PROJECT REPORT

Vipassanā– Insight Awareness (VIA) System and Website

Supervisors

Mona Wendel Andersen

Michael Viuff

VIA University College

VIPASSANĀ InsightAwarenes

Students

Andrei Cioanca – 266105

Stefan Harabagiu – 266116

Nikita Roshkov – 266900

Dawei Li – 269053

Characters – 1233312

ICT Engineering 1st Semester

11/12/2017

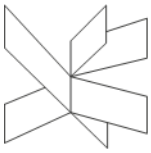
Students

Andrei Cioanca – 266105
I hereby declare that my project group and I prepared this project report and that all sources of information have been duly acknowledged.

Stefan Harabagiu – 266116
I hereby declare that my project group and I prepared this project report and that all sources of information have been duly acknowledged.

Nikita Roshkov – 266900
I hereby declare that my project group and I prepared this project report and that all sources of information have been duly acknowledged.

Dawei Li – 269053
I hereby declare that my project group and I prepared this project report and that all sources of information have been duly acknowledged.



Acknowledgements

The team wishes to formally acknowledge VIA University College's active implication in the evolution and completion of this project work. All documents contained or referred to from this project are based on VIA University College's templates and designs. All information present in this document, while original in itself, could not have been made possible without VIA's designs and templates.

Version history

Current: Version 1.8.1

Version 1.0.0 – 27.11.2017

Version 1.1.0 – 04.12.2017

Version 1.2.0 – 05.12.2017

Version 1.3.0 – 11.12.2017

Version 1.4.0 – 13.12.2017

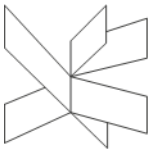
Version 1.5.0 – 15.12.2017

Version 1.6.0 – 17.12.2017

Version 1.7.0 – 18.12.2017

Version 1.8.0 – 19.12.2017

Version 1.8.1 – 19.12.2017



Abstract

Vipassanā - Insight Awareness (VIA) is a center for spiritual events and practices. They currently lack any kind of system that keeps track of information about events, members or lecturers. They also lack any kind of presence online and have approached our developer team in order for us to create a robust system that complies with their requirements and also a website that would make their presence online known.

The website has been developed rather quickly with a very clean look and neutral colors. It is fully responsive working on most, if not all devices correctly. The website was made based on some early sketches provided by VIA themselves to the developer team.

The system has been developed in Java and incorporates early concepts of primitive database design. It fulfills all of VIA's requirements and stands as a robust and easily manageable system that can hold the company's amount of data.

Based on their requirements, the system can keep track of Events, Members and Lecturers, each being able to be added, edited, or removed. The system makes use of file storage in order to keep track and not lose the information from one application usage session to another.

The system also incorporates a GUI, made simple and clean in order for VIA's staff to navigate it easily. It facilitates the use of the system by providing an intuitive interface for interaction.

In the end, VIA received a robust and malleable program coupled with a clean website in order to further their business endeavors.

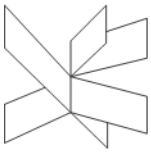


Table of Figures

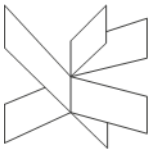
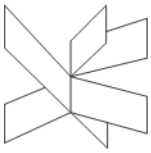


Table of Contents



I. Introduction

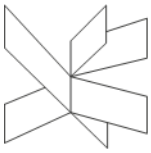
Vipassanā - Insight Awareness (VIA) is a center for spiritual events with a strong base in Buddhist practices of meditation and spirituality. Its' purpose is to offer people a safe heaven where they can hold and take part in different spiritual events of many creeds (VIA Interview). Today's events at VIA, amongst the spiritual kind, also include practices not directly linked to any religious agendas such as dream interpretation, healing, astrology, reincarnation or karma. The organization also holds a large number of lectures, workshops, seminars and through partnerships with third party organizations also trips (Example of an Event).

Thus said, the company lacks any kind of system that holds or retains the information they need in order to run and organize all the events properly. Vital information such as start times, dates, attendees, lecturers or sponsor names get lost with the day-to-day activities and practices the organization delves in. The employees usually have a hard time remembering certain details about the lectures or workshops such as specific food requirements or simple names or subjects the lectures might contain.

VIA also has issues with creating a stable online presence. A high percentage of the target audience for these kind of spiritual events remains untapped due to VIA having a largely unknown presence to its local community both online and in real life.
They currently have no way of advertising themselves online or retaining information about their current members, lecturers or potential sponsors.

A system tailored specifically to VIA's requirements even if at best would only faintly resemble a true database, would contribute immensely to the organization's day-to-day activities. Having such a system at their disposal would not only give the employees a much-needed break when it comes to remembering certain details but it would also give VIA the opportunity to focus their efforts on improving itself as a company and give a greater level of satisfaction to their customers.

The team's objective as developers is to create this system from scratch and tailor it to the organization's specific needs. The team aims to implement the baseline requirements provided by the company's leadership which include basic information storage and readability, while also embellishing the system with much needed quality



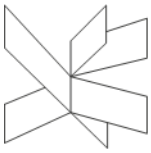
of life changes such as the ability to store starting dates for events, duration, what lecturers will take part in it and so on and so forth.
The team's primary objective is to provide VIA with a stable and robust system that not only would fulfill their requirements but also improve on them significantly with what their employees need in order to facilitate day-to-day activities.

This task does come with its' fair share of technical difficulties that would need to be tackled in a professional manner by the developer team. Obstacles such as file usage in secondary storage and providing employees with a readable and easily understandable and distinguishable GUI are not only tackled during development but also detailed in this project report. Creating a system from scratch and delivering it to the stakeholder in itself is a technical difficulty and every substantial development in the program is documented and thoroughly tested.

As for project delimitations, as junior developers the team will not handle any kind of linkage between the customer's ordered website and the system itself even though either of their developments have taken inspiration from one another. While the group retains the responsibility of having a simple, easy to understand and most importantly functional UI, it will not make a priority of having a complex and detailed one seeing as the endeavor would largely fall short of the focus of this project. Any guidance towards the usage of the system will be done remotely through the team's provided user guide, the team won't divert resources to teaching the end-user how to operate the system manually. On this note, the actual deployment of the system will also not be handled by the junior developer team.

