

---

# PROJECT REPORT

---

## Students

---

**Flemming Vindelev – 251398**

**Ionel-Cristinel Putinica – 266123**

**Miruna Botusan – 268013**

**Mihail Rumenov Kenchev – 266106**

## Supervisors

---

**Line Egsgaard – RWD**

**Michael Viuff – SDJ**

**Mona Andersen – SSE**

**14819 Characters**

**ICT Engineering**

**1<sup>st</sup> Semester**

**17-12-2017**

## Abstract

Vipassana – Insight Awareness is a spiritual center whose request is a single user system meant to serve the purpose of a data management system (storing members, employees and events). This project report has the aim to explain and describe the methods and stages of this system's implementation in the Java programming language. It has been designed to the customer's needs and requests, allowing a company's employee to manage its members, lecturers or sponsors, as well as create events and modify data. As preferred, the system gives the user the possibility to search for entities by entering specific criteria, and stores data efficiently through a methodical use of files and lists. The system offers the user an easy, straight-forward experience, as it automatically manages the stored batch of data. Its design has been produced such that it makes allowances for company changes and expansion.

## Table of Content

Abstract.....	2
Table of Figures.....	5
Introduction .....	6
Analysis.....	7
Requirements .....	7
Use Case Diagram.....	9
Use Cases.....	10
Activity Diagram .....	11
Analysis Class Diagram .....	13
Design.....	14
GUI – Design .....	14
Design Class Diagram .....	16
Sequence Diagram.....	18
Implementation .....	19
Test.....	21
Search .....	21
Procedure .....	21
Result .....	21
Create .....	21
Procedure .....	21
Results.....	21
Modify .....	22
Procedure .....	22
Results.....	22
Result.....	23
Discussion.....	23
Conclusion.....	24
Appendices.....	24
Appendix 1:    Use Case Diagram .....	24
Appendix 2:    Use Case Descriptions.....	24
Appendix 3:    Activity Diagrams .....	24

Appendix 4: Class Diagram.....24

Appendix 5: Sequence Diagram.....24

Appendix 6: User Guide .....24

Appendix 7: Project Description .....24

## Table of Figures

Figure 1: Use Case Diagram .....	9
Figure 2: Activity Diagram.....	12
Figure 4: Front Menu of the System .....	14
Figure 5: Create Lecturer Menu of the System.....	15
Figure 6: Search Lecturer Menu of the System .....	16
Figure 8: Sequence Diagram .....	18
Figure 9: Lecturer & Sponsor Classes.....	19

## Introduction

Buddhism is a religion which promotes a set of values and beliefs which have been formerly introduced by Buddha sometime between 6<sup>th</sup> and 4<sup>th</sup> centuries BC. It represents the fourth most followed religion in the world, as of 2010, with 7.2% of the global population. One of the associated practices is meditation, amongst many other spiritual activities including dream interpretation, healing, reincarnation etc. Today there are organizations which put forth the opportunity of trying some of these activities in a professional environment, such as Vipassana – Insight Awareness.

Currently, there are around 350 million Buddhists and a growing number of them are Westerners, and the numbers only seem to grow. Ultimately, this can logically only mean an increase in the popularity of spiritual centers and spirituality itself. Having a community that promotes self-growth can be life-changing, and individuals have been globally discussing the matter of spirituality amidst a community for a few years now ('There is nothing challenging about having deep thoughts all by oneself. What is interesting is doing this work in community [...]').

For the mentioned reasons, Vipassana – Insight Awareness is looking to grow and expand. It is a non-profit organization that offers lectures, workshops and journeys with the aim of helping individuals find inner peace. It offers clients the possibility to get educated in the art of meditation and karma.

A faster, better organized structure would allow Vipassana to manage their business in real time, which could offer the company a great advantage amongst the other popular spiritual centers around the world, but most importantly, boosting their relevance locally, becoming an attraction even for the less spiritual groups. To see a more in-depth background description, look at Appendix 7.

## Analysis

### Requirements

A list of requirements for VIA system has been set up, which allows the opportunity to analyze the interaction between user and system. The list has a section for functional requirements, what the system should be able *to do*. This section has been split in to two sub-sections, *top-priority* and *non-priority*, which defines what *has* to be in the system and what *may* be in the system. The second section is non/functional requirements, which tells some qualities about the system.

FUNCTIONAL REQUIREMENTS	
Top Priority Requirements	<ol style="list-style-type: none"><li>1. The user should be able to create events (Workshops, seminars, lectures, journeys)</li><li>2. The user should be able to register member my name, email address, phone number and payment status.</li><li>3. The user should be able to register employees by name, field of work, email address and phone number.</li><li>4. The user should be able to search for events, members and contributors.</li><li>5. The user should be able to modify the data stored for events, members and contributors.</li></ol>
	<ol style="list-style-type: none"><li>1. The system should facilitate searches with a filter.</li></ol>

Non-priority Requirements	
<b>Non-functional Requirements</b>	
<ol style="list-style-type: none"><li>1. The system must be programmed in Java.</li><li>2. The system must be destined for a single user.</li></ol>	



## Use Case Diagram

The requirements of the VIA System has been generalized and made into a use case diagram, which explains the interaction between user and system in a simple way. The diagram shows the functions of the system, of which the user can access.

