add a member to the systems.

The user has to fill out all of the required fields in order to create a new member. When all of the fields are filled out the next step is to press the button “Add member” to add the newly created member to the system.

**Edit a Member** (Same steps as for the “Edit sponsor” and “Edit Lecturer”)

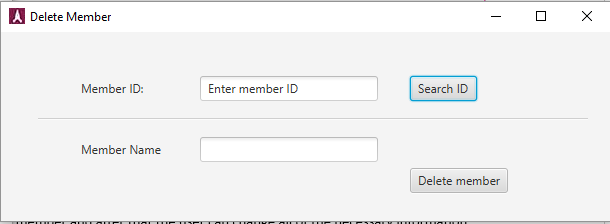


In this interface it is possible to edit information regarding to a particular member.

First the user has to find the member it wishes to edit. In the text field next to “Member ID” the user has to input the corresponding members ID. After that the button “Get member” has to be pressed in order to find the member in question.

After the member is found, all of the required text field will be filled out with information regarding the member and after that the user can change all of the necessary information. After the changes are made the user has to press the button “Save changes” in order to save all of the edited information to the system.

**Delete Member** (Same steps as for “Delete sponsor” and “delete lecturer”)

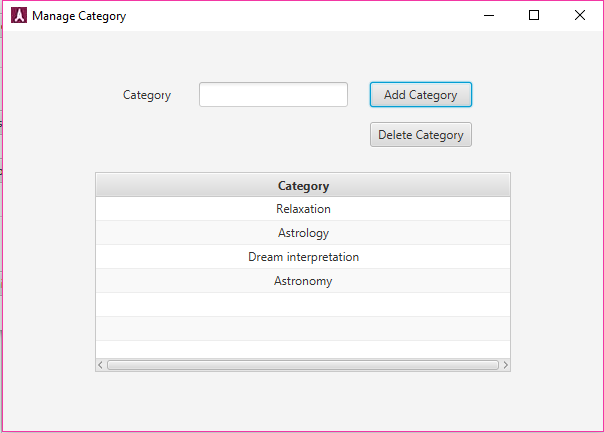


In this interface the user can delete a member from the system.

First the user has to input the ID of the member it wishes to delete. After entering the ID, the user has to press the button “Search ID” to find the member in question.

After the member has been found the user the presses the button “Delete member” to remove the member for the system.

