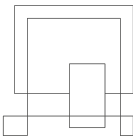


Vipasanna Management System

Group: 6

Student Name	Student No
Daniela Koch	266502
Matej Michalek	266827
Michaela Golhova	266099
Michal Karol Pompa	266494
Remedios Pastor Molines	266100

Contents

I hereby declare that 6 project group and I prepared this project report and that all sources of information have been duly acknowledged.

Abstract

The main aim of the project was to make a maintainable system that provides Vipassana (a center for spiritual events) easy access to information, as well as provides solution for the organisation's struggle concerning finding proper lecturers. The system had to grant the user a list of actions, such as signing up new members, creating events, searching for lecturers or generating a newsletter. In this following document all of those actions, that have been derived from an interview conducted with the organisation, have been stated in a list of requirements. The list is followed by analysis of the requirements, whichs examples are presented in a visual form- as diagrams. The next step is a description of the design of the system. It involves among other an explanation of the choice of the Model - View - Controller architecture pattern, as well as an explanation of the choice of using a SWING library for GUI and not using the Window builder. It is followed by a brief introduction to the code, which has been written in Java. At this point, attention should be paid to one of the most interesting elements of the implementation- the search engine. Tests of the system have been made and it has been proven to had been completed as requested. Finally a positive outcome is stated and the results are presented. Last but no least, the future of the project is being discussed. It mentions developing the system, to give an example by adding a database in due to store data.

1 Introduction

Nowadays due to the fact that people are living in a rush and that the professional life has taken advantage on the other aspects of life they are often exposed on a huge amount of stress (WHO, 2017). Together with the popular unhealthy lifestyle- both physically (the sedentary lifestyle) and mentally (continuous stress)- it often causes depression. Globally, more than 300 million people suffer from this disease. That is one of the biggest issues of the 21st century (WHO, 2017).

One way of solving the issue of spreading of depression is provided by Vipassanā - Insight Awareness (VIA). VIA is a center for spiritual events originally with a base in the Buddhist principles of meditation as an insight with awareness of what is happening as it happens. Today events at VIA also include spiritual practices not directly linked to any religion. Such practices, to take an example meditating, can reduce stress (Psychiatry online, 2006). Furthermore taking part in lectures, seminars or workshops which are provided by VIA keeps one's mind away from work and lets it rest and relax. What helps people even more are the trips promoted by the organization. Not only do they allow one to forget about the everyday routine, but also force one to move, breathe fresh air and spend time close to the nature.

However Vipassanā - Insight Awareness is not adapted to the today's world based on new technology. As the world depends more and more on the work of computers, keeping a paper-based system is hardly possible. Due to the fact that today the internet is one of the most popular sources of information (Taylor & Francis, 2017), a computerized system would increase the range of attendance on the events. The organization is in a need of keeping track of events, members, lecturers in an easy way. It needs a system that would store data and provide an easy access to them (to take an example to simplify sending emails to the members). Moreover, a computer-based system would solve the complication of searching for proper lecturers. What is more, the system should be maintainable, what would enable Vipassana to easily add new functionalities. Even though similar systems already exist they do not meet the needs of our client. That is due to the fact that VIA expects a simple server and the existing ones are complicated.

2 Analysis

Successful projects are based on analysis. It has a significant impact not only on the amount of spent time, but also on the general project outcome. Therefore, next stages as Design, Implementation and Test were derived from deep analysis of the stated requirements. Main questions concluded from the interview are listed below:

- How to store data about events, members and lecturers?
- How to find proper lecturers for events?
- How to make the system easy to extend and provide new functionalities?

Extended analysis takes into consideration every issue, but sometimes there are some difficulties that are not possible to overcome and that is why creating a list of delimitations is needed. Due to restriction on storing data and lack of experience and time, the delimitations are summarized as follows:

- The system will not use database to store data.
- Data will be provided by files, not from real users.
- The system will not be storing feedback.
- The system will not be sending emails automatically to users.
- The system will not look for events between a specific time period.

2.1 Requirements

Bearing in mind the needs of VIA presented in the introduction, as well as stated problems and delimitations, the requirements concerning the needed system are divided into functional and non-functional requirements.