Even though legal issues have a fair positive percentage in the development cycle of the website and system, the team will not pursue any endeavor of tackling them whatsoever.
Expectancy of resolution on part of the developer team on these issues is understandable but what is more important is the fact that they currently fall out of the primary focus of this project and that is handling the complete development of a single user system and website alongside any documentation it requires.

The development itself started with a list of functional and non-functional requirements, the baseline of the system and a blueprint to what the system should be able to do and potentially not do according to the stakeholder's needs. They are both detailed in the next section of this report.

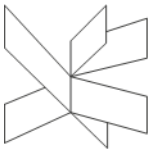


II. Functional and non-functional requirements

The requirements overall represent what the employer Vipassanā - Insight Awareness would like the system to do overall. Due to VIA's untrained status in software development the team has taken a few liberties with some specific requirements, adding quite a handful to ease development and overall build a more stable and robust system.

II.1 Functional requirements

1. The user should be able to create a new event.
2. The events can be of four different types (lectures, seminars, workshops and trips).
3. The system should store the following information about the lectures: a title, start date, start time, duration, lecturer, 1 subject, sponsor name, price, finalized or not, total number of tickets, discount.
4. The system should store the following information about the lectures: a title, start date, start time, duration, lecturer, 1 subject, sponsor name, price, finalized or not, total number of tickets, discount.
5. The system should store the following information about the workshops: a title, start date, start time, duration, lecturers, food included (vegan or not), price, finalized or not, total number of tickets, discount.
6. The system should store the following information about the trips: a title, start date, start time, duration, locations, price, finalized or not, total number of tickets, discount.
7. The user should be able to search events by: finalized or not, start date, subject, price, lecturers, sponsors.
8. The user should be able to modify every aspect of an event.
9. The user should be able to store members' information.
10. Members are defined by name, email, address, phone, payment year, date of registration, newsletter subscription, attended events.
11. The user should be able to search members by name, payment year, date of registration, attended events.
12. The user should be able to update the information of each member.
13. The user should be able to store lecturers' information.
14. Lecturers are defined by name, email, phone, sponsor or not, subject.
15. The user should be able to search lecturers by name, subject, email, phone, sponsor or not.
16. The user should be able to update the information of each lecturer.



II.2 Non-functional requirements

1. The system has to be implemented in Java.
2. The system has to be compatible with Microsoft Windows 7,8,10.
3. The system needs to be able to run with no complications for long periods of time.
4. The system should use files for secondary storage only.

ID	Requirement	Use Case reference	Test reference
1	Functional.1	Store event	Test.SYSTEM.1
2	Functional.2	Store event	Test.SYSTEM.2
3	Functional.3	Store event	Test.SYSTEM.3
4	Functional.4	Store event	Test.SYSTEM.4
5	Functional.5	Store event	Test.SYSTEM.5
6	Functional.6	Store event	Test.SYSTEM.6
7	Functional.7	Search event	Test.SYSTEM.7
8	Functional.8	Edit & Delete event	Test.SYSTEM.8
9	Functional.9 & Functional.10	Store member	Test.SYSTEM.9
10	Functional.11	Search member	Test.SYSTEM.10
11	Functional.12	Edit member	Test.SYSTEM.11
12	Functional.13 & Functional.14	Store lecturer	Test.SYSTEM.12
13	Functional.15	Search lecturer	Test.SYSTEM.13
14	Functional.16	Edit lecturer	Test.SYSTEM.14

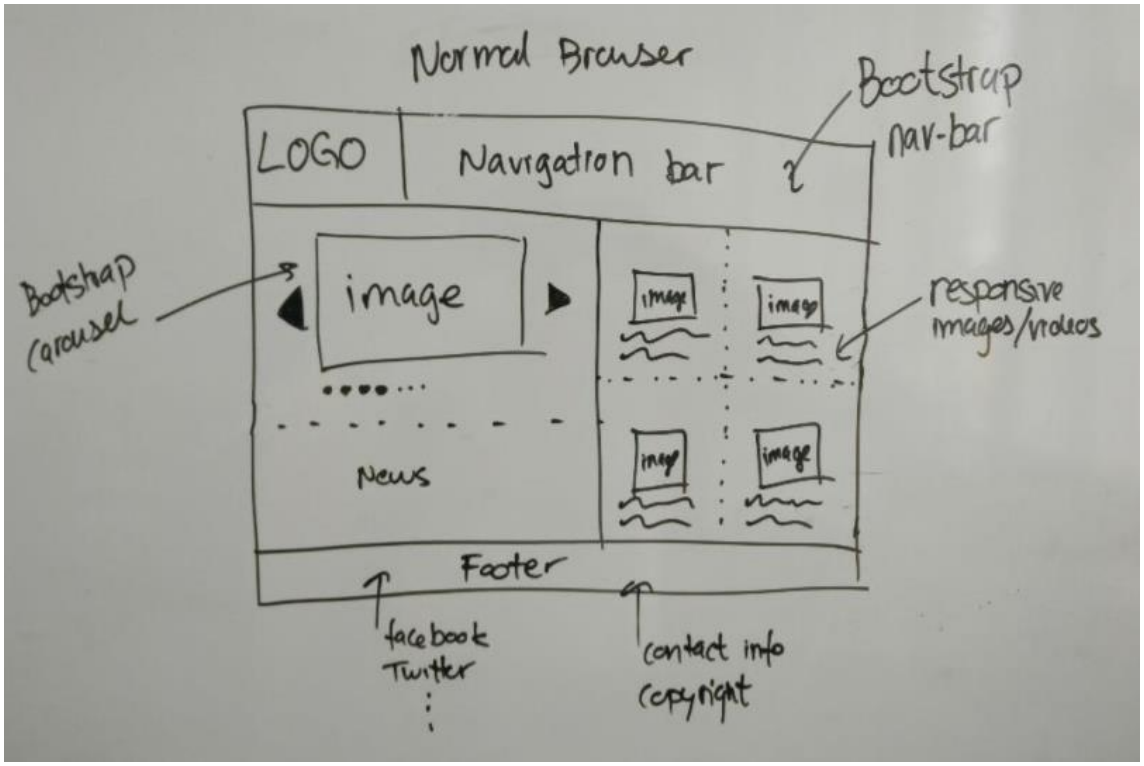
The team feels these requirements are suitable for Vipassanā, making the system complex in nature while retaining a robust and clean structure with good maintainability. These requirements are better explained and detailed in the next section of the project report, the analysis, where the group presents the system and website itself through diagrams that would aid the uninitiated in understanding the team's design.

