ICT Engineering









Semester Project Report

Group 2

Claudiu Rediu-266129

Boris Sidlo-251341

Eduard Costea-266078

Michal Ciebien-266908

Peipei Su 266128

Supervisors:

Michael Viuff

Mona Wendel Andersen

Date: 19. 12. 2017



# Contents

[1 Requirements 5](#_Toc501279026)

[1.1 Functional Requirements 5](#_Toc501279027)

[1.2 Non-Functional Requirements 6](#_Toc501279028)

[2 Analysis 7](#_Toc501279029)

[3 Design 17](#_Toc501279030)

[4 Implementation 26](#_Toc501279031)

[5 Test 28](#_Toc501279032)

[6 Results and Discussion 29](#_Toc501279033)

[7 Conclusion 30](#_Toc501279034)

[8 Project future 31](#_Toc501279035)

[9 Sources 32](#_Toc501279036)

[10 List of Appendixes 33](#_Toc501279037)

**Abstract**

*Vipassanā is a non-profit center in Horsens, Denmark. It provides events, which are based on Buddhism tradition. The system can be used to manage events(i.e.create events,remove events,find lecturers and find sponsors), manage members or search events which are finalized.*

*Vipassanā needs a system, which is made in Java, to store information regarding members, lecturers, sponsors and events. In this system, users can input basic information regarding events and members, the GUI containing tabs, text fields, buttons, check boxes, and text areas. User can choose one of two tabs(members and events) to store information.*

*In the members tab, there are two main tabs inside.One is the tab for managing members,which can be used to add members by storing his or her name, email, address, telephone number, year of payment, category, and the register date. The other is the tab for listing members, which can list members both by category or payment condition of the annual fee.*

*In the events tab, there are two main tabs inside. One is the tab for managing events, which can be used to create or modify events by storing its basic information, name of lecturers and sponsors, starting day, and feedback from members. Another is the tab of searching events and listing lecturers and sponsors by their category, which can also be used to search for events by category or finalization state.*

**Introduction**

Vipassanā - Insight Awareness is a non–profit centre providing spirituals events which are based on Buddhism traditions. Focusing on meditation, specially finding the - Insight Awareness, but also providing lots of events. Customers can find all sort of activities to take part in like interpretations, healing, astrology, reincarnation, Karma, alternative health care and much more (Interview for Case SEP,2017).

In the present there are over 360 million followers of Buddhism worldwide and over a million American Buddhists today. Buddhist concepts have also been influential on western culture in general, particularly in the areas of meditation and nonviolence (religionfacts.com, 2017) 1. There is a big community that needs to have at its disposal the tools to practice their religion in centres like Vipassanā - Insight Awareness. Buddhism is also the fastest growing religion in Western societies in terms of new converts (J. Perera, 2008) 2. Therefore the non-profit organization looks forward to expanding their services to as many people as they can.

The centre is already working on sending out newsletters and articles to its members as well as on balancing their expenses. But what they want is to further improve their management to better serve the people that are interested in their services.

Even though they managed up until now, Vipassanā - Insight Awareness needs a system to advance their communication with their customer base so that they can better deliver what they offer. By improving the management system of the organization, it will become more attractive to the followers of Buddhism as well as will become more efficient.

The delimitations are a result of the requests that the client had, together with the current limitations that will be experienced building the product. In the project, it will not be expected for the website to have a direct connection to the system made in JAVA, as it is not requested by the customer. The system made in JAVA will not be using a database or the internet to receive data. The main way of interacting with it is direct input from the user and secondary file storage. The system will not be in its final state as it needs to be open to further improvements in the future.

In the Requirements section it will be presented what the system will be capable of and what is expected from it to do.

# Requirements

The user is the only actor interacting with the system and is responsible of directing in what action should the system perform and introducing new information that is not present already in files that are part of the secondary file storage of the system.

## Functional Requirements

1. The user should be able to create an event.
2. The user should be able to modify information regarding an event.
3. The system should be able to store an event’s name, number of participants, number of members, length, discount, finalization state, type of event, if it will have a vegan food, location, starting date, feedback, lecturer and a sponsor.
4. The user should be able to remove a specific event from the system.
5. The system should be able to list all the events and specific events by their category or finalization state.
6. The system should be able to store lecturer’s name, sponsorship state, email address and a category.
7. The system should be able to store sponsor’s name, email address and a category.
8. The system should be able to provide a list of sponsors and lecturers sorted by their category.
9. The user should be able to create a member.
10. The user should be able to edit information regarding a member.
11. The system should be able to store member’s name, e-mail address, address, phone number, year of last fee payment, category and registration date.
12. The system should be able to list out the name and email of all the members.
13. The system should be able to list out the name and email of members by their given preference.
14. The system should be able to list out the name, e-mail and the year of the last payment of members that did not pay their annual fee.
15. The user should be able to remove a specific member from the system.

## Non-Functional Requirements

1. The system must be implemented in JAVA.
2. The system must be compatible with Microsoft Windows 7,8,10.
3. The system must answer in 2 seconds 95% of the time.
4. The system must process the info in maximum 3 seconds 85% of time.
5. The system must use files for secondary storage only.
6. The system must be usability tested by end users.
7. The system must retain the data of the current process for at least 5 minutes.

# Analysis

**2.1 Use Case Modelling**

In this part of analysis, we focus on use case modelling for which we used Astah.

**Use Case Diagram**

From the use case diagram, you can see how the system can be used by the user. For the successful running of VIPASSANA, employees must be able to **Manage events, List lecturer/sponsor by category, List members, Manage members** and **Search for events.**

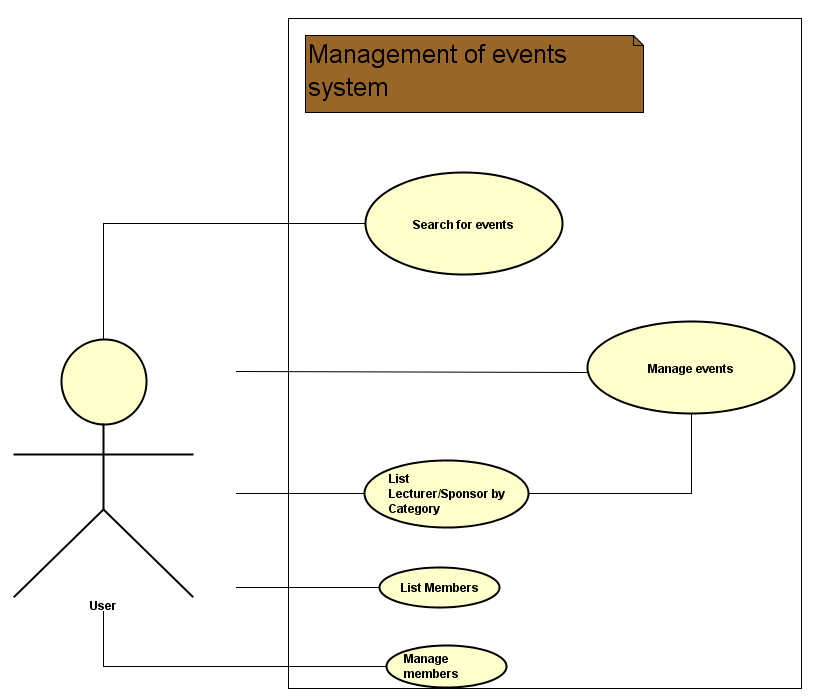


Figure.1 Use Case Diagram

**Manage events**

User can use the system to create, modify aspects of an event or remove it.