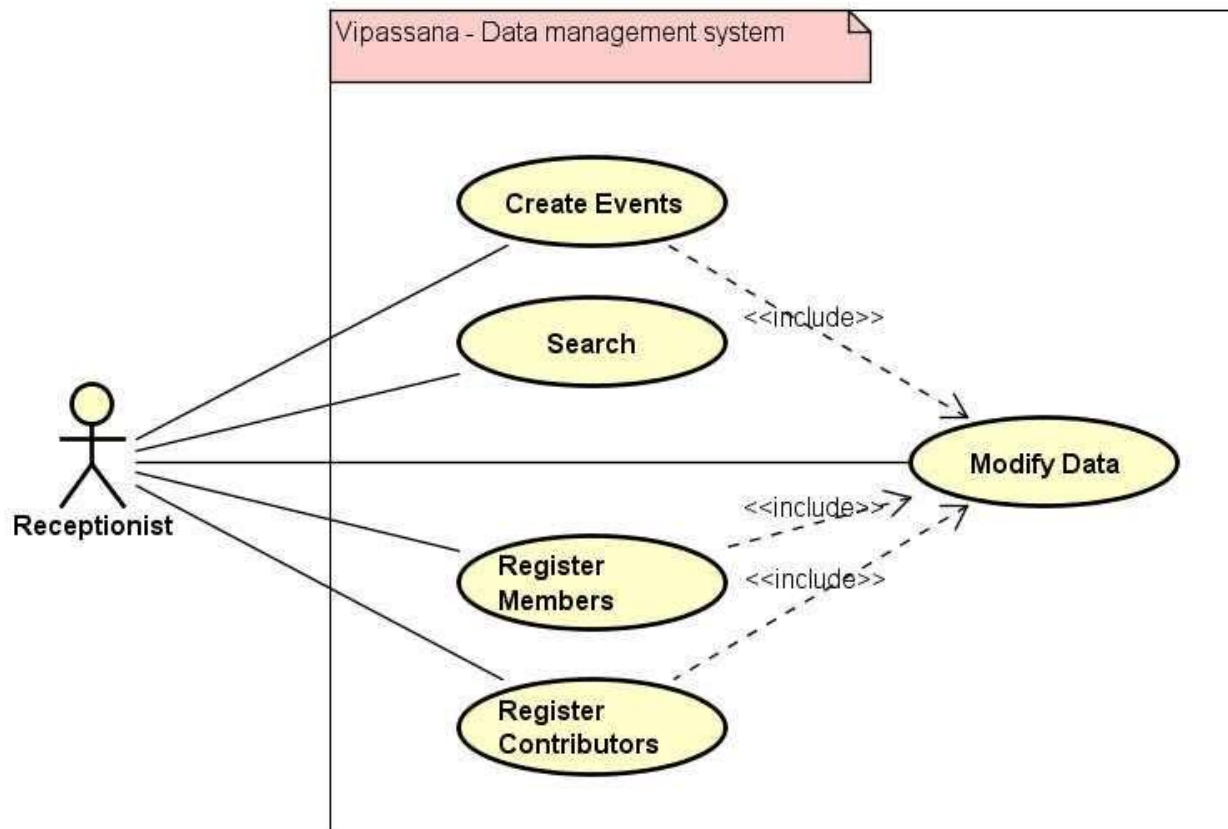


Figure 1: Use Case Diagram

## Use Cases

Below is one of the use case descriptions, which in depth explains how the Create Event function from the use case diagram works. There is a description for every use case found in the diagram. This can help the user to use the system, much like guidelines. For more Use Cases, look in Appendix 2.

Use Case	Create event
Summary	Create a new event defined by the user.
Actor	The receptionist/the employee
Pre-condition	There must be available contributors to lead the event.
Post-condition	The data for the event is stored.
Base sequence	<ol style="list-style-type: none"> <li>1. User chooses the type of event to create (lecture, journey, workshop and seminar).</li> <li>2. User titles the new event.</li> <li>3. User enters event subject/theme.</li> <li>4. User chooses date and time for the event.</li> <li>5. User enters event duration.</li> <li>6. User chooses a contributor, from a list, to be assigned for this event.</li> <li>7. User sets a member limit for attendance.</li> <li>8. User enters the price of the event.</li> <li>9. User enters location for event.</li> <li>10. User clicks “save” button.</li> <li>11. System validates data.</li> <li>12. System creates new event in memory.</li> </ol>
Exception sequence	<p>If the user chooses Journey event type:</p> <ol style="list-style-type: none"> <li>1-3</li> <li>4. User enters date, time and ending date for event.</li> <li>5. No contributor will be assigned.</li> </ol>

	6-12 If the user chooses Workshop event type: 1-3 4. User enters date, time and ending date for event. 5-12 If data is invalid: 1-12
--	--

## Activity Diagram

**Create Event** activity diagram shows the process handled when the user creates a new event. The purpose of this diagram is to guide the user through the steps behind actually creating a new event.