Due to the fact that the website is not the central focus of this project, the team decided to restrict its presence and not present it as extensively as the system.

III. Website Analysis

The sketch below has been provided by VIA to serve as an example of what they would like their website to look like.

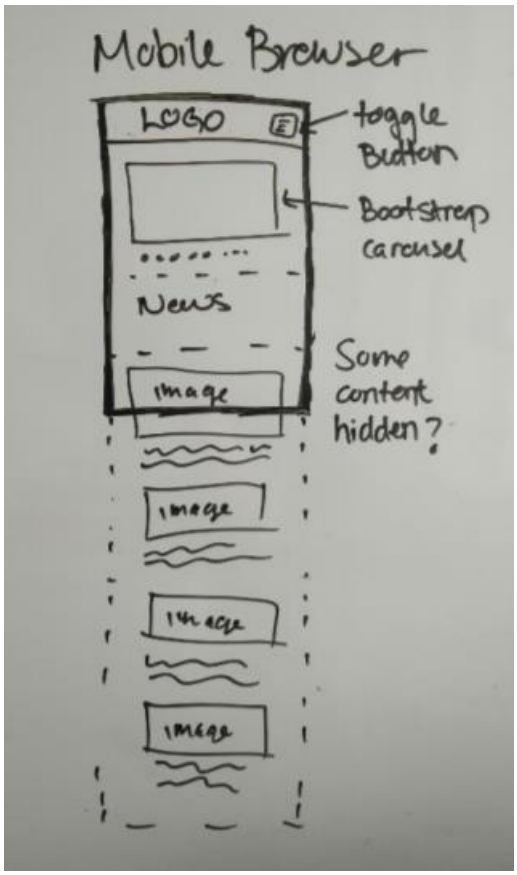
Figure 1 – Website Sketch 1



As you can see from the sketch the members have a very solid foundation of what the website would look like. Apart from a few details, such as the size of different elements within the website or having more pages than usual to aid with the structuring of information, the final website faithfully kept the initial design.

VIA also required that the website be responsive meaning that all the different elements of the website will not become unusable or unreadable upon modifying the window size or switching the device the website is viewed from. In this regard they have also provided another sketch in which they detail how they would like the website to be viewed on a mobile phone.

Figure 2 – Website Sketch 2



Pretty straightforward in terms of design, nothing special apart from a few “carousel” type elements.

Having both of these initial sketches helped the team tremendously in developing the website. Having been spared the process of coming up with interesting designs and present them to the employer before any actual development was done, the team could focus on implementing the actual website and working on providing VIA with the online identity it lacked for such a long time.

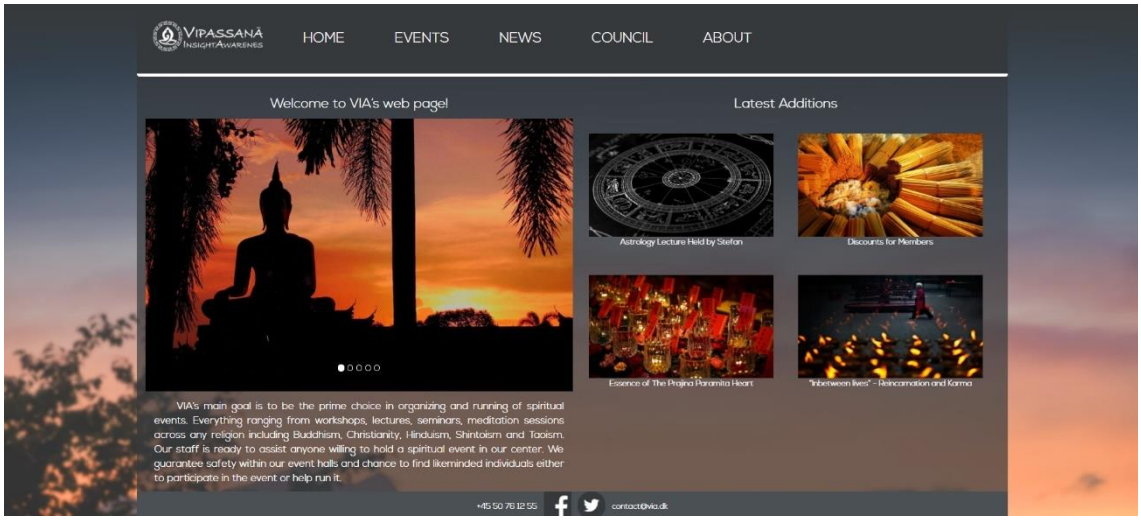
To do all this, the group came up with a few interesting design choices that are detailed in the following section.

IV. Website Design

When designing the website the team already knew that responsiveness on different devices and window sizes would play a key part in the product’s overall presentation. With this information in mind the team decided that implementing it with bootstrap would be the best design choice.

As you can see from the picture below, the team's design does not differ that much from what was provided by VIA.

Figure 3 – Website Home Page



The team have maintained the look of the navigation bar, with its logo to the left and buttons to the right, the group has also maintained the structure of the main content, having the moving carousel element to the left side of the screen and the "Latest Additions" section to the right.

The aesthetics of the page are what the team came up with since the stakeholder gave us freedom over the looks. The developers gave the website a neutral look, with a black and white colour scheme with a grey sliver underneath, overlooking a beautifully contrasted background image.

As for the other pages, the group has applied the same design principles, keeping VIA's requirements in focus at all times.