**Manage Category**

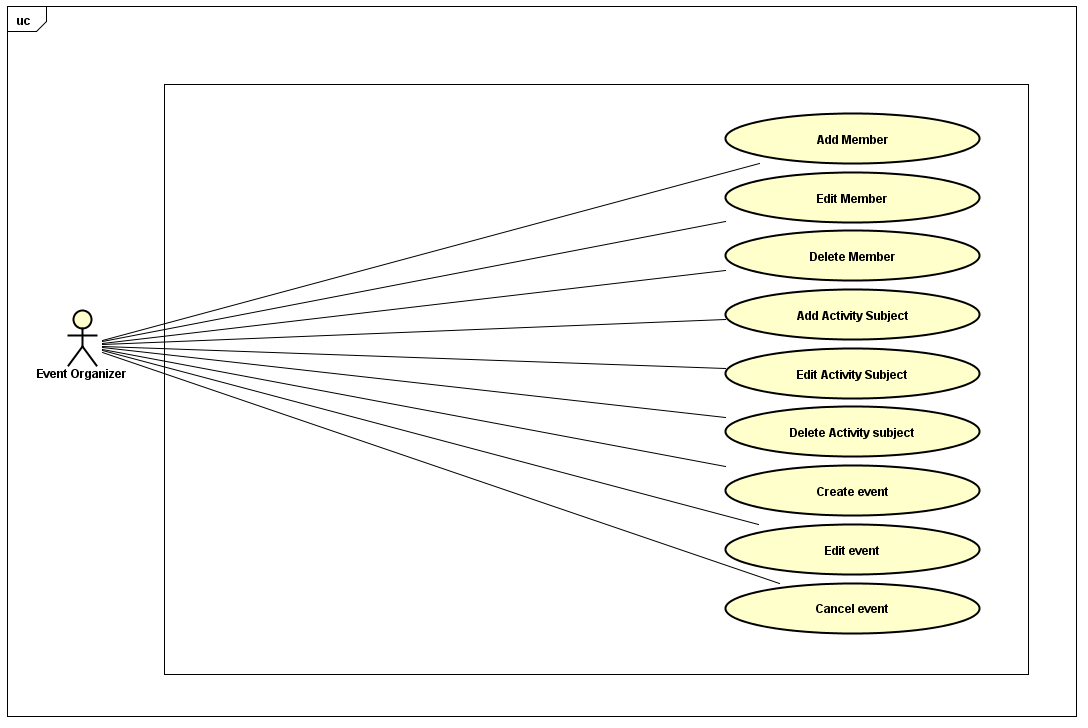


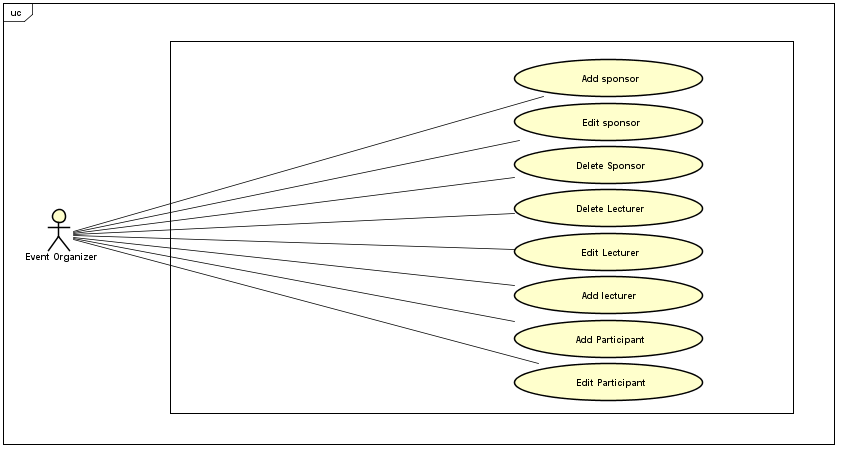
In this interface it is possible for the user to add/delete a category to/from the system.

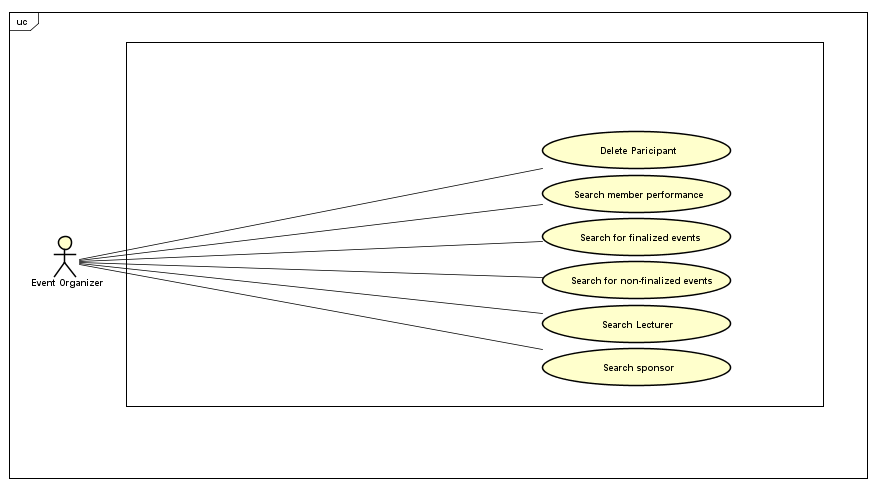
The user has to enter the title of the category and the press either the button “Add category” or “Delete category”.