**List lecturer/sponsor by category**

User can use the system to list the lecturer and sponsors by their category.

**Manage members**

User can use the system to create a member, modify aspects of it or remove it.

**Search for events**

User can use the system to search for events based on category and state of finalization.

**List members**

User can use the system to return all the member's emails, returns the members by their preference or if they did not pay the annual fee.

**2.2 Use Case Description**

“Manage Events” and “List Members” are the most important use cases because they fulfill the most of the requirements regarding the system. “Manage Events” is used to create events, modify or remove them which is one of the main appliances of the system. “List Members” is used to return lists of members respecting the requirement of having a list of members that did not pay the annual fee.

Appendix.1 Use Case Description

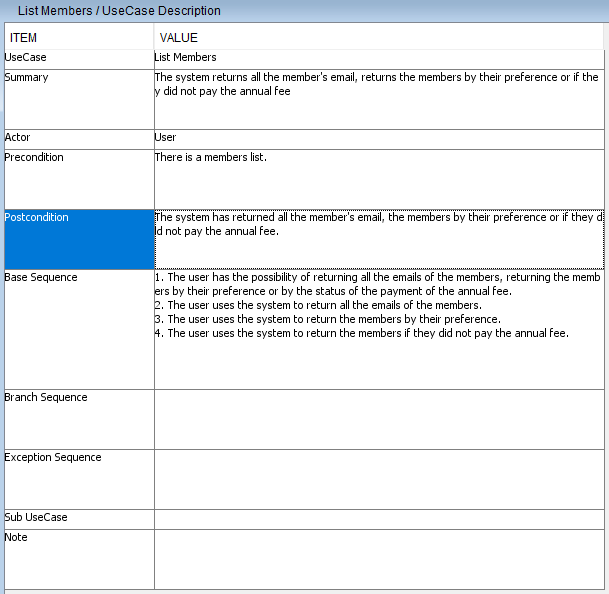


Table.1 List Members

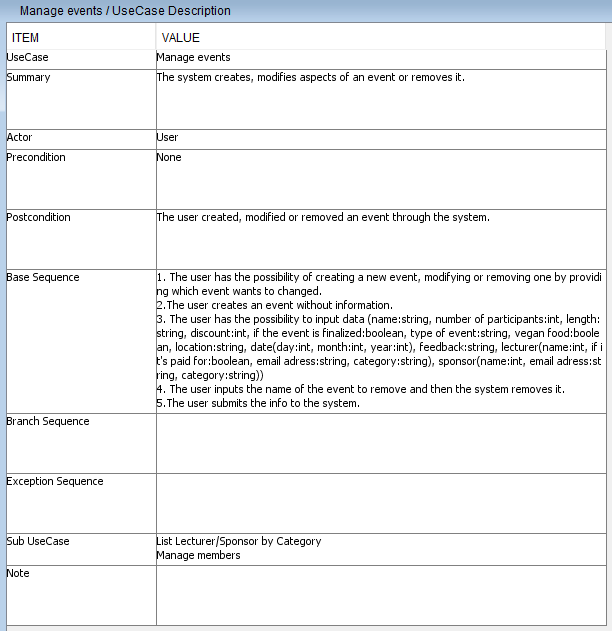


Table.2 Manage Events

**2.3 Actor descriptions**

The users are the only actors who are employed by Vipassana.They interact with the system, and are responsible of directing what action should the system perform and introducing new information that is not present already in the files that are part of the secondary file storage of the system.

**2.4 Activity diagram**

By using the Activity Diagrams you can see various paths that the user can take in the system GUI in order to achieve particular goals. It shows exactly what choices are availiable for the user and what does the system do when they are followed.

In the ”List Lecturer/Sponsor by Category”(Figure 1.) diagram user can interact with the system by choosing between listing lecturers or sponsor by their category, which is the first input by the user. The system then saves either of the lists into an external text file.

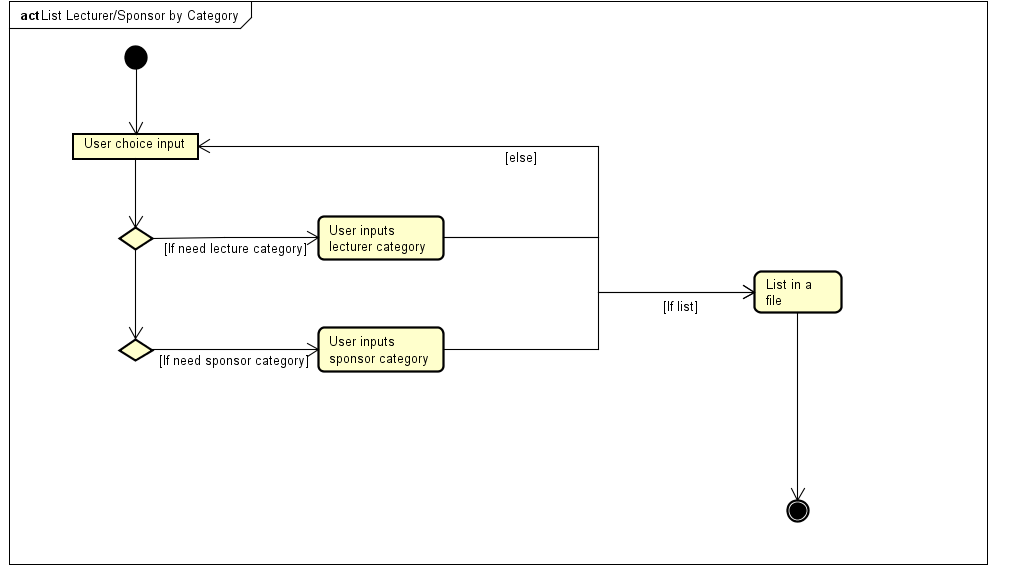


Figure 1. *“List Lecturer/Sponsor by Category” Activity Diagram*

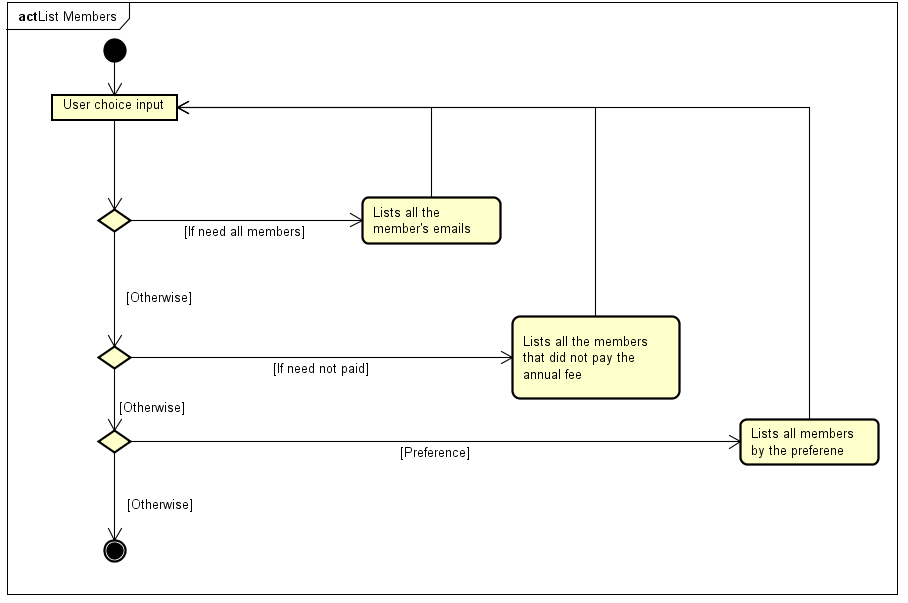
In the ”List Members”[Figure 2.] diagram, the user can choose between listing members by emails, listing them by their current annual fee payment state, listing them by their preference or quiting the List Members GUI part. If user chooses one of the listing parts, after the listing part ends, the interface is again ready to be used. 