Figure 4 – Website Events Page

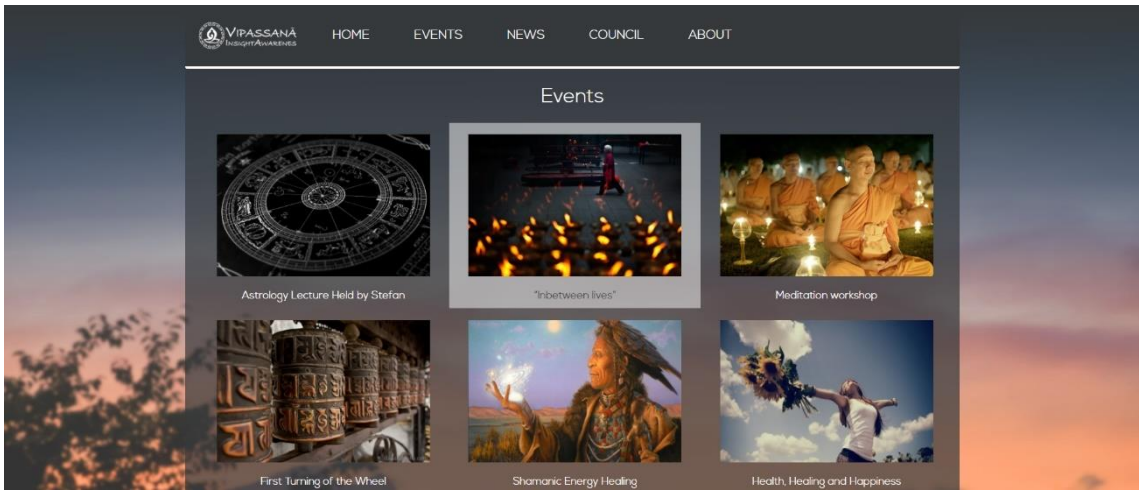


Figure 5 – Website Council Page

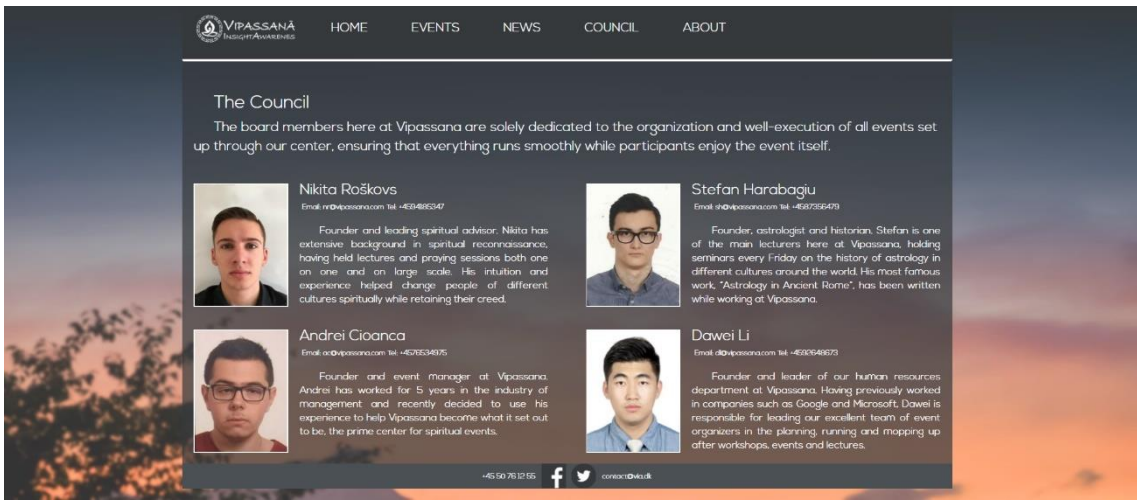
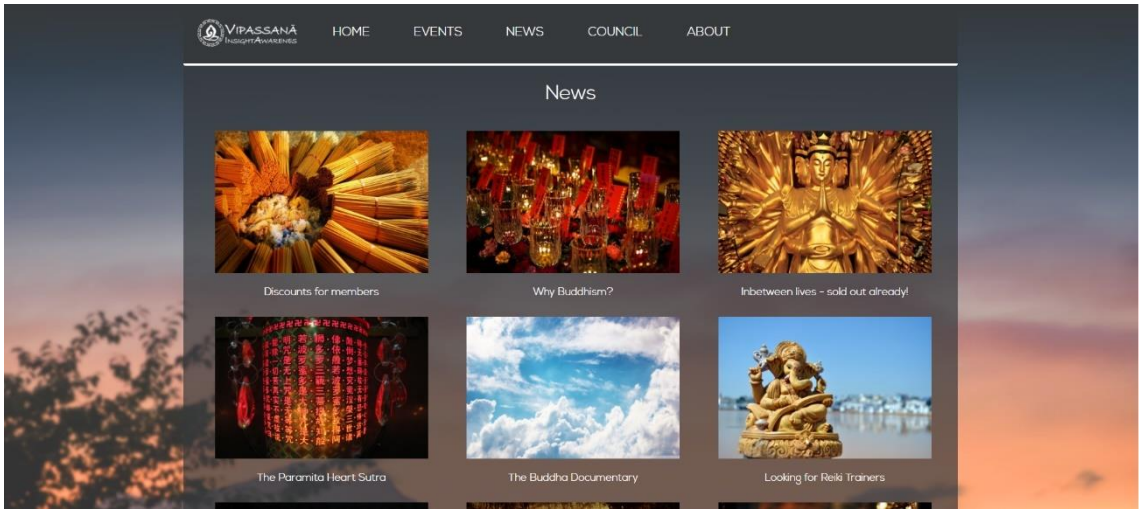


Figure 6 – Website News Page



Due to the fact that the website isn't the main focus of the project and that the developers have already demonstrated its' abilities to the stakeholders, the group has purposefully omitted the implementation part of this report.

V. System Analysis

The diagram below shows a basic representation of the system and how it overall functions.