2.1.1 Functional Requirements

All functional requirements are stated in the list below:

1. The administrator must be able to sign up a new member storing information about him (name, address, phone, email, date of membership and payment year).
2. The administrator must be able to sign up a new lecturer.
3. The system has to search for lecturers of a given category.
4. The administrator must be able to plan events on specific dates, about specific topics, and with specific lecturers.
5. The administrator must be able to modify event data.
6. The system must allow the administrator to search for not finalized events.
7. The administrator must be able to sign up participants and members to specific events.
8. The system must count the price for members, having considered the discount for events depending on the event type.
9. The system must be able to return the amount of available places for every event.
10. The system must collect a list of lecturers who want a fee or advertisement for newsletter.
11. The system must collect finalized events for newsletter.
12. The user must be able to generate newsletter.
13. The system has to generate a list of emails of members who have not paid

Use Case Description and Activity diagrams for all functional requirements are attached in Appendix E

Some of the most important Use Cases are listed below.

Use Case #3: Search for a lecturer of given category

Search for a lecturer of given category / UseCase Description	
ITEM	VALUE
UseCase	Search for a lecturer of given category
Summary	Searching for lecturers
Actor	VIA Administrator
Precondition	None
Postcondition	Lecturers have been found
Base Sequence	1. Initialize new empty list of suiting lecturers 2. Get list of all lecturers 3. Check if list of all lecturers has next element 3.1. If yes: 3.1.1. Take next lecturer from the list of all lecturers 3.1.2. Get list of all his categories 3.1.3. Check if list of categories has a next element 3.1.3.1. If yes: 3.1.3.1.1. Take next category from the list 3.1.3.1.2. Check if category equals lecturers category 3.1.3.1.2.1. If yes: 3.1.3.1.2.1.1. Add lecturer to the list of suiting lecturers 3.1.3.1.2.1.2. Go to step 3 3.1.3.1.2.2. Otherwise: 3.1.3.1.2.2.1. Go to step 3.1.3 3.1.3.2. Otherwise: 3.1.3.2.1. Go to step 3 3.2. Otherwise: 3.2.1. Return list of suiting lecturers

Use Case #4: Planning an event

Plan an event / UseCase Description	
ITEM	VALUE
UseCase	Plan an event
Summary	An administrator plans an event in event calendar on specific date
Actor	VIA Administrator
Precondition	None
Postcondition	An event is planed
Base Sequence	1.Choose type 2. User decides if he/she wants to add topic 2.1. If yes, user adds a topic 3. System checks if it is a trip 3.1. If yes, user decides if he/she wants to add location 3.1.1. If yes, user adds location 3.2. If no, user decides if he/she wants to add lecturer 3.2.1. If yes, user chooses a lecturer (USE CASE: Search for lecturer...) 4.User decides if he/she wants to add date 4.1.If yes, user adds a date 5.User decides if he/she wants to add price 5.1. If yes, user adds a price 6. System checks if event can be finalized 6.1.If yes, sysem enables "Finalize" checkbox 6.2. User desides if he/she wants to finalize the event 6.2.1. If yes, user selects "Finalize event checkbox" 7. System saves the event

Use Case #7: Sign up participant to an event

Sign up participant to an event / UseCase Description	
ITEM	VALUE
UseCase	Sign up participant to an event
Summary	Administrator signs up new particioant to an event
Actor	VIA Administrator
Precondition	There is an event and participant to sign up
Postcondition	Participant is signed up
Base Sequence	1. IF user wants to add member to an event 2. Search member in MemberList by ID 3. Add member to Event
Branch Sequence	IF user want to add non member participant 1. Create new participant 2. Add participant to Event