The first step is for the user to choose which type of event (workshop, seminar, etc.) they wish to create. If the user chooses Journey, the user also has to choose a journey destination, and duration. Afterwards, the user can input title, date, member limit, duration and price, which subsequently will be referred to as practical information. After inputting the practical information, the user gets to either cancel, save or finalize the event. Cancel will delete the event, which is currently being worked on. Save will save the event in the system, but not have it as a set event which is ready to be used as a reference. Finalize will make the event finished, which means this is now an event ready to be held.

If user chooses lecture the user will have to choose a subject, and assign a contributor to the event. Afterwards the user has to input the practical information, and either cancel, save or finalize the event.

If the user chooses workshop/seminar, the user has to input subject and/or speeches. Furthermore, the user assigns a contributor, inputs the practical information and either cancels, saves or finalizes the event. For more Activity Diagrams look, in Appendix 3.

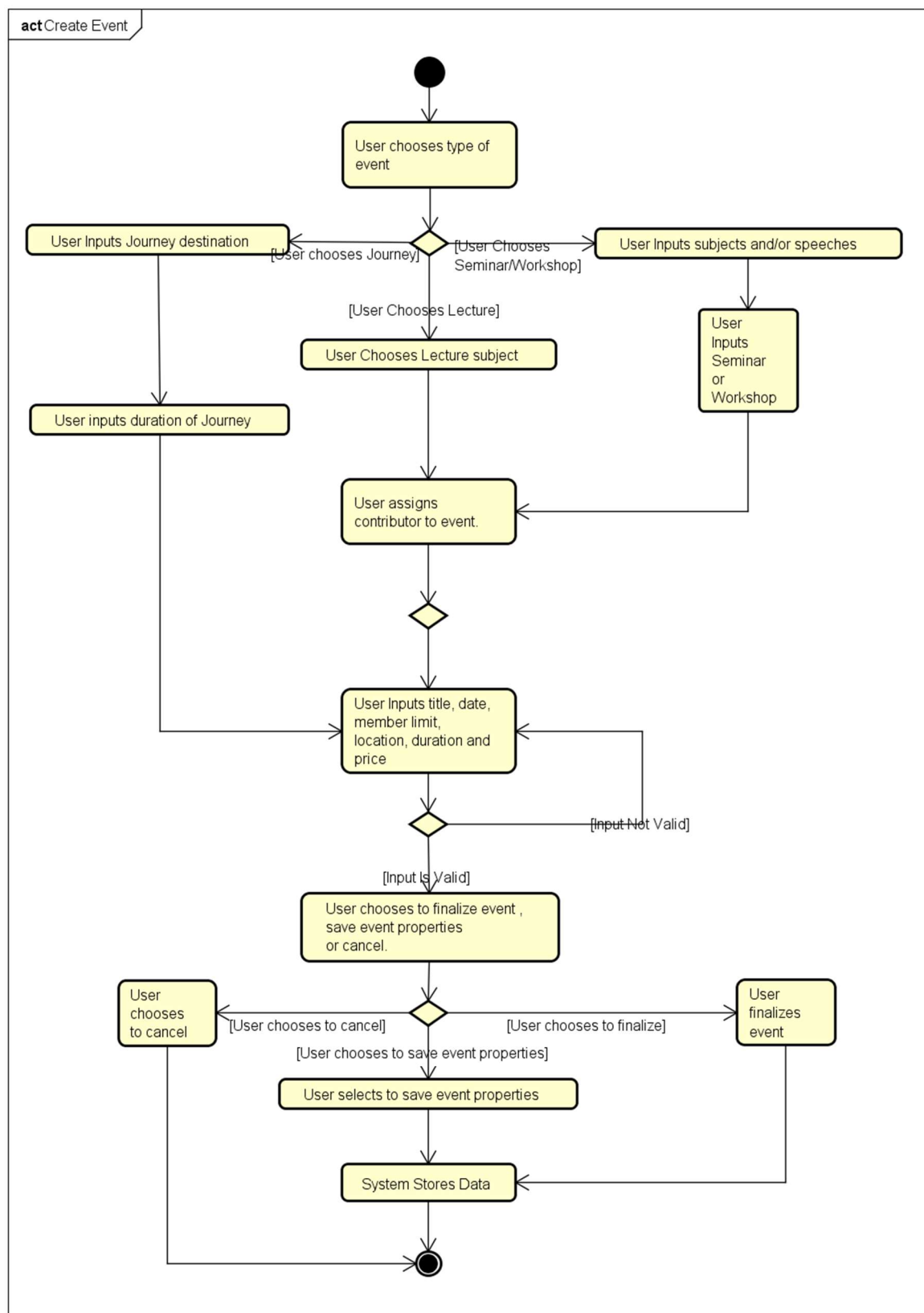
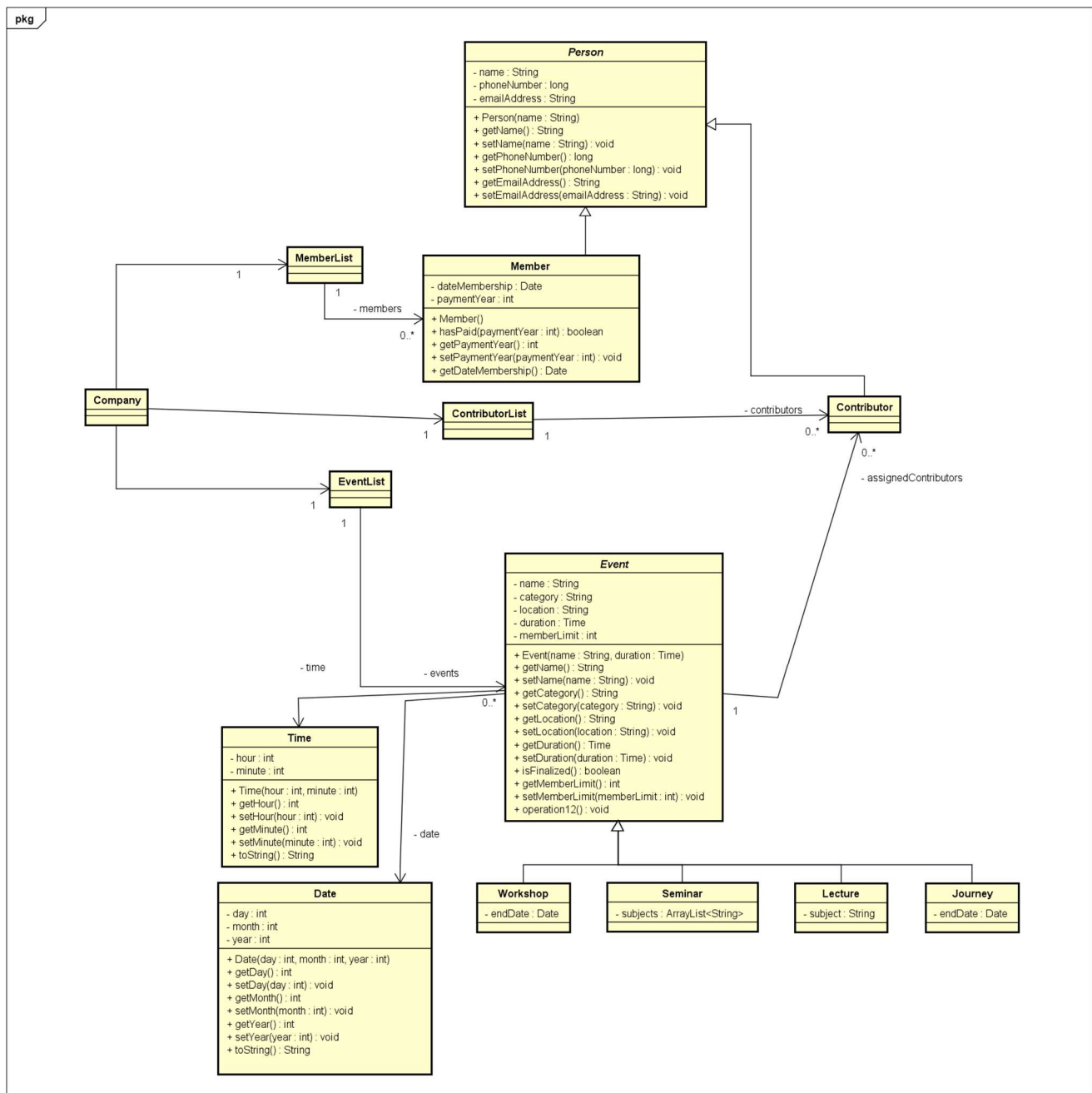