Figure 7 – VIA Basic Functionality - Events

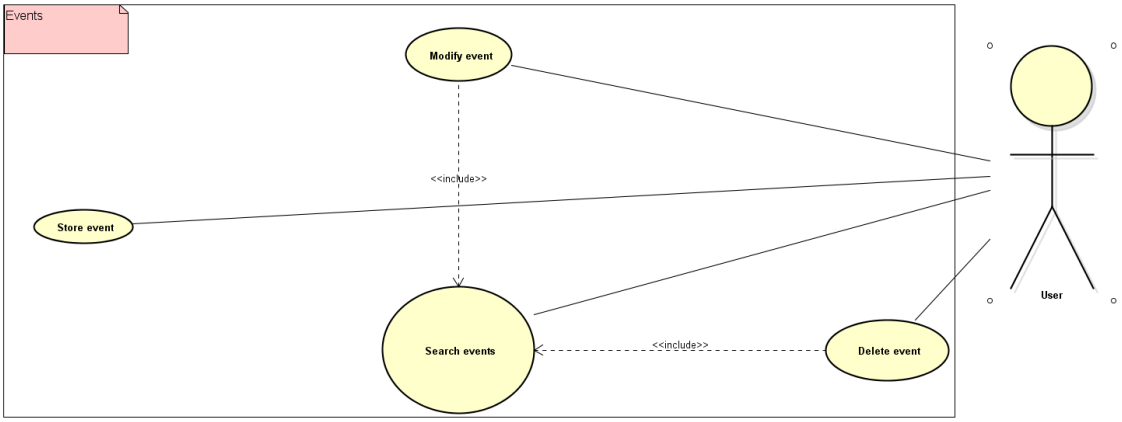


Figure 8 – VIA Basic Functionality - Members

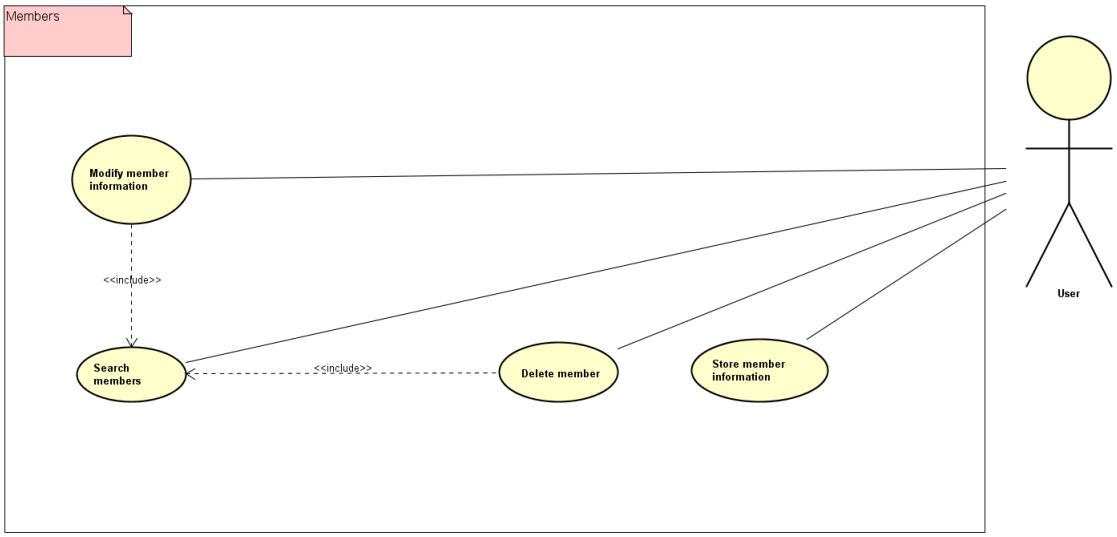
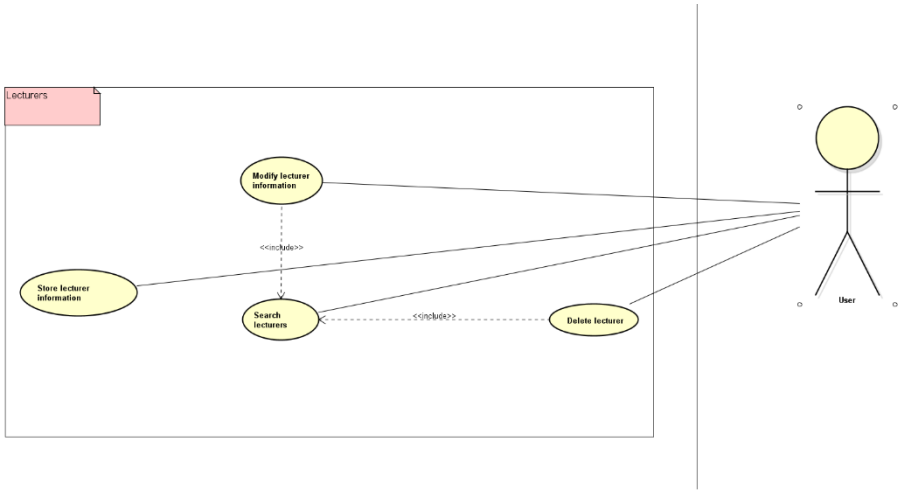


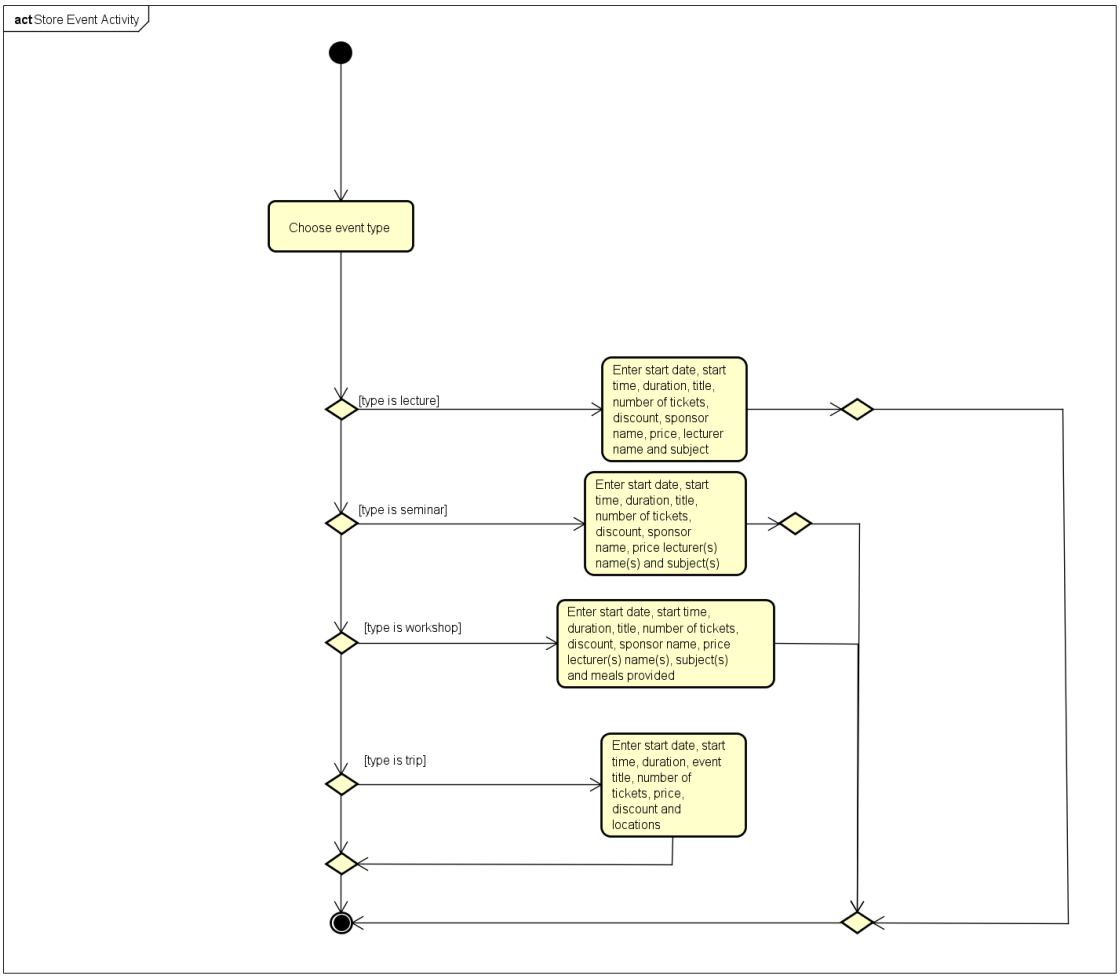
Figure 9 – VIA Basic Functionality - Lecturers