Use Case #12: Generate newsletter

Generate newsletter / UseCase Description	
ITEM	VALUE
UseCase	Generate newsletter
Summary	An administrator ganerates newsletter, which will be sent to members
Actor	VIA Administrator
Precondition	None
Postcondition	Newsletter has been genereted
Base Sequence	1. The system generates a list of finalized not finished events 2. The system generates a list of lecturers who want advertise 3. The system generates todays date 4. User decides if he wants to add additinoal information 4.1.If yes, user adds additional information to newsletter 5. System saves newsletter
Branch Sequence	
Exception Sequence	1. There are no lecturers to advertise THEN generate without advertisments 2. There are no finilized events THEN generate without eventsOR search for events to finalize (Use Case: Sea rch for not finalized events)
Sub UseCase	Search for finalized events Collect the list of lecturers who wants an advertisement

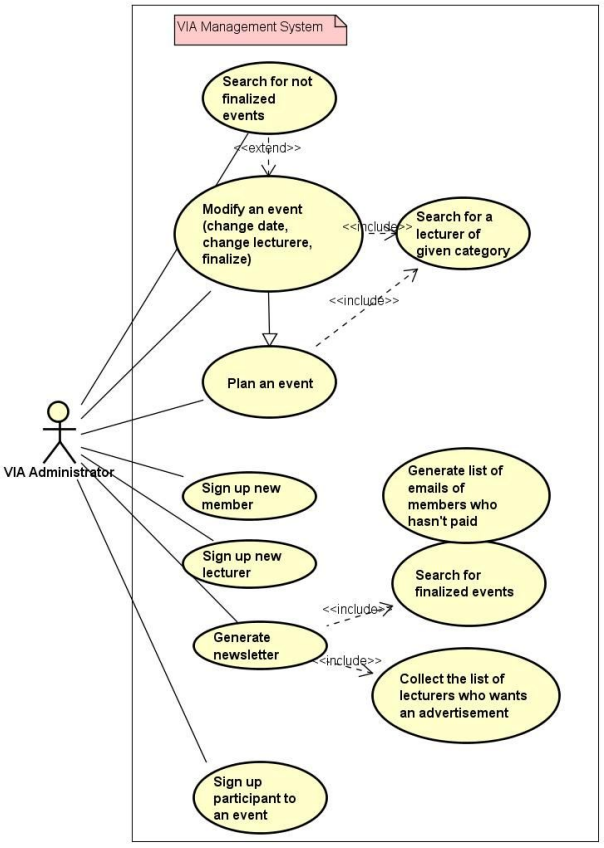
2.1.2 Non-Functional Requirements

Below there is a list of all non-functional requirements for this project.

1. The system must be implemented in Java
2. Secondary storage needs to be done with files
3. The parts of the system must be accessible among them (feedback with events/sponsors/event calendar).