Figure 2: Activity Diagram

## Analysis Class Diagram

ANALYSIS: The model class diagram displayed here shows the relationships between the classes chosen, based on the requirements proposed by the company. It represents a first draft of the 'mapping' of the system, to which no changes have been made, moving into the Design stage.



## Design

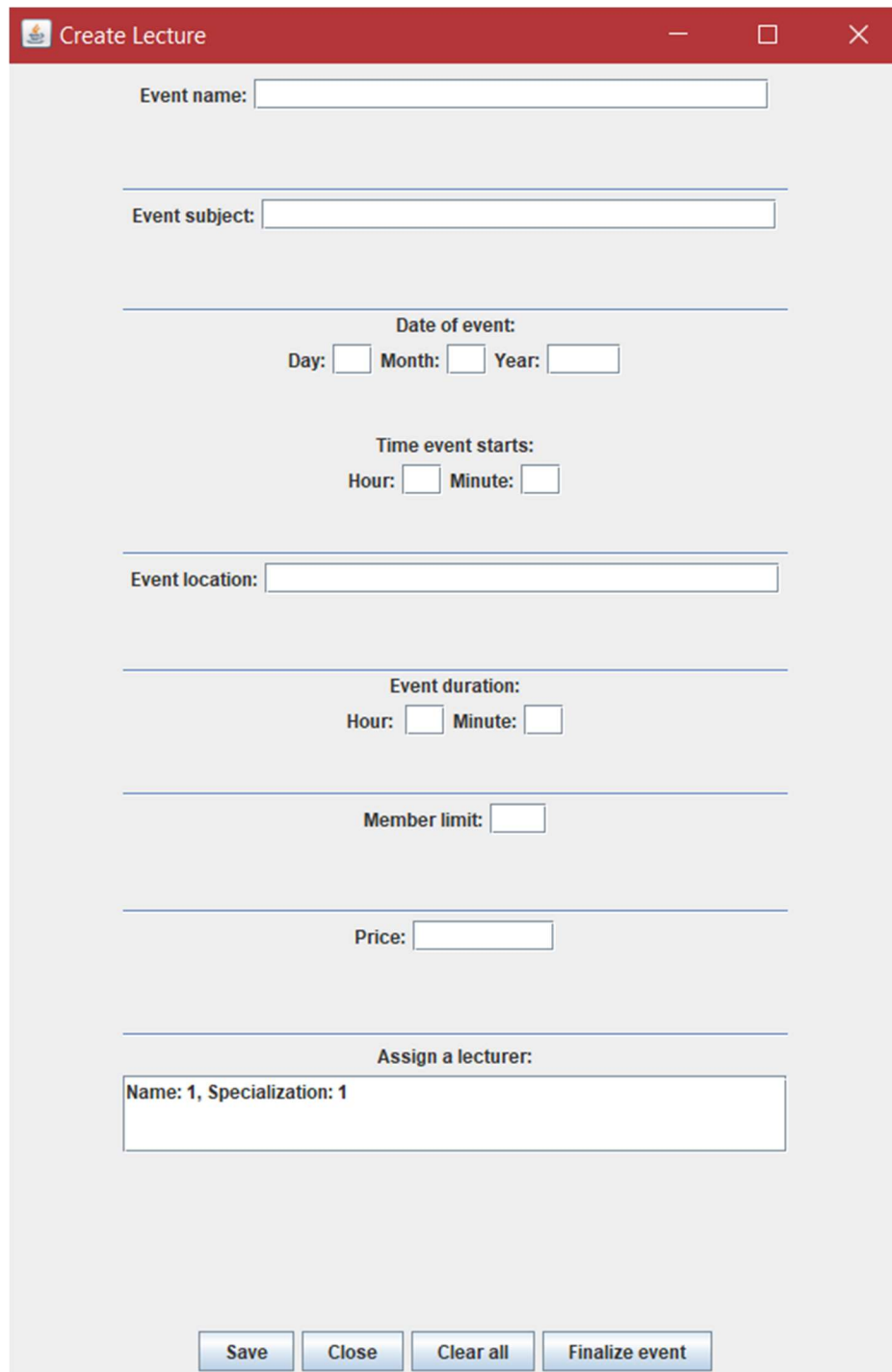
### GUI – Design

The first aspect about the GUI design is a front menu which presents 8 choices to the user, out of which 4 are for creating new instances, and the other 4 are designed for searching and updating or modifying existing instances: 'Create event/member/lecturer/sponsor', 'Search event/member/lecturer/sponsor'. To see a full user guide of the system, look at Appendix 6.



Figure 3: Front Menu of the System

The create buttons open a menu with a few fields to be filled out, depending on the user's Create choice. For instance, the 'Create Event' button opens a menu with 4 other buttons, 'Create seminar/workshop/lecture/journey', which in turn open a menu with fields which are initially inputted as Strings, being parsed into their specific data type. Such fields are 'Event name', 'Date of event', 'Assign a lecturer' etc. The 'Assign a lecturer' is the interesting part in the menu, having been implemented through a JList. At the bottom of this window, there are a few buttons to either save, clear or finalize the specific instance. The Finalize button allows the user to lock the created instance from having its fields modified at a later time, which in turn, allows the user to create some instances which do not have all fields filled in. The Clear function resets the values of the fields.



The image shows a software window titled "Create Lecture" with a red header bar. The window contains several input fields for creating a new lecture event. The fields are organized into sections separated by horizontal lines. The sections are: "Event name:" with a single-line text box; "Event subject:" with a single-line text box; "Date of event:" with three separate boxes for "Day:", "Month:", and "Year:"; "Time event starts:" with two separate boxes for "Hour:" and "Minute:"; "Event location:" with a single-line text box; "Event duration:" with two separate boxes for "Hour:" and "Minute:"; "Member limit:" with a single-line text box; "Price:" with a single-line text box; and "Assign a lecturer:" with a single-line text box containing the text "Name: 1, Specialization: 1". At the bottom of the window, there are four buttons: "Save", "Close", "Clear all", and "Finalize event".

Event name:

Event subject:

Date of event:  
Day:  Month:  Year:

Time event starts:  
Hour:  Minute:

Event location:

Event duration:  
Hour:  Minute:

Member limit:

Price:

Assign a lecturer:  
Name: 1, Specialization: 1

Figure 4: Create Lecturer Menu of the System

The search buttons allow the user to list instances of a specific data type, as well as filter the results based on one criteria. In this menu, the user has the possibility to modify one of the displayed items – this procedure opens the same initial 'Create' window with the specific data type fields, which are editable. When editing an instance, the 'Close' button is hidden because at

this point, the instance is deleted from both the file and the list of its specific data type. Once the modified version is saved, a new instance is stored in the file and the mentioned list.