As it is seen in the diagram, the user is able to access three distinct parts of the system, the member list, the lecturers list and the event list. Each part of the system will allow him to add, remove, edit and search by a parameter for a member, an event or a lecturer. The diagram shows all of the basic requirements stated by the stakeholder.

The team has also provided a diagram description of the same process.

Figure 10 – Activity Diagram for Storing an Event



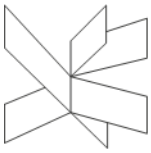


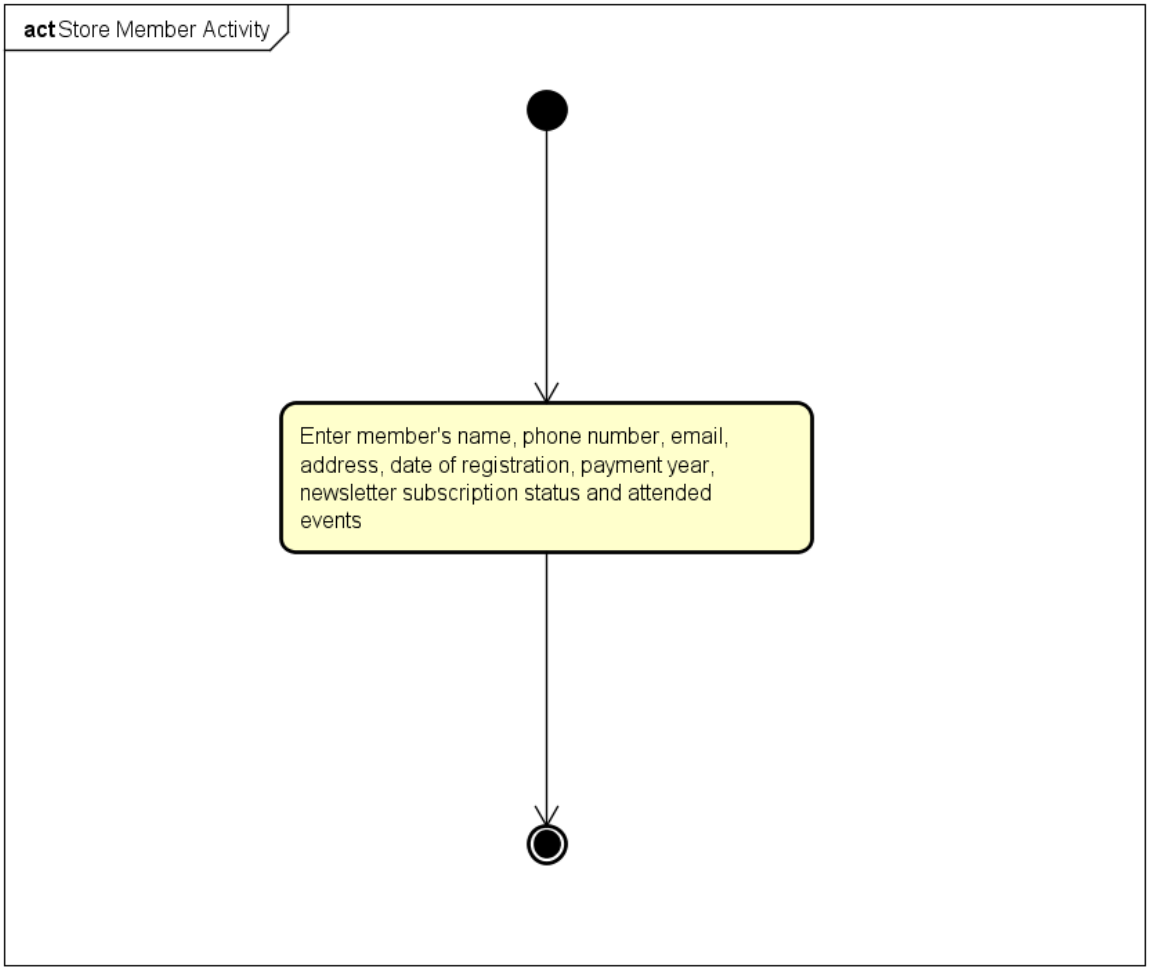
Figure 11 – Use-Case Description for storing an event

ITEM	VALUE
UseCase	Store event
Summary	The user creates an event on a given date.
Actor	User
Precondition	None
Postcondition	An event has been created and data for the event, lecturer and members is saved in the system.
Base Sequence	1. Choose the type of event that is to be created. 2. If the type is lecture enter start date, start time, duration, event title, number of tickets, discount, sponsor name, price, lecturer, subject. 3. If the type is seminar enter start date, start time, duration, event title, number of tickets, discount, sponsor name, price, lecturers and subjects. 4. If the type is workshop enter start date, start time, duration, event title, number of tickets, sponsor name, discount, provided meals(yes or no), vegan food(yes or no), lecturers and subjects. 5. If the type is trip enter start date, start time, duration, event title, number of tickets, price, discount, locations. 6. Verify the data and store it in the system
Branch Sequence	
Exception Sequence	No available lecturers in the given time interval: use case ends
Sub UseCase	
Note	

Following the diagram from its entry point you can begin to understand how the system stores its events. The only complexity that may arise from the diagram would be the branching of flows depending on whether the event is a lecture, a seminar, a workshop or a trip. This representation should offer a solid understanding of how the most critical part of the system functions, the storing of events.