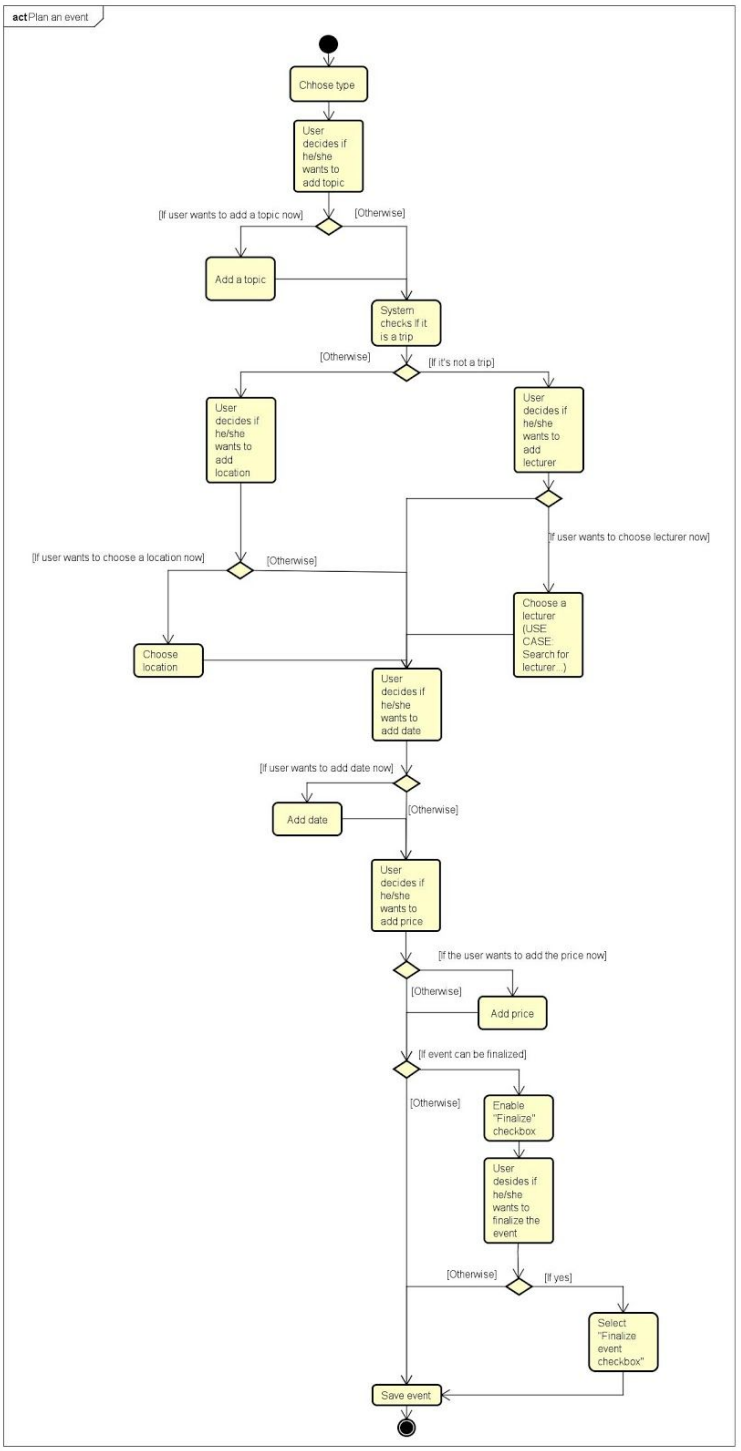
2.2. Analysis of needed classes and relations between them

Basing on the requirements, which are based on the interview, needed classes can be distinguished. The system contains several main objects such as event, member, lecturer and newsletter. These objects are controlled by VIA administrator in the way described in the figure below:



EventList, MemberList, LecturerList and Newsletter objects are needed for primary data storage, which will also handle basic operations like signup a member/lecturer/participant, generate newsletter or add new events.

Events type must be distinguished within lecture, seminar, workshop and trip. The difference between these types is handled in each class separately, meanwhile general event fields and functions are stated in parent abstract class - Event. For instance, Lecturer class is not related to Trip as VIA does not provide lecturers for this type of events. On the other hand, just one lecturer can be assigned to lecture and many lecturers to seminar and workshop. The activity diagram of creating an event is shown below.



Furthermore, class Event contains the list of participants which consists of both participants (non-members) and members who were signed up by administrator. The idea behind this is that whereas every member can be a Participant of some event, not every participant is necessary a member.