Figure 2. *”List Members” Activity Diagram*

In the ”Manage events”[Figure 3. Activity Diagram] activity diagram, the user has various paths to follow based on the goal. The user can here choose between creating an event, listing lecturers or sponsors by their category, modifying an event or searching for an event. If the user chooses to create an event, the event is then created with default. If the choice is to list lecturers or sponsor by their category, the user then inputs the category and then the system list the lecturer and sponsors.. If the user chooses to modify event, the user first neeeds to choose the event, after that, input the information and then event is ready to be overwriten and modified. If the user chooses to search for an event, then the following paths showed in (Figure 4.”Search Events” Activity Diagram) need to be followed.

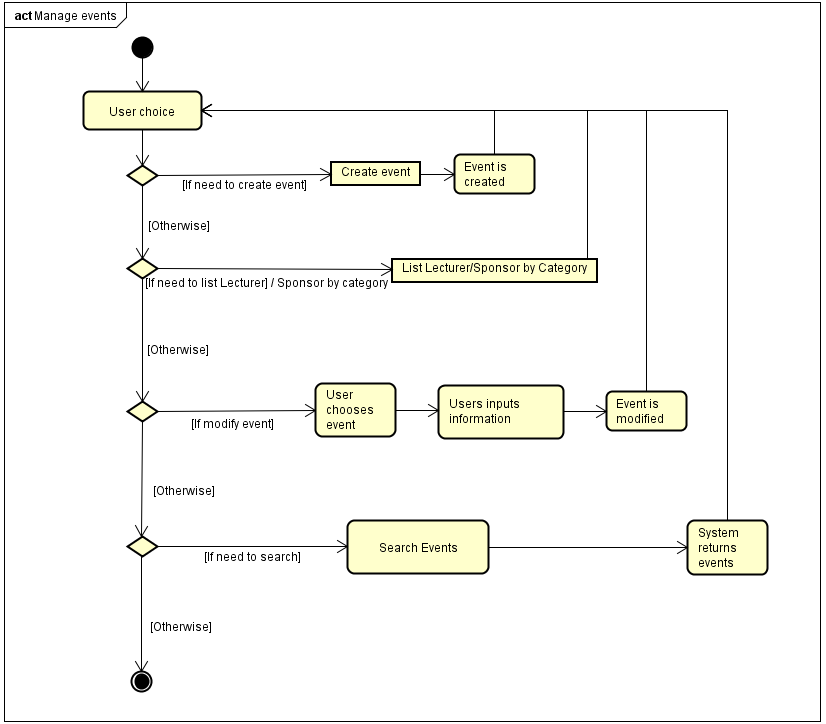


Figure 3. *”Manage events” Activity Diagram*

In the ”Search Events” (Figure 4. ”Search Events” Activity Diagram) diagram, the user can here choose between leaving with no action, receiving all the events or receiving events by their finalization state or the category. After any listing, the user is able make a choice again.

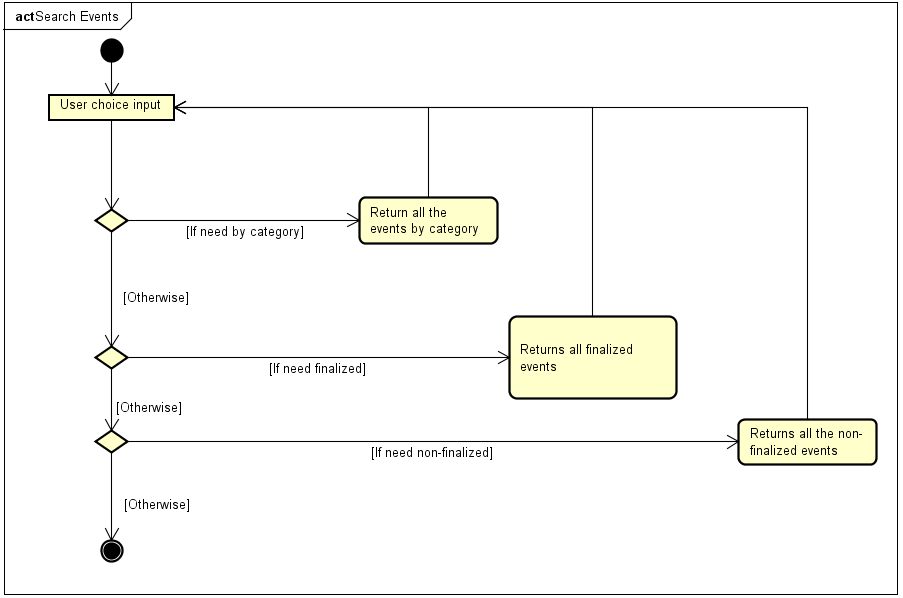
****

Figure 4. *”Search Events” Activity Diagram*

In the ”Manage members”(Figure 5. ”Manage members” Activity Diagram) the user can choose between creating a new member, modifying a member, removing a member or searching for a member. If the user chooses to create a new member, the system creates a new member with information given by the user. If the user chooses to modify the member, system recieves input from the user in order to find a particular member, then the user can modify the member’s details, after which the system saves the changes. If the user chooses to remove a member, the system receives input from the user in order to find a particular member, then removes this member from the list of members. If the user wants to find a member, the system receives input from the user in order to find a particular member.