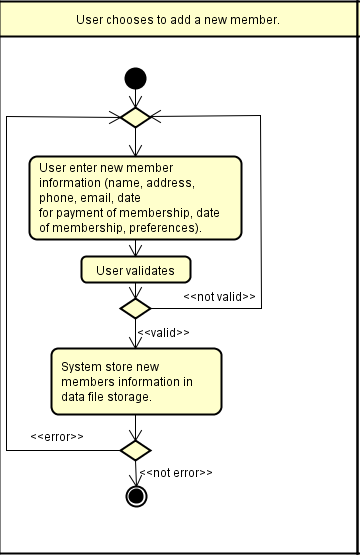
**Appendix**

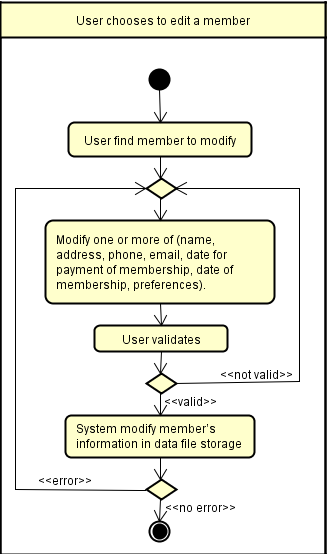
Appendix Diagrams

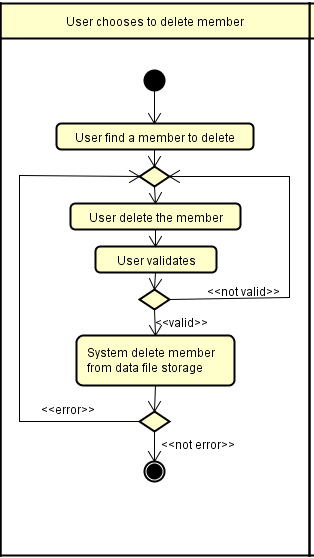


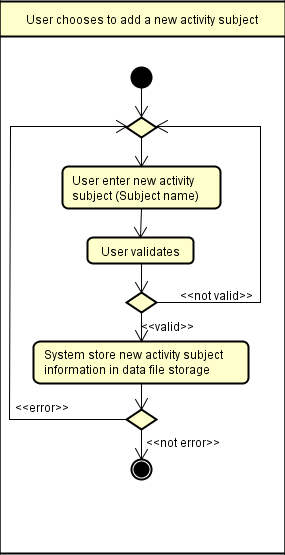


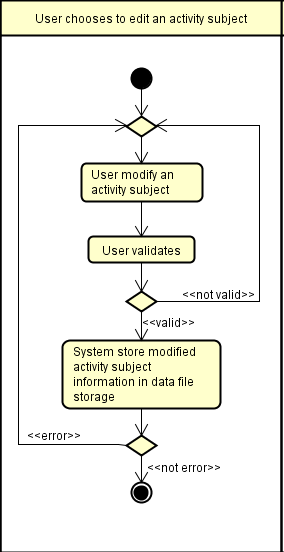


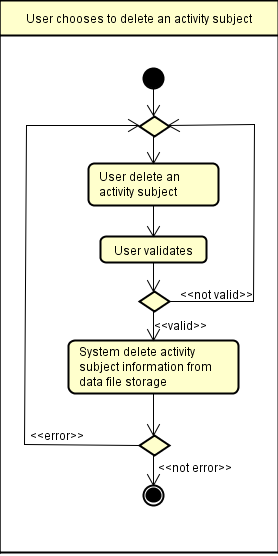