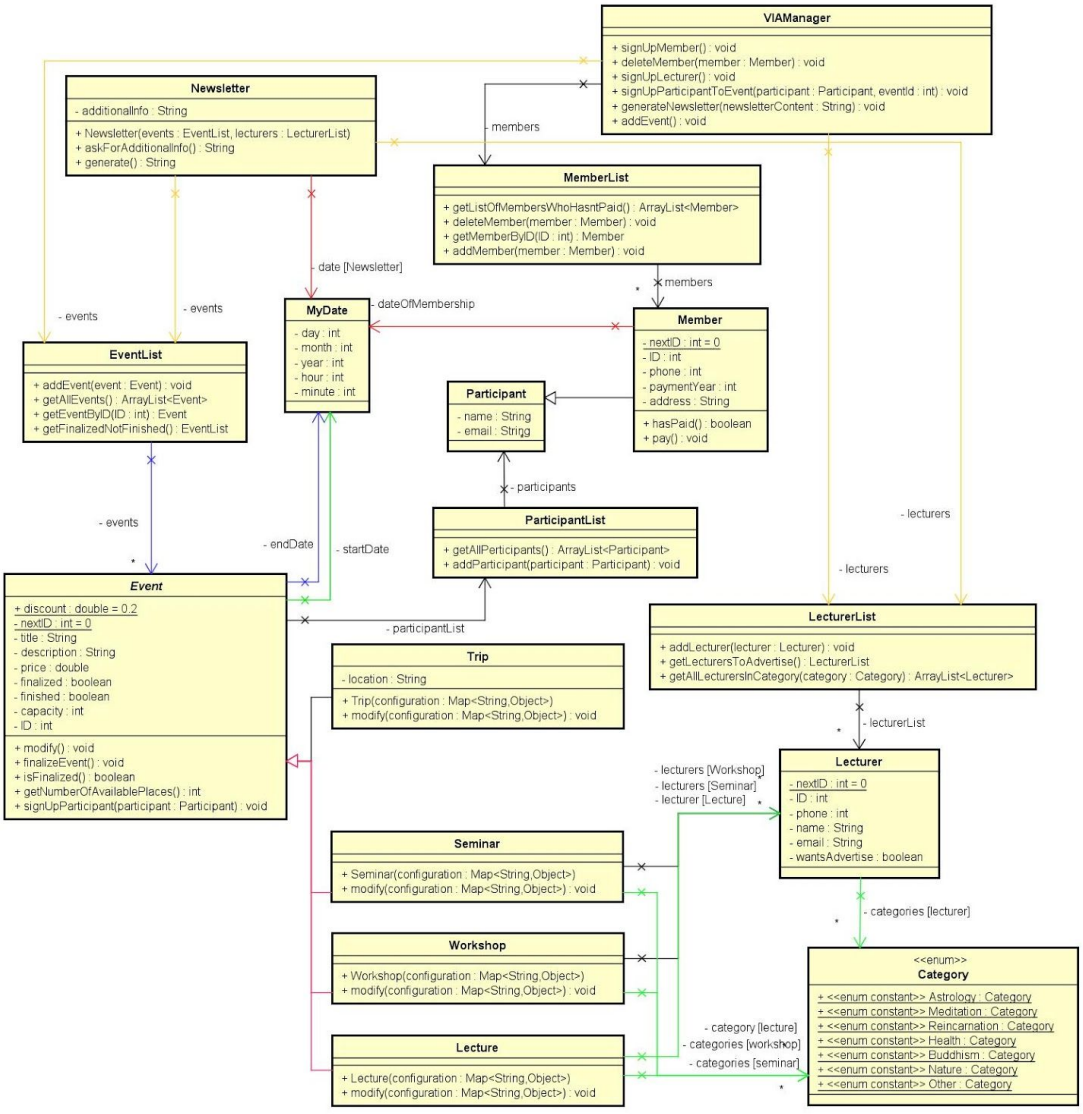
3 Design

Based on analysis, Model - View - Controller architecture pattern has been chosen. System is divided into following parts:

1. Model - contains all classes describing logic of the system, ie. types of Events and Lists of members and lecturers
2. View - part dedicated for user interface
3. Controller - connection between view, controller and file manager, provides functionality of system
4. File Manager - handles secondary storage

That approach allows to develop parts of the system independently and combine them at the end of implementation. This architecture would also make unit testing easier if they were to be used.

Structure of the system can be seen on class diagram, the whole class diagram can be found in appendices (Appendix E).