For comparison in terms of similarity and complexity the team has also provided a diagram for adding a member in the system.

Figure 12 – Activity Diagram for Storing a Member



Both processes are strikingly similar only diverging in terms of programming logic approach and order. There is no denying that adding a member is made very simple by the fact that there are only one kind of members and not many different ones as is the case for events.

Below is an updated description of the same process of adding a member in the system that should paint a clearer picture of how the process works.

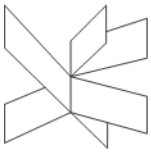
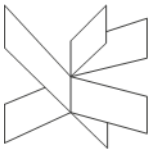


Figure 13 – Use-Case Description for Adding a Member

ITEM	VALUE
UseCase	Store member information
Summary	The user enters a member's information into the system.
Actor	User
Precondition	None
Postcondition	
Base Sequence	1. Enter the member's name, phone number, email, address, date of registration, payment year, newsletter subscription status and attended events 2. Verifiy the data and store it in the system
Branch Sequence	
Exception Sequence	
Sub UseCase	
Note	

The base sequence is executed in order, from 1 to 2, when the user wishes to add a new member. This is similar to how all the other parts function such as removing a member, adding an event, etc.

Following the analysis of the system and the team's initial diagrams they decided upon the following design choices.



VI. System Design

The team knew from the start that the system would make use of a GUI but seeing as this was the first time the team dealt with anything remotely resembling a user interface the group decided to start off by making the whole system work on its own without any aid from a GUI.

Naturally the team began expanding on the basic functionality diagram and it came up with a complex UML diagram that served as the basis of their implementation. Due to its sheer size the developers have decided to show glimpses of it here, such as the event list and event classes as well as all of the latter's children.

Figure 14 – UML Diagram Events 1

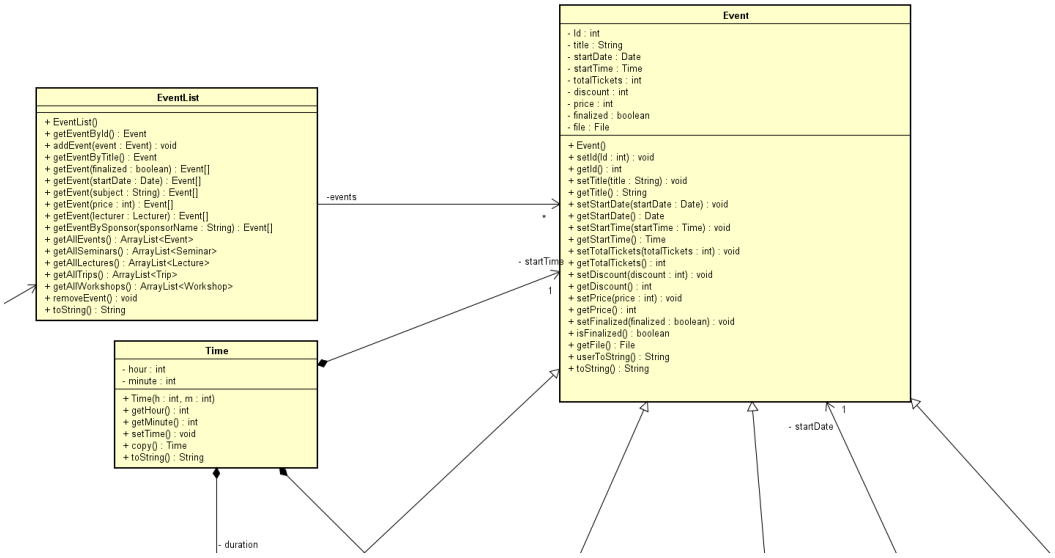
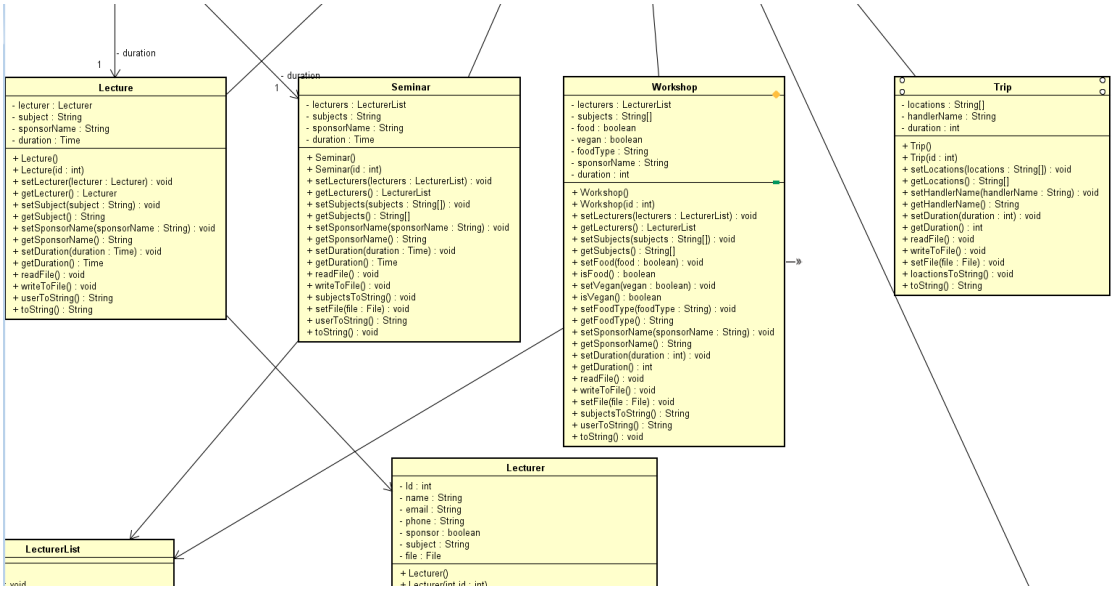


Figure 15 – UML Diagram for Events 2



The full UML diagram is available in the appendices section of this report.

The team has structured the system, without the GUI, in 15 different classes. They include a separate class for dates the system uses, a separate class for time and a unique identifier generator class which automatically assigns a UID to any member, lecturer or event when they are created.