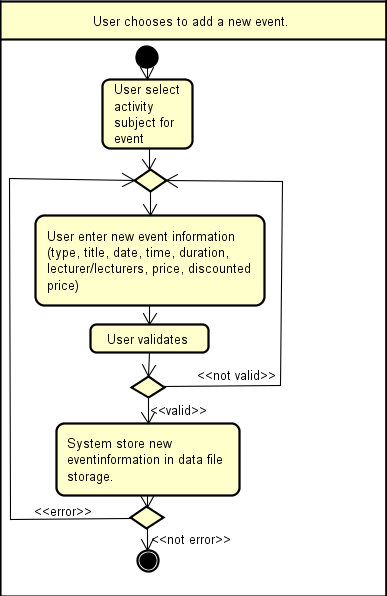


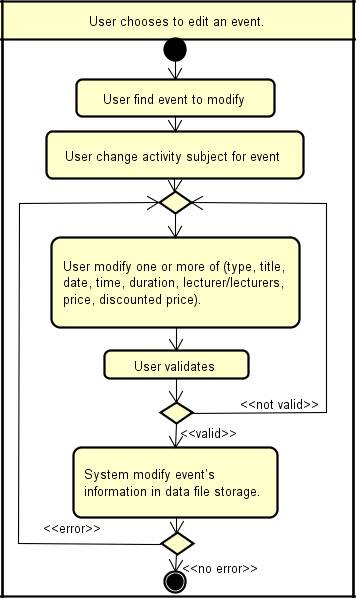


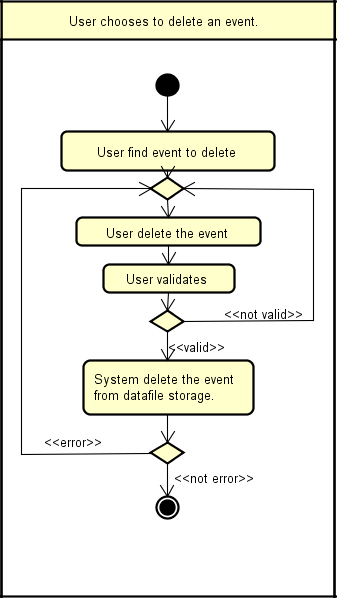


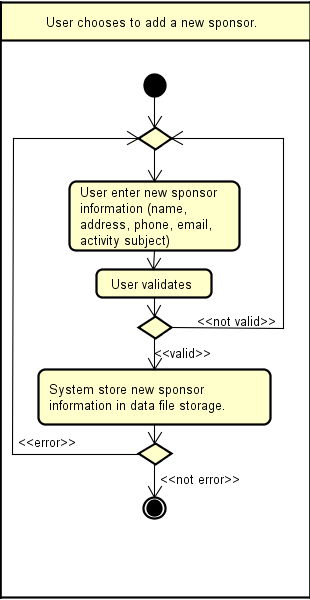


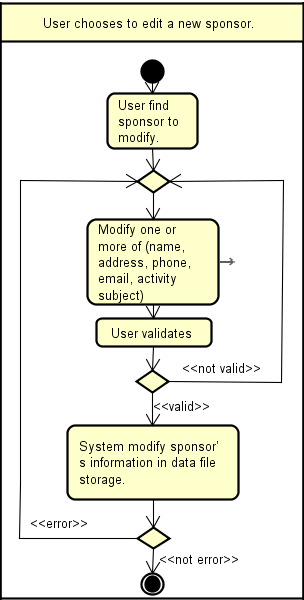


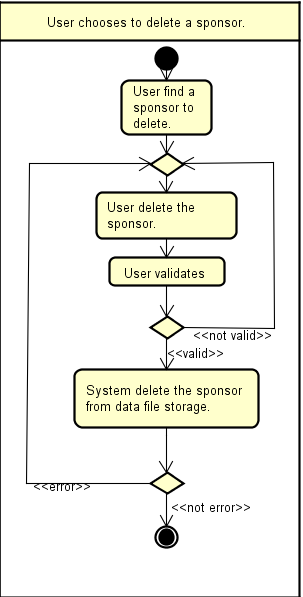


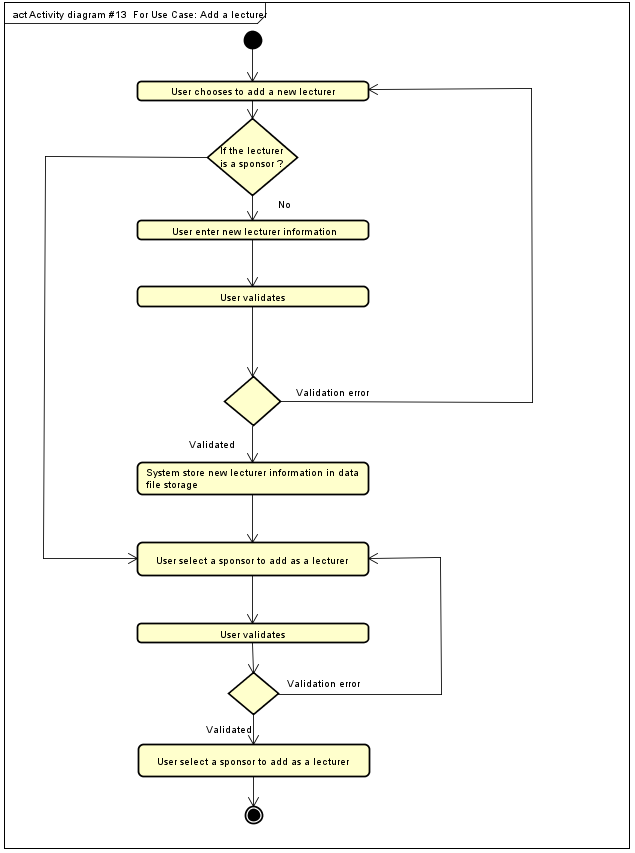