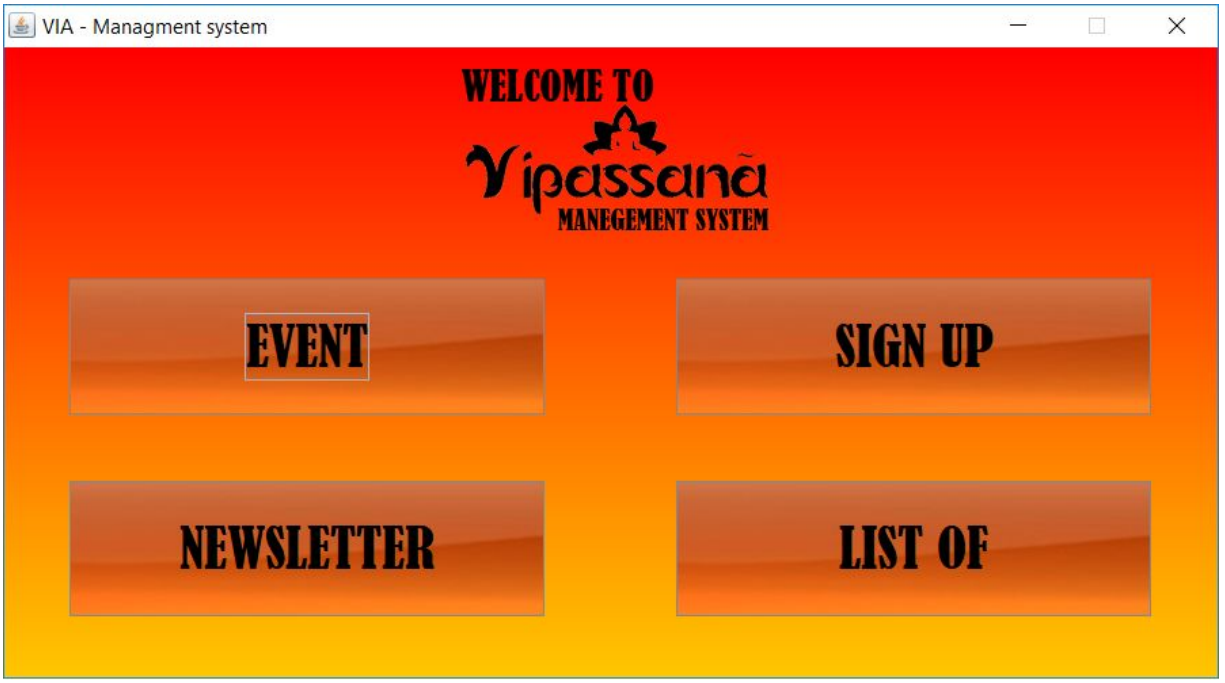


The system was developed in Java, using Swing library for GUI as it was introduced in the course and it is more beginner friendly than JavaFx. We haven't used Window Builder for GUI in order to achieve full control of what is displayed. Every part of the system has its own package, with group classes of one purpose in one place. Data are stored in binary files, saved using Java objects serialization.

One of the most extensively used design pattern in the system is Singleton. For instance, VIAController, VIAManager and VIAWindow classes are singletons, which manage work of controller, model and view. Another used design pattern is Facade, wto take an example VIAManager which is a hook point to all operations on the model.

Design of GUI was made having in mind simplicity and clarity. The application is divided into decision panels, form panels and lists. In decision panels user decides where to go next. Form panels are for creating events and signing up members and lecturers. List panels have informative function. They display content of our system.

See also User Guide in Appendix C.



4 Implementation

Implementation process started right after completing design, according to waterfall model. Work has been divided and one part of group started with model classes, when the other part was designing GUI and proceeded with its implementation. During this process, the design was constantly changing as to adjust to new ideas or requirements which came out, therefore activity and class diagrams had to be revised and updated.

One of the interesting code parts is the search engine which allows searching within MemberList, LecturerList and EventList by desired information. Searching is performed using regular expressions and Java standard library classes, Pattern and Matcher.

First, both searched string and compared string are converted to lowercase. The approach of using Pattern and Matcher is not standard since the string which is searched is Matcher (not Pattern). If desired string matches or appears at the end of string, function returns true. If not, string in which word is searched is cut by one character at the beginning, because of matching and hitEnd methods behavior. Method is executed recursively with the modified word, until it becomes shorter than the string which is looked for.

```
public static boolean search(String stack, String needle) {
    stack = stack.toLowerCase();
    needle = needle.toLowerCase();
    Pattern p = Pattern.compile(stack);
    Matcher m = p.matcher("");
    m.reset(needle);

    if (m.matches() || m.hitEnd()) {
        return true;
    } else if (stack.length() > needle.length()) {
        String newStack = stack.substring(1, stack.length());
        return search(newStack, needle);
    } else {
        return false;
    }
}
```

This method is used extensively during searching. Process starts when user enters desired expression in searching field. As response controller handle this event and calls appropriate method in Search Engine, ie. searchForLecturers(). Sequence of this operation can be seen in sequence diagram in Appendix E. Whole source code is attached in Appendix D.