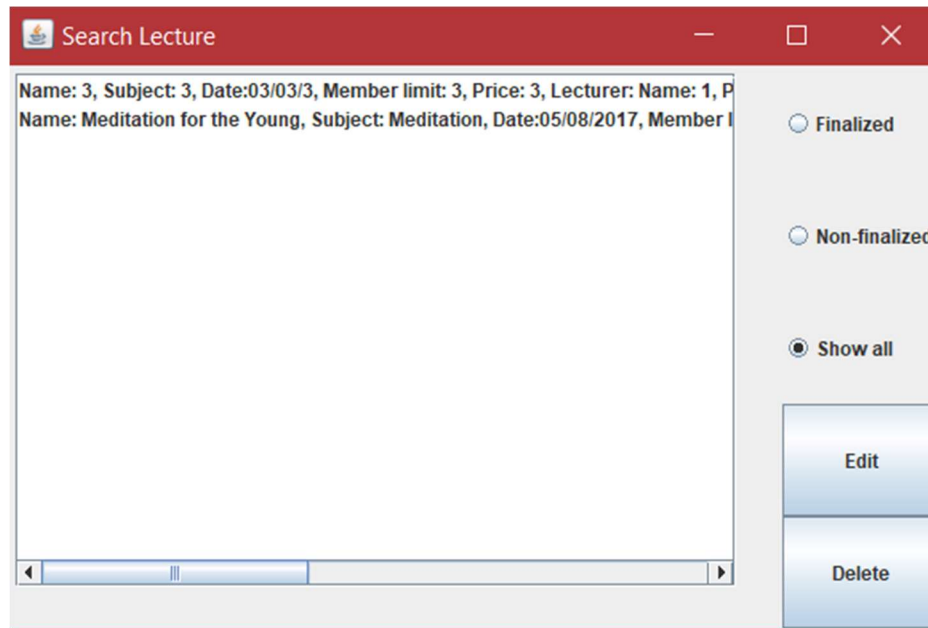


Figure 5: Search Lecturer Menu of the System

## Design Class Diagram

DESIGN: The design class diagram is a final version of the one created in the previous stage. Its purpose is to describe and illustrate the relations and functions of the system. The attributes and methods have been created in such a way to satisfy the requirements of the customer. It serves as a blueprint of the system's functionalities, ergo the code is generated accordingly. This diagram has been updated throughout both the Design and Implementation stage, in order to keep the mapping relevant. To get a better look at the Class Diagram look in Appendix 4.





## Sequence Diagram

**Create Lecture GUI “save”** sequence diagram is a diagram that shows the flow of operations in a method, and in which order it happens.

Firstly, the CompanyFile reads EventList, then the actionPerformed calls setName to the event. Next it creates a new Date, which is used to setStartingDate. After that, it creates a second new Date, which is used for the setEndingDate. Next, it calls the Time class to create a new Time, which is used to setScheduledTime. Then another new Time to setDuration. Then actionPerformed calls the setSubject to Event, setMemberLimit to Event, setPrice to Event and setLecturer to event. Lastly the Event calls the addEvent method to the EventList, which then calls writeEvents. This puts the event into a file.