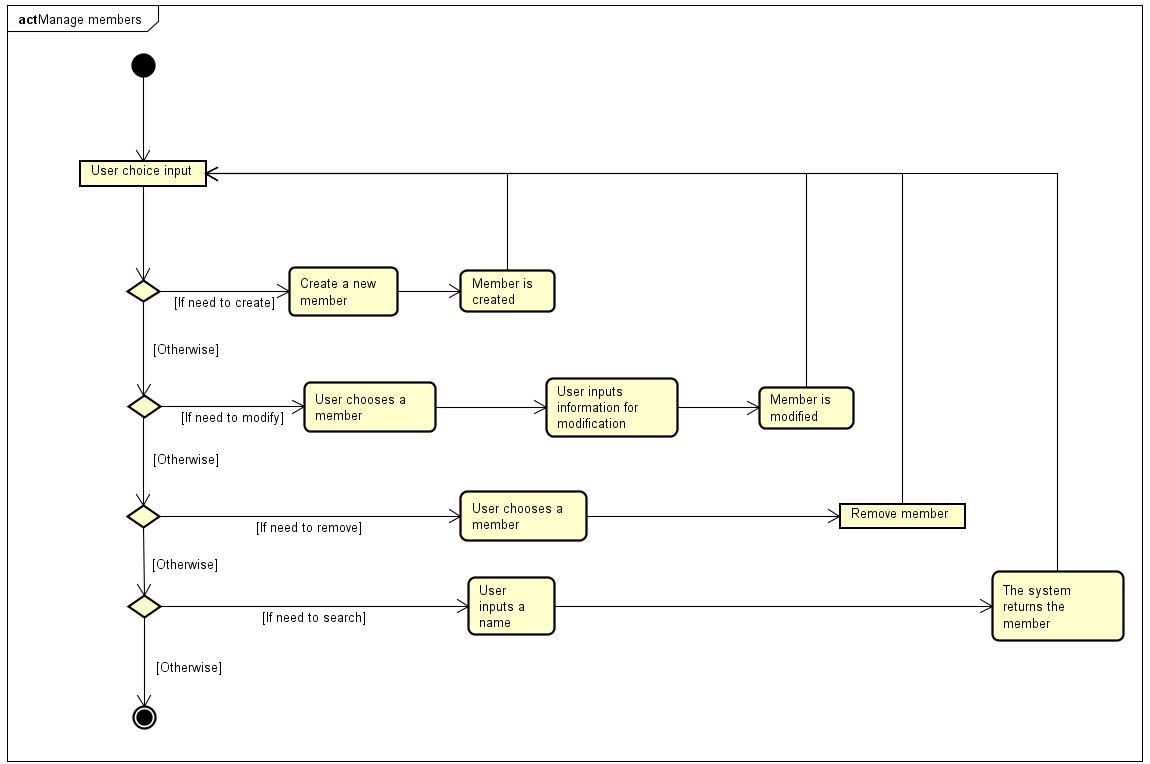
****

Figure 5. *”Manage members” Activity Diagram*

**2.5 Analysis diagram**

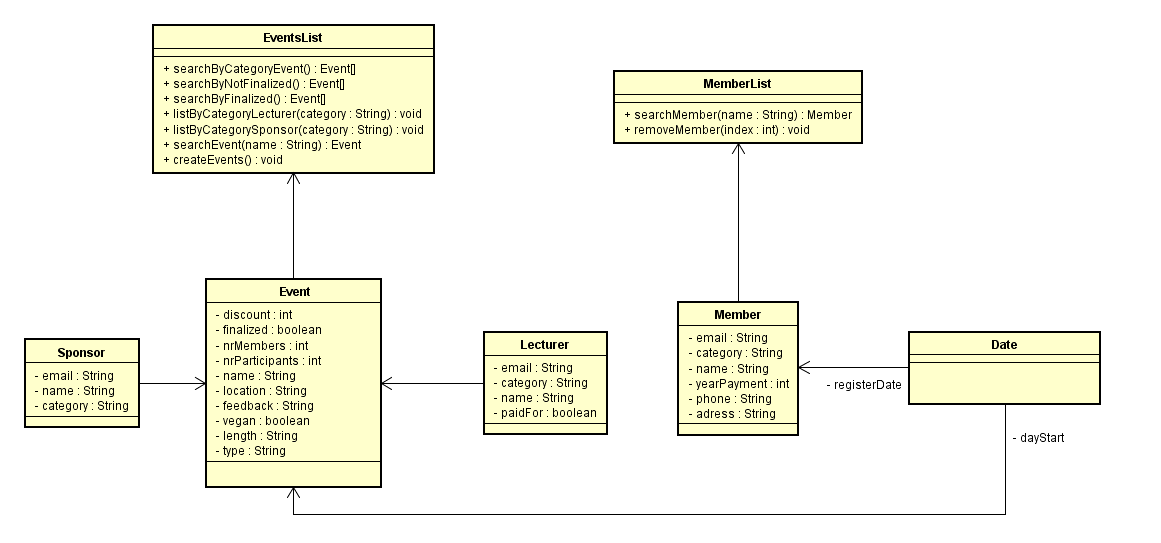


Figure.1 Analysis diagram

The analysis diagram was built considering the use cases. The methods that are the most important are handled by EventList and MemberList. Both Event and Member use the Date class because they Event has a start date and the Member a registration date, but only Event has the Sponsor and Lecturer. In Event, Member, Lecturer and Sponsor the fields are present as they are crucial information that is needed by Vipassana.

# 3 Design

Design Class Diagram

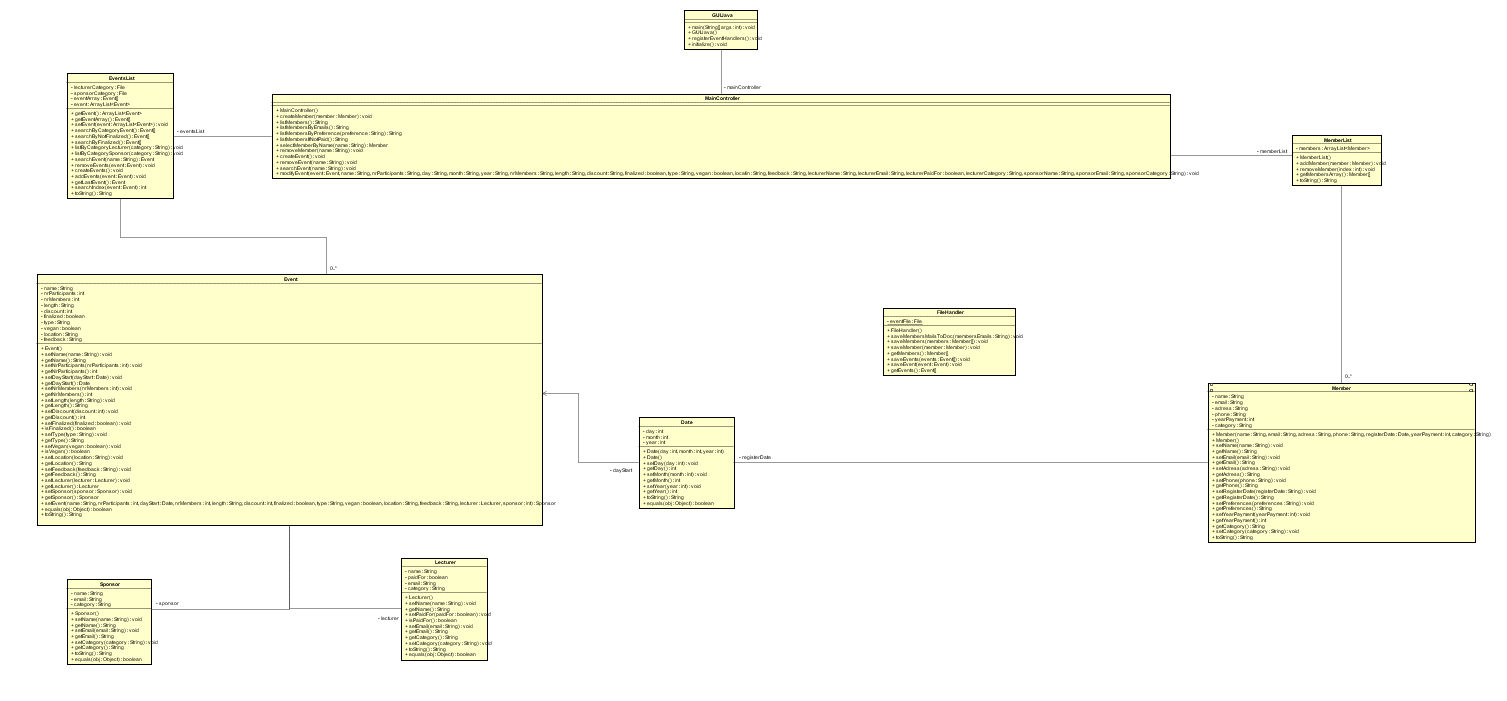


Figure.1 Design Diagram

The most basic class in the system is “Date”,in this class,users can store information of date(day,month,year).