5 Test

For testing the system has been checked continuously, especially after having added new features, and most of bugs were fixed before implementation moved to the next stage. Being finished, it was tested manually again by all group members, as well by one of the members friend, who is a professional manual tester, with ISTQB certificate. Moreover, unit tests were supposed to be introduced, however because of a strict schedule and the amount of new information to learn, it was postponed to the next project.

USE CASE	WORKING
The administrator must be able to sign up a new member storing information about him	**YES**
The administrator must be able to sign up a new lecturer	**YES**
The system has to search for lecturers of a given category	**YES**
The administrator must sign up participant to an event	**YES**
The administrator must generate newsletter	**YES**
The system must generate the list of members who has not paid	**YES**
The administrator must be able to plan events	**YES**
The administrator must be able to modify event data	**YES**
The administrator must be able to search for not finalized events	**YES**
Sign up participants and members to events	**YES**
System must count the prices	**YES**
System must return the amount of available places	**YES**
System must collect a list of lecturers who want fee or advertisement	**YES**

System must collect finalized events	**YES**
Users must be able to generate newsletter	**YES**
System has to generate a list of emails of members who have not paid	**YES**

6 Results and Discussion

The system is keeping track of events, members and lecturers. That is done by storing data of lists of members, lecturers and events in binary files. This is also what makes the system maintainable. In the case of generating and storing newsletters, text files are used with the intention to enable reading them directly. Providing an easy access to stored data, handling the customer's issue concerning searching for proper lecturers, as well as searching for specific members and events is provided by the search engine. The search engine is one of the most important functionalities presented by the system, especially concerning the fact that it is supposed to be storing more and more data in the future.

However, the system is not able to handle some of the expected functionalities, such as storing feedback or sending reminder emails to users. Furthermore, the system lacks in searching for events in a specific time period. However, this functionality could be implemented without any difficulties. One of the solutions could be adding a method of type boolean to MyDate class with parameters startDate and endDate. This method would check whether the startDate of the event is in that specific time period by comparing the values of given days, months and years. Finally, it would be called in the search engine. Nevertheless, another function has been added as a substitute. It is searching for finalized and not finished events. What it returns is a list of all events that are finalized and upcoming.

Nevertheless, in general the final outcome of the project can be stated as positive as it meets all of the essential requirements and is created to be easily maintainable.

7 Conclusion

The main aim of the project was to make a maintainable system that provides Vipassana easy access to information, as well as provides solution for the organisation's struggle concerning finding proper lecturers. To do so, requirements have been stated and then presented as use cases. As the Test section shows, all of the use cases have been performed with a positive outcome, what means that all of the requirements have been fulfilled. Analysis is strictly connected with the design, which leads directly to implementation. What that means is that the documentation is consistent with the system and that the system has been built basing on analysis and design. This leads to the conclusion, that the purpose of analysing and designing the system has been archived. Also the implementation has been accomplished with a success, it has been checked by both all group members and a professional manual tester and it has been declared that it satisfies all of the organisations need. Moreover hardly any bugs in the code has been found.

8 Project future

If the project had been continued, the first things done would be implementing the delimitations. That means using database to store data, storing feedback and using it while planning events, sending automatically reminder emails about payments to users, searching for events between a specific time period and providing data from real users. To add the last feature the system would have to not be a single user system anymore. Users would have to be able to access it from their own computers.

9 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].

10 List of Appendices

Appendix A Project Description - Project_Description.pdf

Appendix B Group Contract - Group_Contract.pdf

Appendix C User Guide - UserGuide.pdf

Appendix D Source code - SDJ1.zip

Appendix E Diagrams as Astah project - Vipasanna-Manegement-System.asta

Appendix F Web Site source code - RWD.zip