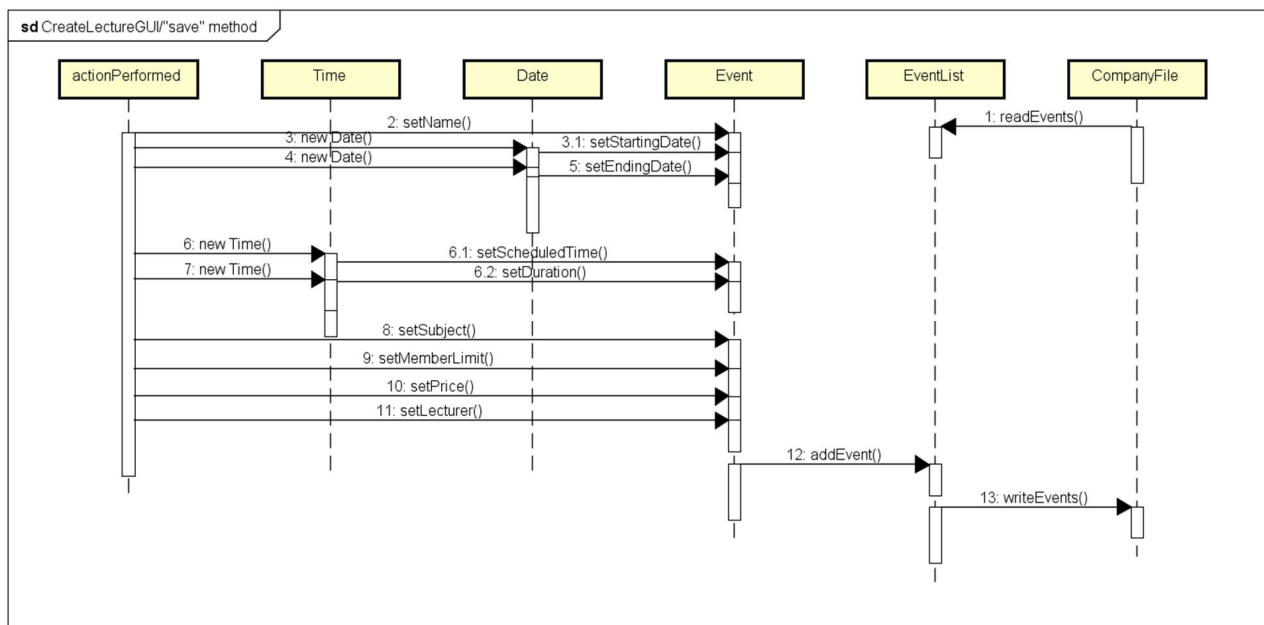


Figure 6: Sequence Diagram

## Implementation

The implementation phase of VIA System required the group to split some of the classes, add some methods and fix the association between the classes.

One of the biggest problems during the implementation phase was having two of the most important classes “Sponsor” and “Lecturer” combined into one class “Contributor”. Due to the different attributes each had, “Contributor” class was split into two child classes, with their own separate lists and files, having “Person” as their parent class.

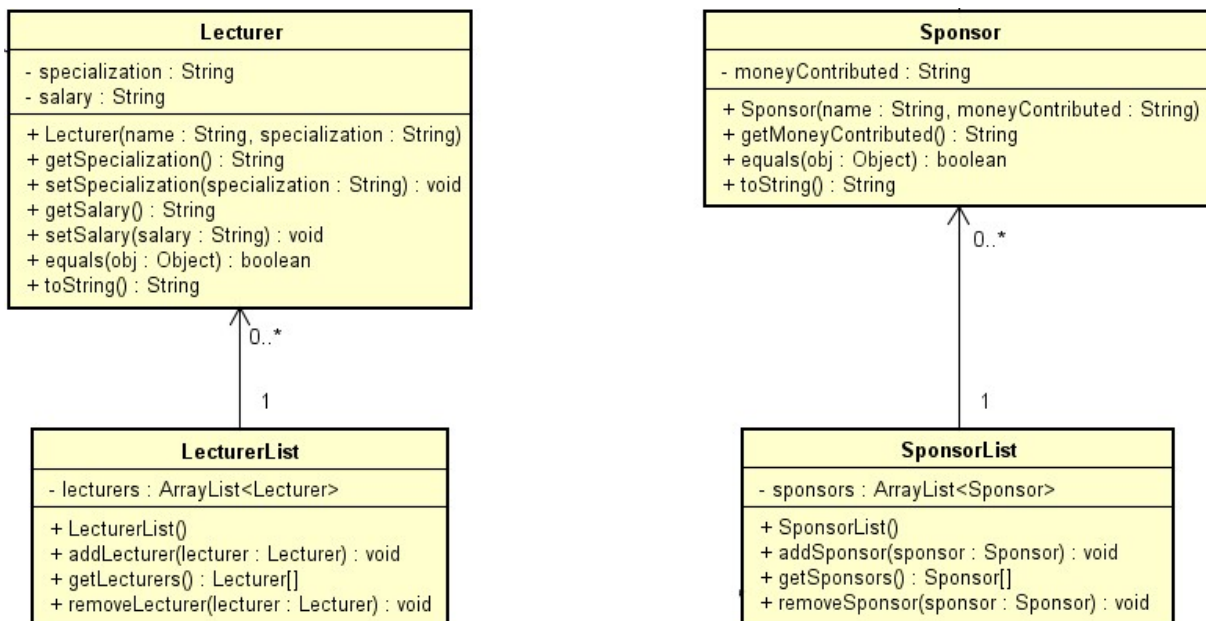
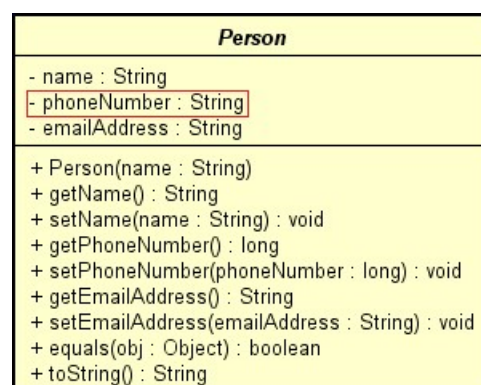
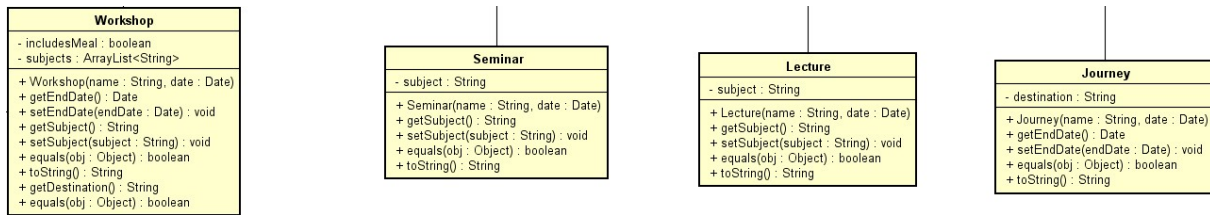