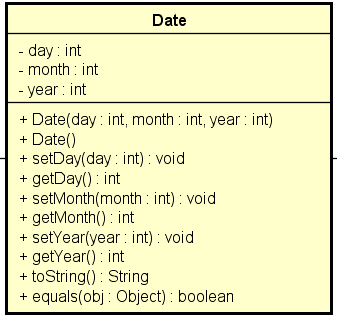


Figure.2 Date

In “Lecturer” and “Sponsor” classes,users can store names,emails and categories for them,but for lecturers,there is a function “paidFor” to check their payment condition.

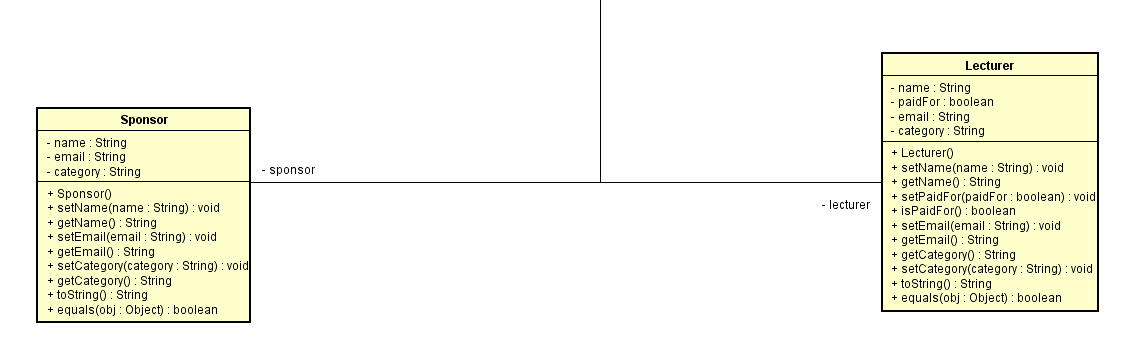


Figure.3 Sponsor and Lecturer

In “Event” class,methods in “Lecturer”,”Sponsor” and “Date” will be used.Users can store names,starting day,amount of members and participants,length,events’ type,discount,location,feedback from members.Users can also check finalization and whether the member is vegan or not.

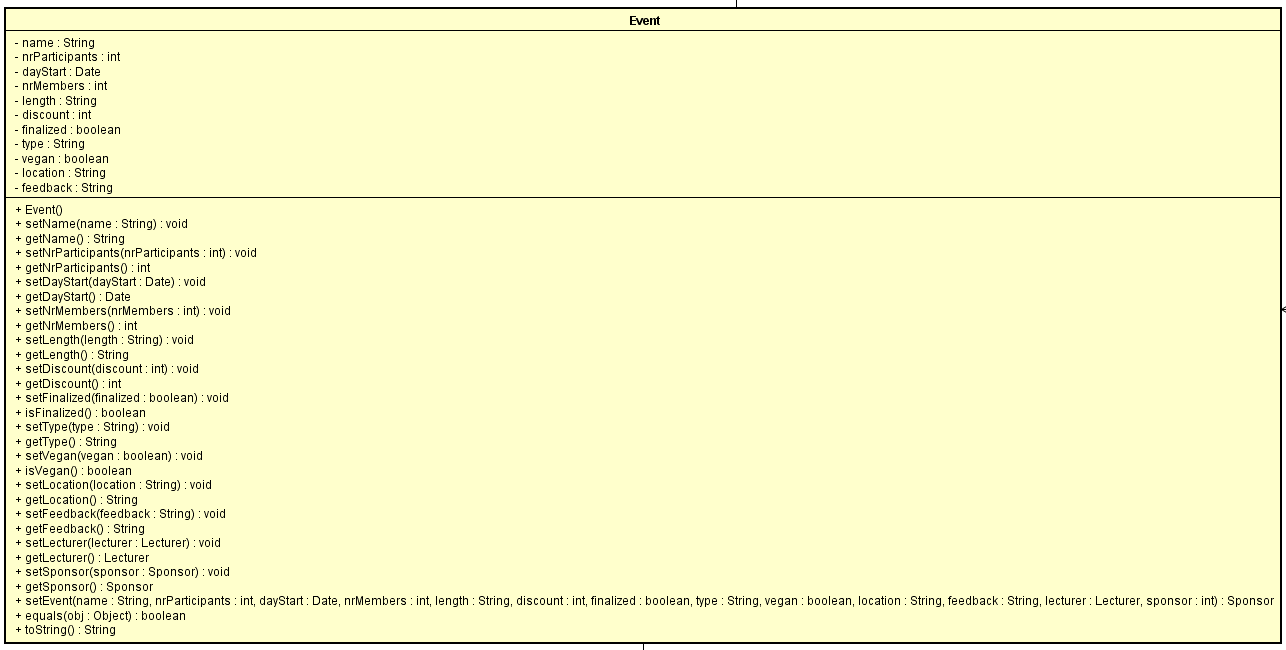


Figure.4 Event

In “Member” class,users can store names,emails,phone numbers,address,register date,year of payment,and the category.

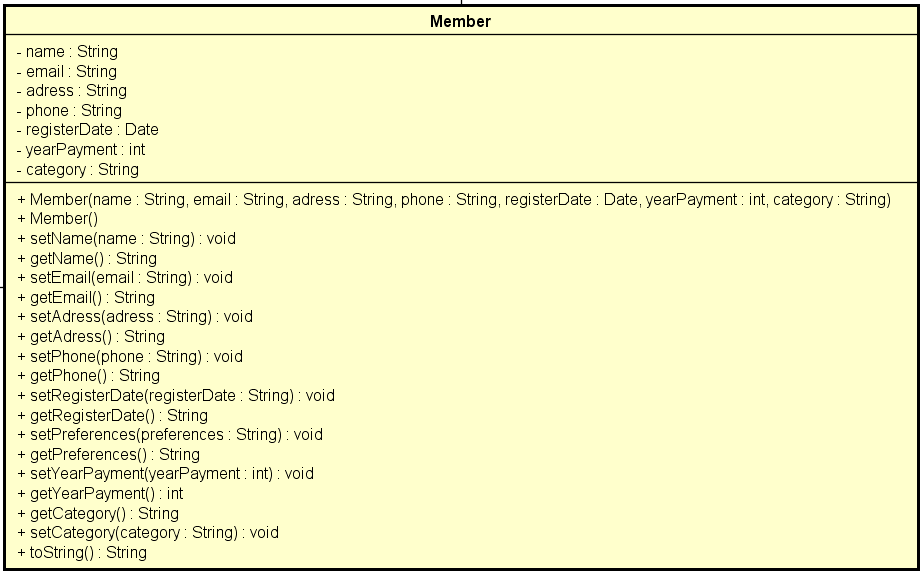


Figure.5 Member

In “FileHandler” class,users can create two files,which are file of events and file of members respectively.Users can write and read information in this two files.