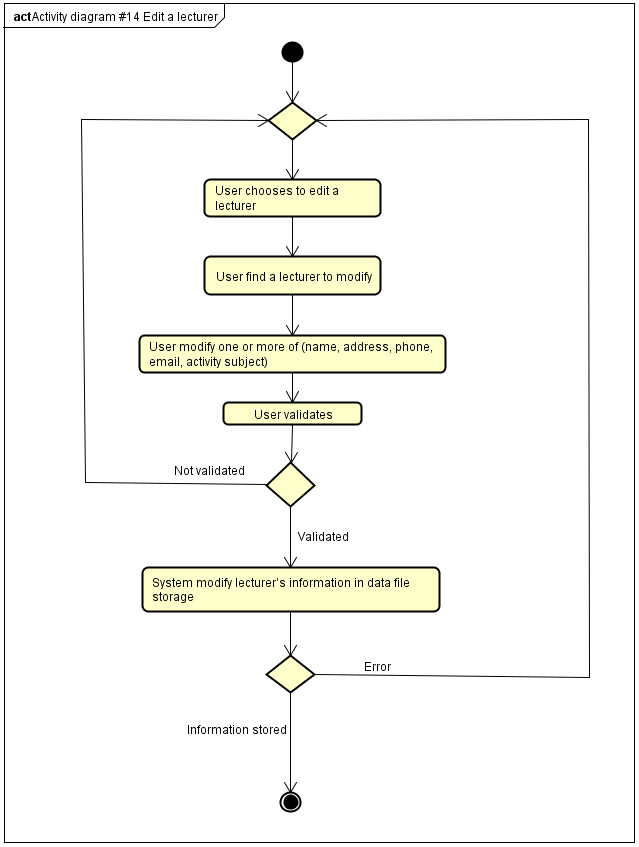


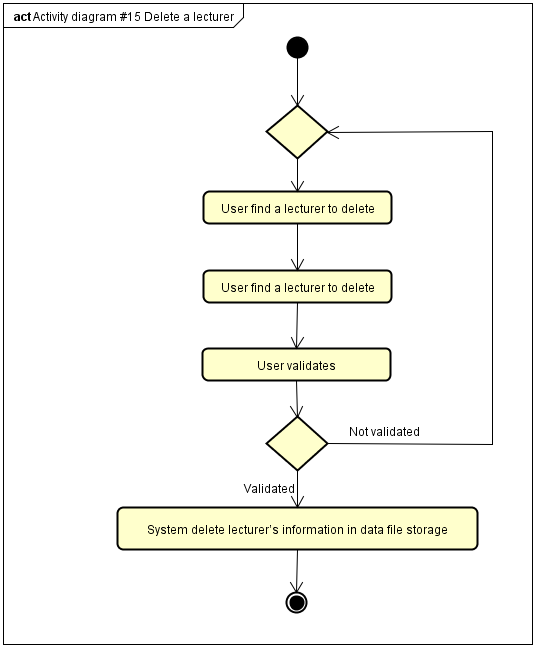


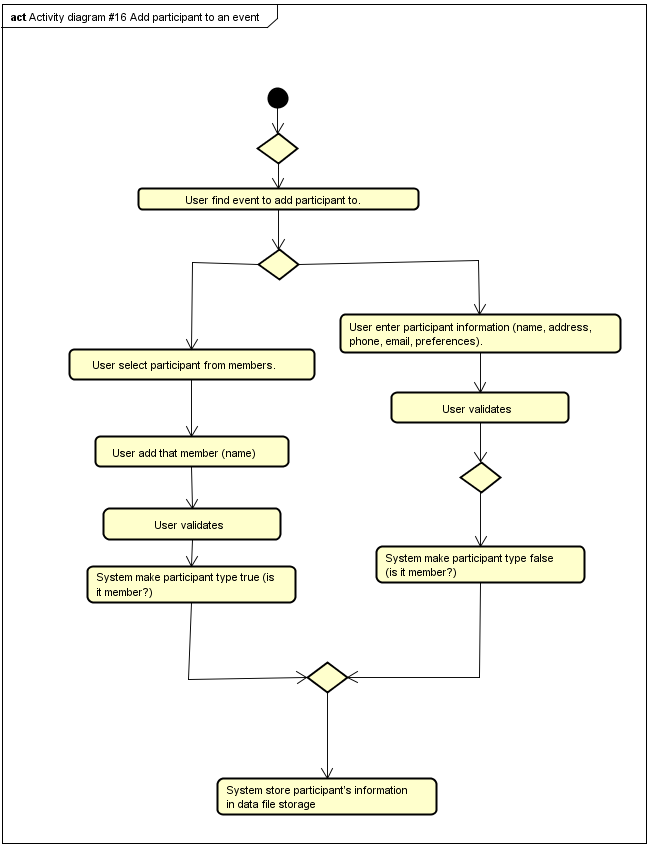


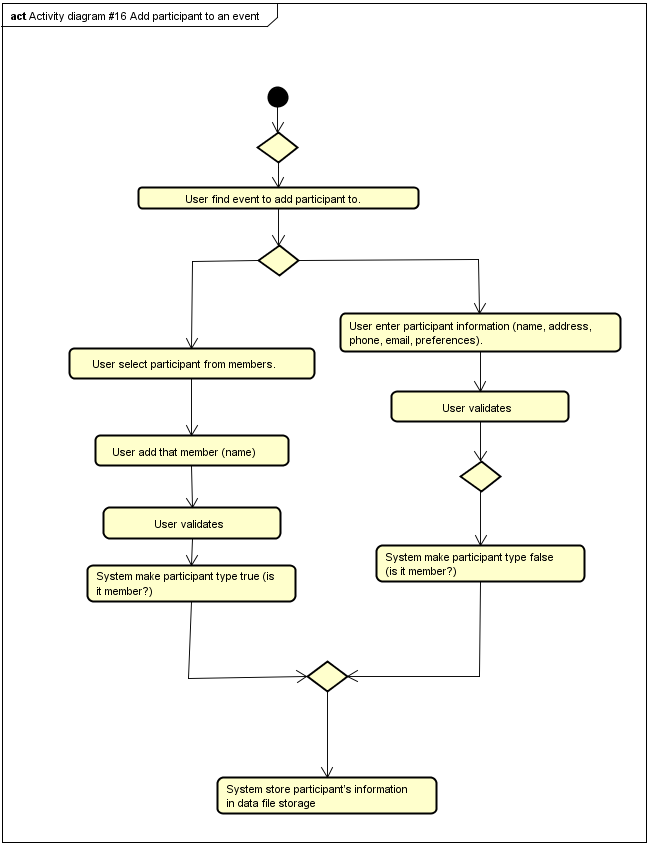


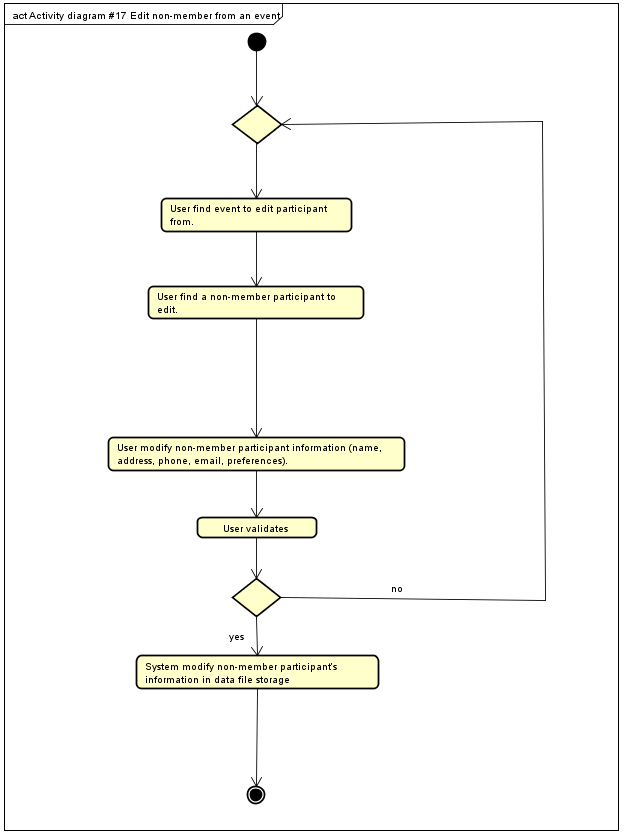


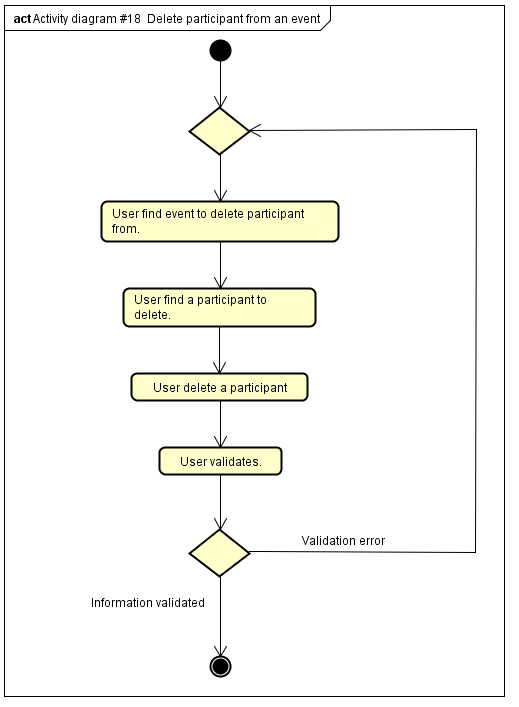


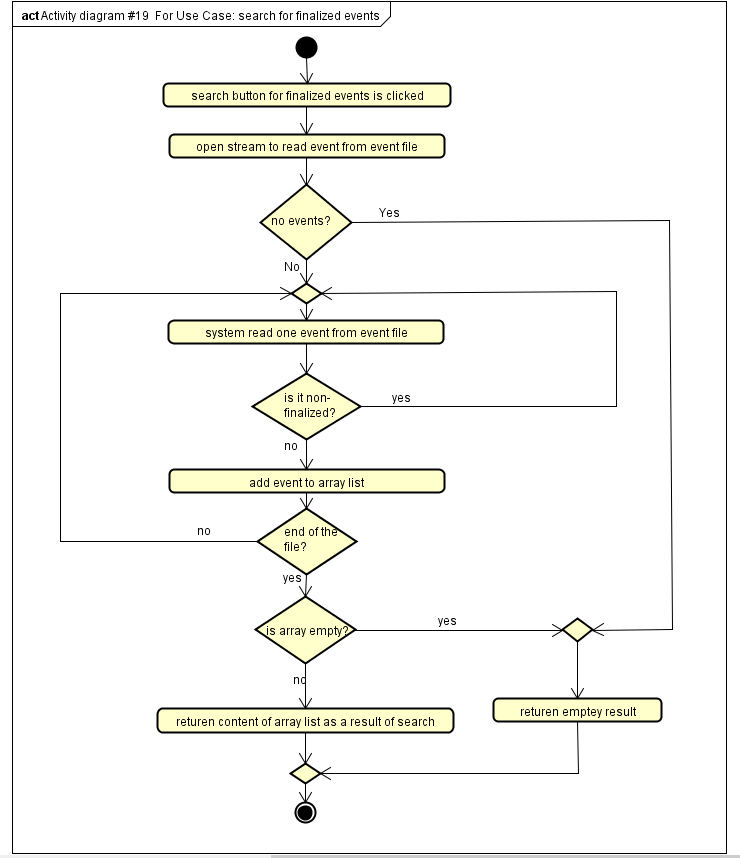