Figure 7: Lecturer & Sponsor Classes

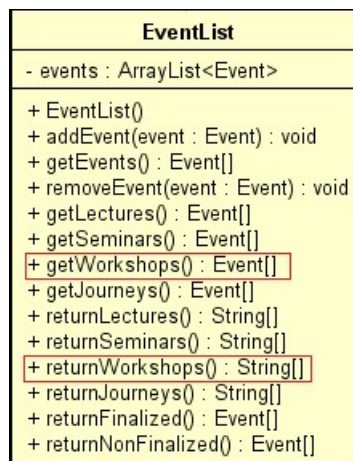
Some fields holding information for specific objects were also changed to “String”, to avoid as many exceptions as possible when invalid type of data is entered.



Additional class-specific methods, like “getSubject” and “getEndDate” were added to “Event” child classes.



List classes were made to include two return types. The first return is of type “String[]” where all the information of the objects stored inside is returned as a String array, conveniently making them easy to be displayed in “JList” format. The second return type is of type “Object[]”, which we included in the advancement of our implementation phase.



Overall the implementation phase was simple, due to the inclusion of all the core elements of the system during the design period. Because of the nicely structured code, all the system bugs were swiftly located and fixed.

## Test

### Search

The main function of the Search feature is to find a specific piece of data, which the user then can use for any purpose.

### Procedure

The test is performed by clicking one of the four “search” buttons in the front menu of the program (Search Event, Search Member, Search Lecturer and Search Contributor). After clicking one of these buttons, a list with all the stored data in the category is shown.

If “Search Event” button is clicked, a menu will pop up where the user has to choose one of four buttons. Each button represents a type of event (workshop, seminar, etc.) after that a list of all the stored data in this category will be shown.

When the list is shown, there are three buttons in the side which either show “finalized”, “non-finalized” or “all” the data which is stored of this type.

### Result

This feature works as intended, no errors occurred under testing phase.

## Create

The intent of this function is to allow the user to create either a new event, lecturer, sponsor or member. Depending on which of the above categories are chosen, the user has to input a different set of information.

### Procedure

The test is performed by choosing either “Create Event”, “Create Member”, “Create Lecturer” or “Create Sponsor”. At this point, all the practical information has to be filled in, which changes respective to what type of event is chosen. And lastly, either Cancel, Save or Finalize.

### Results

This function works correctly.

## Modify

This function is made for the user to make any adjustments to the data stored in the system.

## Procedure

After having searched for any sort of data in the system, there is a button named “Edit”. If a string of data is selected by a click of the mouse, the Edit button becomes available. When clicked any practical information that has not been filed in, can now be done. And any data already input can be edited. After information has been edited the data can again be saved.

## Results

This feature also works the way it is intended, following the above procedure.

## Result

VIA System was created with the purpose of meeting the requirements of a data management system: storing, searching, editing and deleting information. The system allows the user to register different types of events, members, lecturers and sponsors and also offers the possibility of searching for all the data in the system, modifying it and deleting it.

All the features of the system were implemented into a user-friendly graphical user interface, using buttons, radio buttons and lists that have different functionalities that allow the user to control the system.

- The user should be able to create events (Workshops, seminars, lectures, journeys)
- The user should be able to register member my name, email address, phone number and payment status.
- The user should be able to register employees by name, field of work, email address and phone number.
- The user should be able to search for events, members and contributors.
- The user should be able to modify the data stored for events, members and contributors.
- The system should facilitate searches with a filter.
- The system must be programmed in Java.
- The system must be destined for a single user.

## Discussion

When starting creating the system, the main focus was to follow the requirements, with the idea that the system has to fulfill all of them, integrated into a simple and easy-to-use GUI. Along the way, problems arose when implementing the system, but they have were solved, and the result is a fully working and bug free system.

One thing that has to be mentioned is that, along the way, some of the requirements have been changed or completely removed, and some new ones have been added, but all those changes were done for the efficiency of the system and a better end-user experience.

## Conclusion

The purpose of this project was to create a system that keeps track of Vipassana's lecturers, events, members and sponsors, and the result of the project is a system that perfectly reflects all those requirements, and fulfills all the needs of Vipassana.

When implementing the code, a lot of problems occurred, but, all of them were successfully managed, and the purpose of the project was fully achieved: a system that meets all the needs that Vipassana has, is easy to modify and maintain.

## Appendices

Appendix 1:	Use Case Diagram
Appendix 2:	Use Case Descriptions
Appendix 3:	Activity Diagrams
Appendix 4:	Class Diagram
Appendix 5:	Sequence Diagram
Appendix 6:	User Guide
Appendix 7:	Project Description