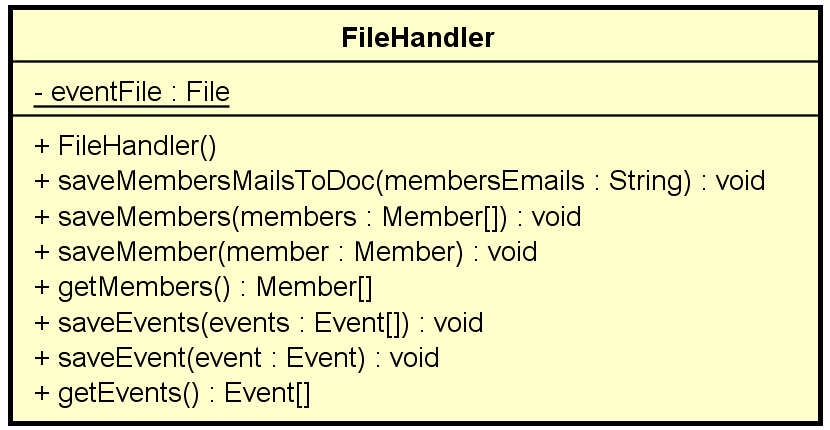


Figure.6 File Handler

In “EventList” class,users can add events,remove events,search events with requirements(names,category and finalization),and read information from the file of events by using the method in “FileHandler”class.

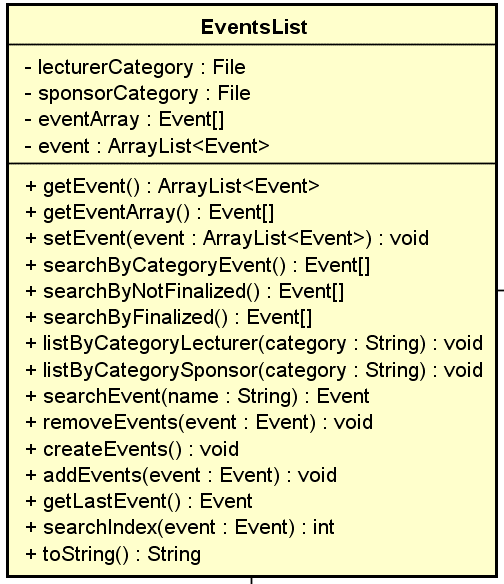


Figure.7 Events List

In “MemberList” class,users can add members,remove members,search events with specification(name,emails,payment condition and preference),and read information from the file of members by using the method in “FileHandler” class.

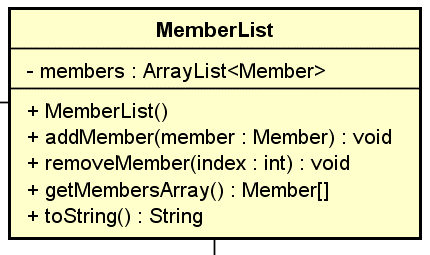


Figure.8 Member List

The sequence diagram describes the how the system creates an event and each object interacts with each other to bring the desired result. The method createEvent() in MainController calls createEvents to add an event to the arraylist of events. In Event the event object is created by giving to each field a value. Also there the date, lecturer and sponsor object are created and sent back to the contructor of Event.

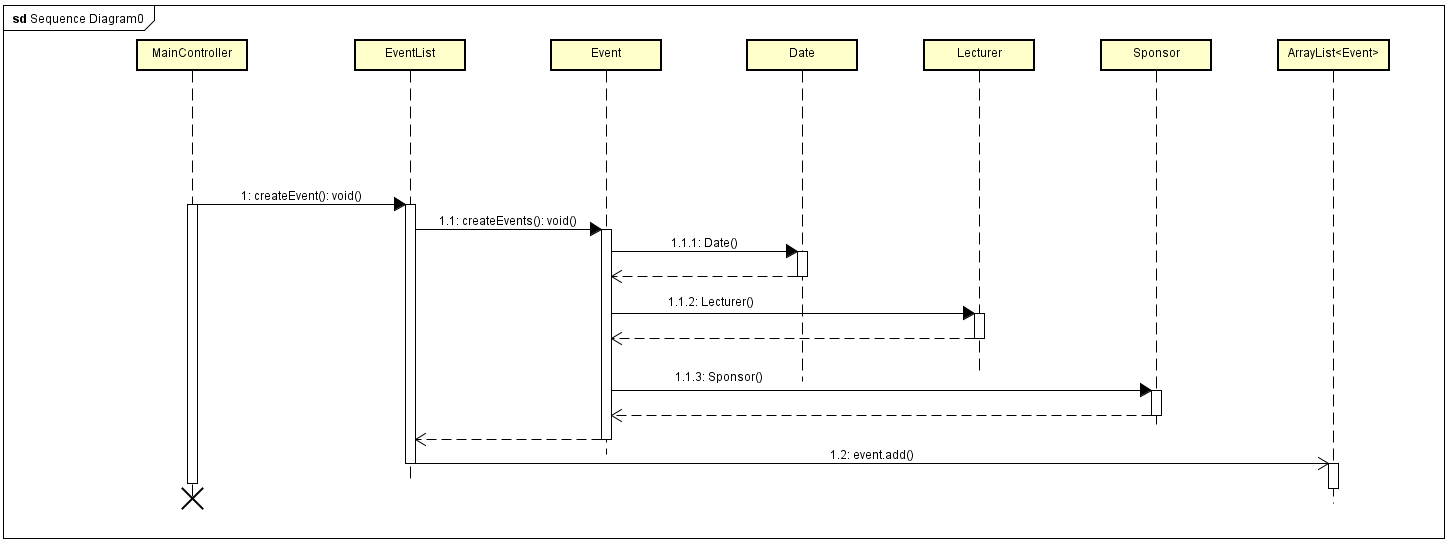


Figure.9 Sequence Diagram

The GUI has four tabs inside which all the elements are put. It is built it in this way so that access to everything that the system does is easier.

The Manage Event tab is built with four buttons with action listeners, text fields and check boxes for the fields of an event. When the Create Event button is pressed all the fields are filled with the values that are given in the system, the reason being that the user would know where he needs to input data.When submitting the changes made to the event, the original name needs to be inputted in the textfield next to the Search button so the system would know what event to change.