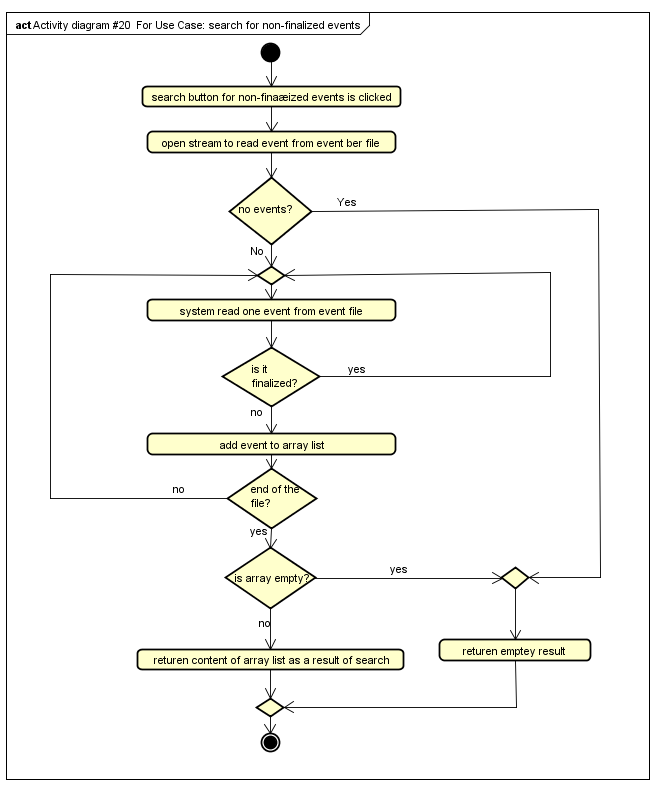


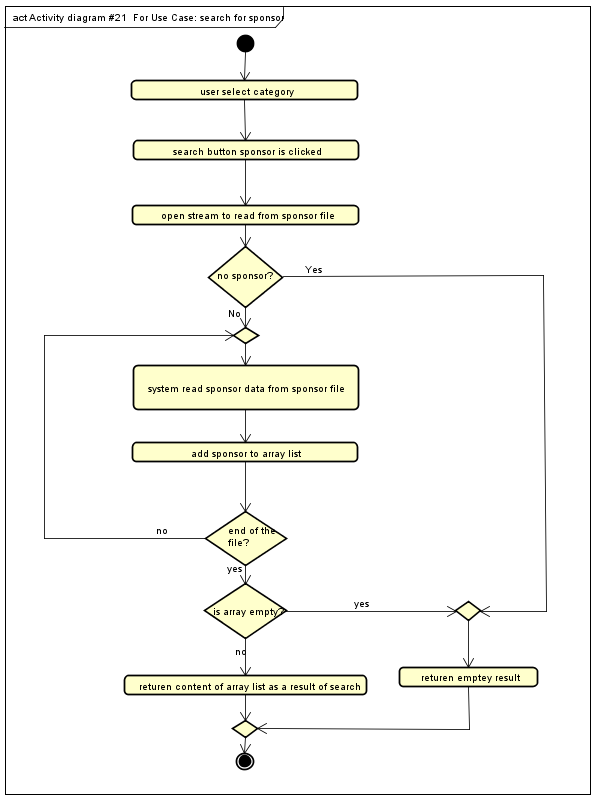


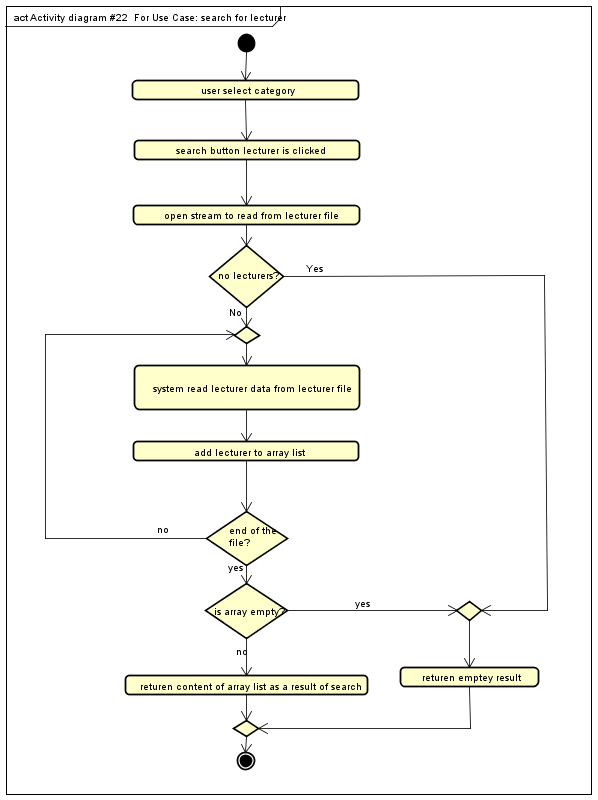


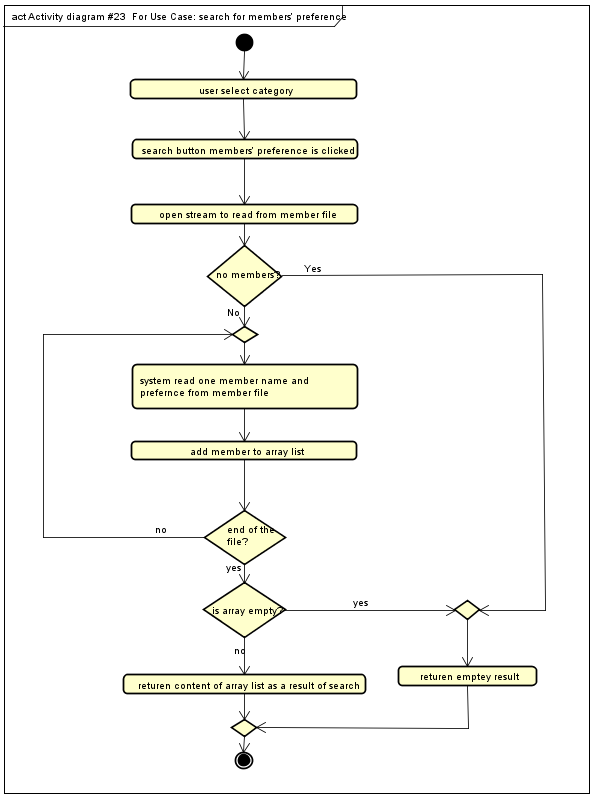


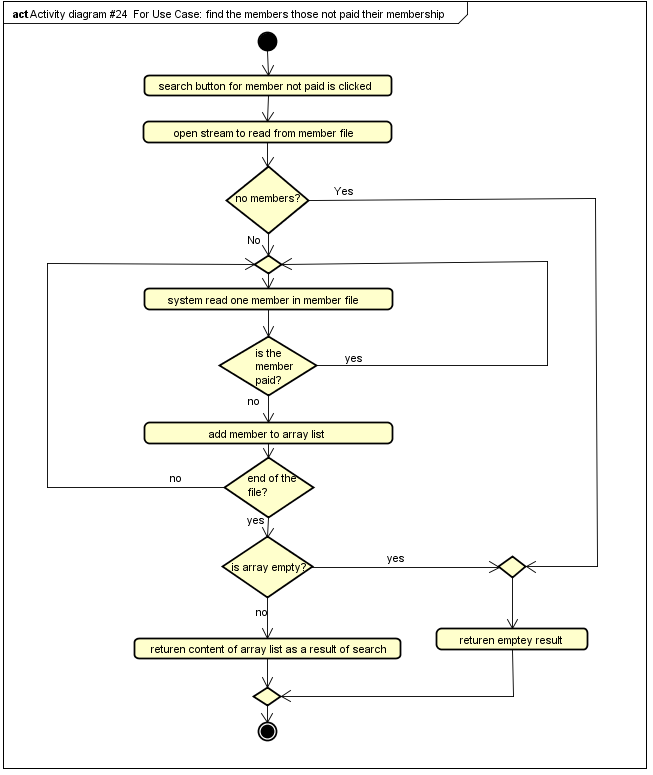












|  |  |
| --- | --- |
| ID: | #1 |
| Title: | Add a member |
| Description: | The user will add a member |
| Primary Actor: | User |
| Preconditions: | None |
| Postconditions: | A is member added |
| Main  Success Scenario: | 1. User chooses to add a new member. 2. User enter new member information (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, |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| 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. |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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, |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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 |