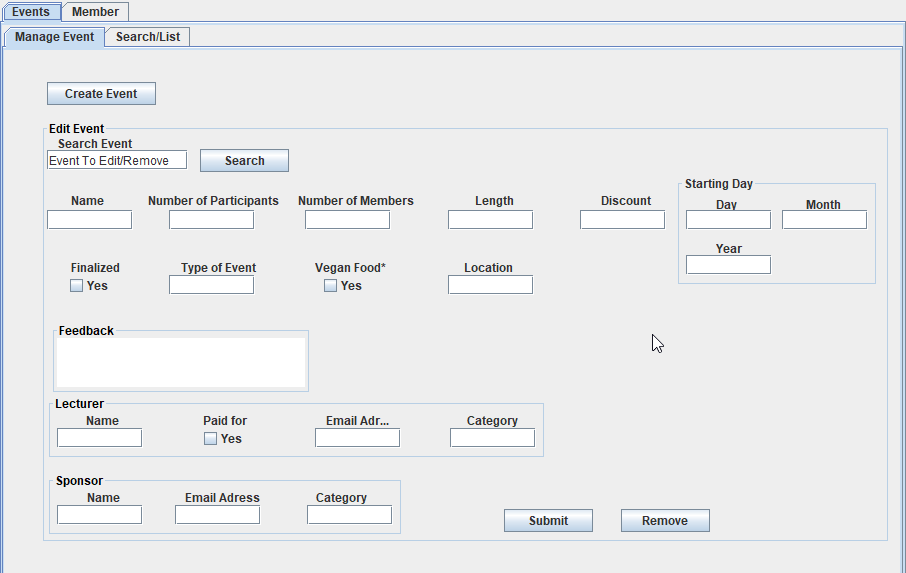


Figure.10 GUI Manage Event tab

The GUI has an textArea object so all the information outputted when events are searched will be there, the reason for this is that the user would search for certains events for which it would not have the name yet. When the List button is pressed a file is created with the information of the lecturers and sponsors by their category. The List button does not show anything in the textArea because it was considered not relevant in fulfilling the requirement of having a list of lecture and sponsors.

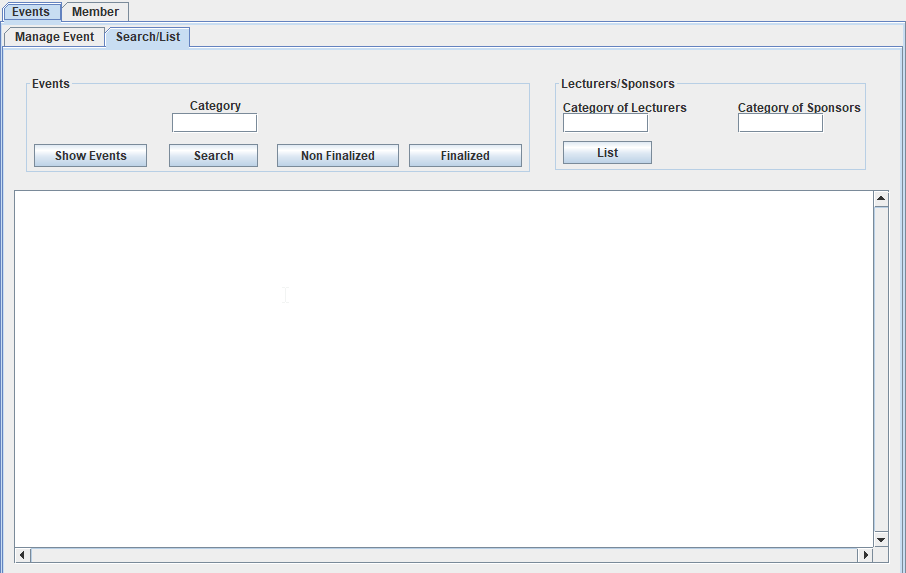


Figure.11 GUI Search/List tab

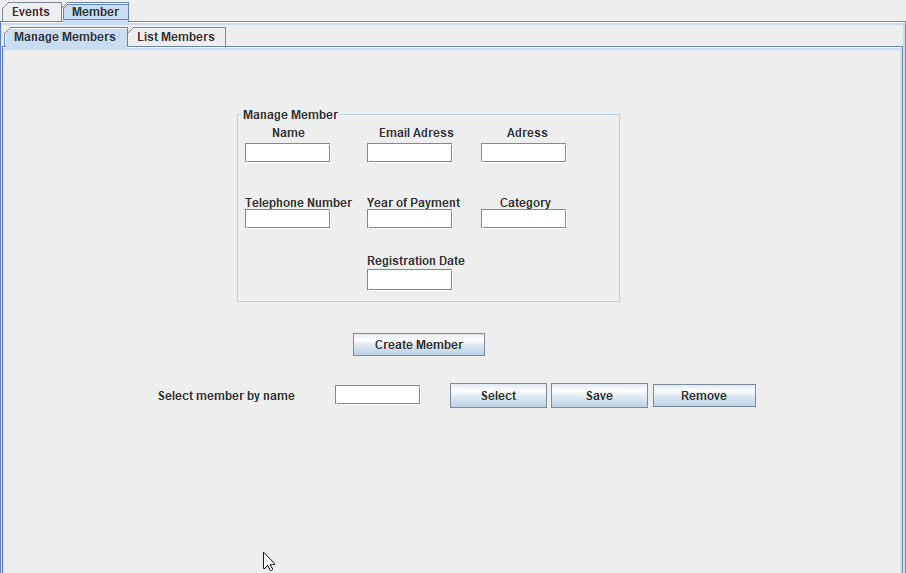
In the Manage Members tab, in order to create a member all the fields except Registration Date need to be filled. This is because most of the times a member would be created on the day the information is given.

Figure.12 GUI Manange Members tab

The List Members tab has a textArea object in which every time a button is used it shows the the result of the search. The List all Emailsl button also creates a file when pressed. It was built this way as in the interview it was mentioned the need for a list of the emails for the newsletter.

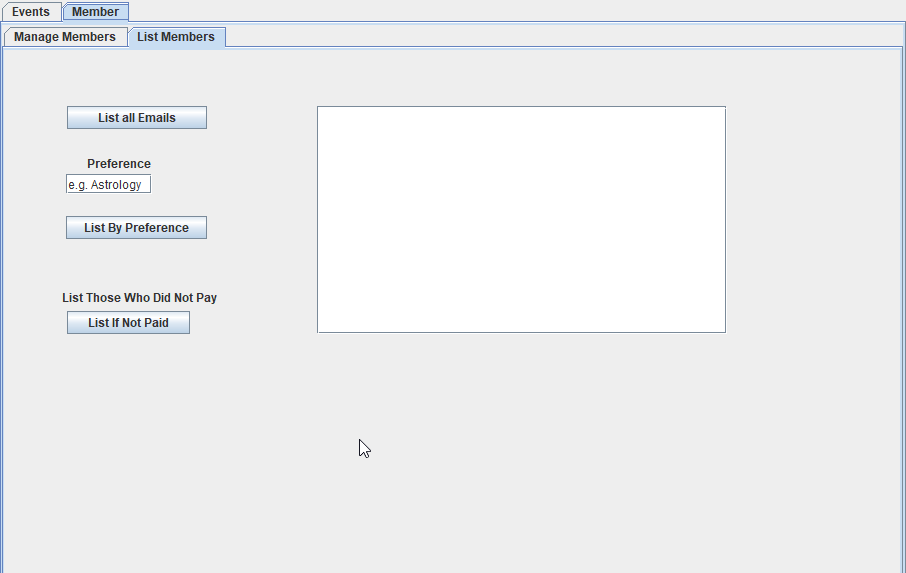


Figure.13 GUI List Members tab

**Classes for persistence**

The FileHandler(Figure.6) is the main class for persitence because it handles the reading and writing into the two main files of the system regarding the events and members. The FileHandler also handles an output file that writes the name and email of all the members. In EventList(Figure.7) two other output files are created for outputting the lecturers and sponsors by their category. dsas

# Implementation

Class MainController is responsible for all the main methods used in the GUI. It is built so it can make the changes and save everything in files that are handled in a static class. One of the most important methods is the modifyEvent as it is used for modifying the fields that describe an event. It is called through an anonymous function in the GUI at the press of the button. To modify a certain event, it searches for the index of the event in the arraylist of events and then calls the set method. In the end of the method it saves the events to a file. The important methods that modifyEvent uses: searchIndex(), saveEvents().

public void modifyEvent(Event event, String name, String nrParticipants,

String day, String month, String year, String nrMembers, String length,

String discount, boolean finalized, String type, boolean vegan,

String location, String feedback, String lecturerName,

String lecturerEmail, boolean lecturerPaidFor, String lecturerCategory,

String sponsorName, String sponsorEmail, String sponsorCategory)

{

int numberParticipants = Integer.parseInt(nrParticipants);

Date dayStart = new Date(Integer.parseInt(day), Integer.parseInt(month),

Integer.parseInt(year));

int numberMembers = Integer.parseInt(nrMembers);

int nrdiscount = Integer.parseInt(discount);

Lecturer lecturer = new Lecturer(lecturerName, lecturerEmail,

lecturerPaidFor, lecturerCategory);

Sponsor sponsor = new Sponsor(sponsorName, sponsorEmail, sponsorCategory);

try

{

eventList.getEvent().get((eventList.searchIndex(event))).setEvent(name,

numberParticipants, dayStart, numberMembers, length, nrdiscount,

finalized, type, vegan, location, feedback, lecturer, sponsor);

}

catch (IndexOutOfBoundsException e)

{

System.out.println("no event found");

}

FileHandler.saveEvents(eventList.getArrayEvent());

}

}

The searchIndex() method returns the position of a certain event that is introduced. It is used to know the position of the event we need to modify in the arrayList. It returns “-1” if no event is found.

public int searchIndex(Event event)

{

for (int i = 0; i < this.event.size(); i++)

{

if (this.event.get(i).equals(event))

return i;

}

return -1;

}

The saveEvents() method uses an array of events to save the objects in a binary file. The choice of using binary files was justified by the flexibility of writing and reading objects from them.

public static void saveEvents(Event[] events) {

ObjectOutputStream out = null;

try {

FileOutputStream fos = new FileOutputStream("events.bin");

out = new ObjectOutputStream(fos);

for (int i = 0; i < events.length; i++) {

out.writeObject(events[i]);

}

} catch (IOException e) {

e.printStackTrace();

}

finally {

try {

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

# Test

The GUI was tested based on the Use Cases assuring proper functionality to it and making sure it respects the requirements. During the implementation period the system was constantly checked for errors. The last version of the system was tested according to the use case and user guide.

1. Manage Events

|  |  |
| --- | --- |
| 1. Has all the options in the Use Case available in the GUI | YES |
| 1. The user can create an event with default fill in the fields | YES |
| 1. The user has the possibility to input data in the GUI fields | YES |
| 1. The user can input what event to remove and remove it | YES |
| 1. The system works according to design choice when the user submits the information in the GUI regarding this use case. | YES |

2. Search for Events

|  |  |
| --- | --- |
| 1. Has all the options in the Use Case available in the GUI | YES |
| 2.The result of the search by category is shown in the GUI | YES |
| 3.The result of the search of finalized events is shown in the GUI | YES |
| 4. The result of the search of nonfinalized events is shown in the GUI | YES |

3. List Lecturer/Sponsor by Category

|  |  |
| --- | --- |
| 1. Has all the options in the Use Case available in the GUI | YES |
| 1. The user can input the category in the GUI | YES |
| 1. The system creates files with the information after the user submits it in the GUI | YES |

4. List Members

|  |  |
| --- | --- |
| 1. Has all the options in the Use Case available in the GUI | YES |
| 2. The system shows all the member’s emails in the GUI and creates a file with the information | YES |
| 1. The system shows all the members in the GUI by their preference | YES |
| 1. The system shows all the members in the GUI that did not pay the annual fee | YES |

1. Manage Members

|  |  |
| --- | --- |
| 1. Has all the options in the Use Case available in the GUI | YES |
| 1. The user can input all the data and create the member | YES |
| 1. The user can input the information to modify a member | YES |
| 1. The user can remove a member by their name | YES |
| 1. The user can search for a member by their name | YES |
| 1. The system works according to design choice when the user submits the information in the GUI regarding this use case. | YES |

# Results and Discussion

The system was created as a single user system which can be used to manage events,members, lecturers and sponsors. Its target users are employees of Vipassana. The user through the GUI can access the tabs that manage events, lecturers, sponsors and members fulfilling the requirments that Vipassana presented.

The purpose of the system is to create a system which can help employees to operate the organization, by saving information of members and events in separate files. For efficiency the system was created to be user friendly.

# Conclusion

The main purpose of the project is to create a system for Vipassana which can manage events and members. Users can create an event, remove an event, add lecturers and sponsors to the event, and check all the relevant information, for example, listing all the lecturers and sponsor by their category .The system was created so it would be easily understandable, simple and fast.

During the project period, there were some issues, for example, which class in the project should be used to handle files. For all kinds of issues, our main solution was to first fix it ourselves. If they could not be worked out, we left them to the booked supervisor meetings. After presenting the issue to the supervisor, we would think of a plan for action, then continue working.

In conclusion,the project is successfully completed.The system works, and meets every requirement.

# Project future

The project works as intended and fulfills the requirments that were given. Our initial idea was more ambitious in design, but ultimately too complicated to implement in the time had at disposal. The system could have been made in such a way to have more features, for example, having the data for each member that participated at the event or having a way to search for a lecturer from external sources. The project could be improved by having a database in the future from where it should get its external information. The project could get very close to production with a better and more modern looking UI and a way of interacting with Vipassana’s website.

# Sources

Gaddis, T., 2015. *Starting Out with Java.* USA: Pearson.

SSE Classes over the period of the semester

SDJ Classes over the period of the semester

SEP Classes over the period of the semester

*Interview for Case SEP1* (2017).

1Perera, J., 2008. *Buddhism the fastest growing religion in West.* [Online]   
Available at: http://www.asiantribune.com/?q=node/10418  
[Accessed 11 2017].

2religionfacts.com, 2017. *religionfacts.com.* [Online]   
Available at: http://www.religionfacts.com/buddhism  
[Accessed 11 2017].

# List of Appendixes

10.1 Activity Diagram

10.2 Use Case Description

10.3 Class diagram

10.4 User Guide

10.5 Sequence Diagram

10.6 Analysis Diagram

10.7 Java Documentation

10.8 Project Description

[](file:///C:\Appendices.docx)