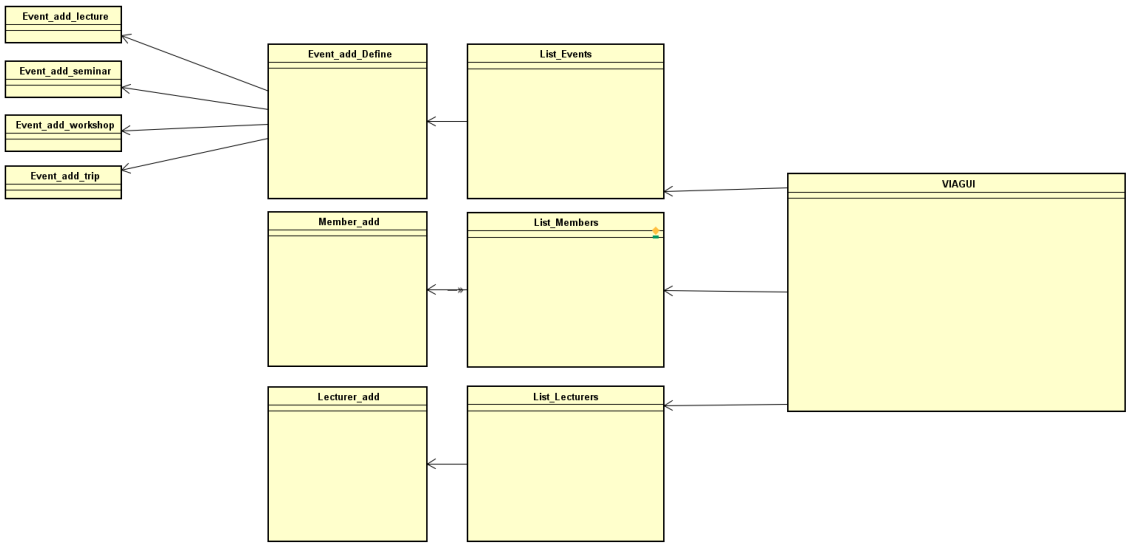
From the start the group made some interesting design choices. For example, the class Event isn't used for every event in the system, instead it has four different children titled Trip, Seminar, Workshop and Lecture. Each child holds specific data available only to that respective kind of event. Workshops are the same thing as Lectures but they can have multiple lecturers and subjects while Lectures only have one of each.

Apart from this, all classes look and act mostly the same. Members and lecturers each have a Member List class and Lecturer List class which helps the system build specific lists of the respective objects whenever needed. For example, the system might need to list a few Members by a specific parameter, it first compares all the existing members with that parameter and if it finds them positive then it creates a list and adds them to it, then it simply displays the list.

The same thing happens with events, only that they are more information heavy than the other two. All classes contain setters and getters so that everything about them can be manipulated.

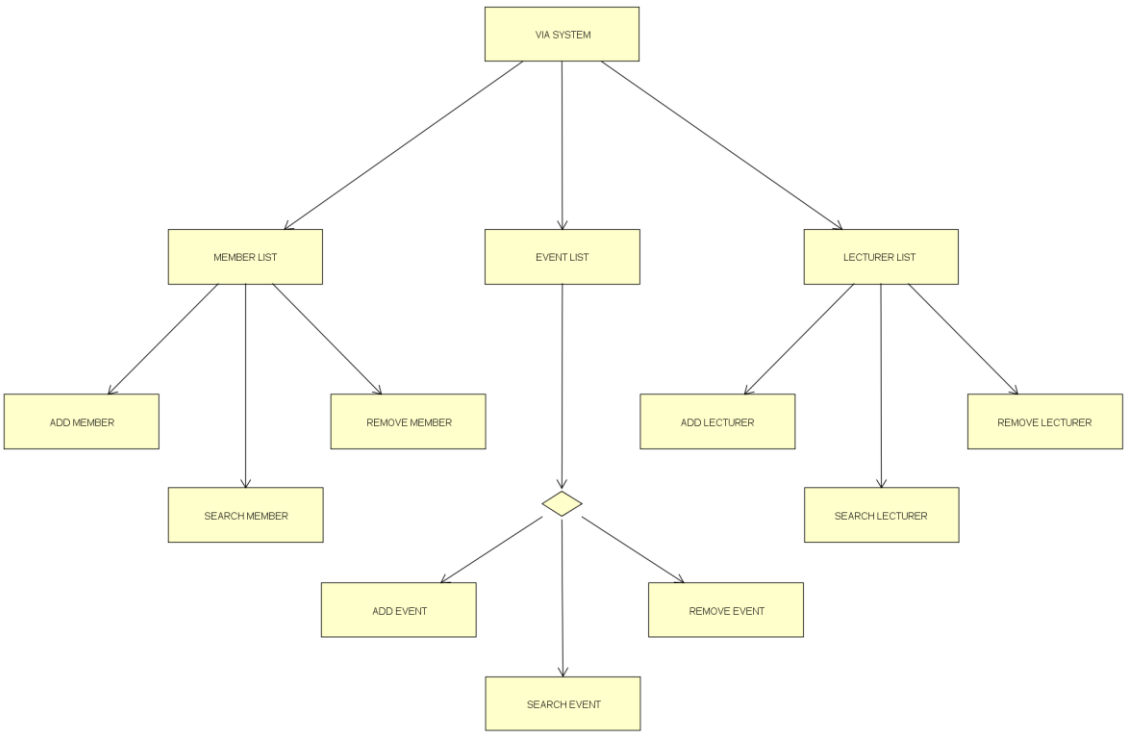
Naturally the system is accompanied by a GUI as well. A very basic model was adopted with no added complexity or any other embellishments in order to keep it as “clean as possible”. The diagram below shows the basic functionality of the GUI.

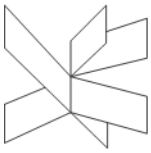
Figure 16 – Basic GUI Functionality 1



The diagram above shows the basic functionality of the GUI.

Figure 17 – Basic GUI Functionality 2





As you can see the boxes can be thought of as “buttons” and the arrows connecting them between each other as “access points”. For example, you can access the LECTURER LIST part of the system through a button located in VIA SYSTEM.
In order to add a new MEMBER in the system, the user would need to access the MEMBER LIST from VIA SYSTEM and then click on ADD. Hopefully, through these simple diagrams and explanations you now have a better understanding of how the UI works and how it is connected to the system.

VII. System Implementation

The system developed by the group consists of three different lists of objects that represent the events hosted by the company, the lecturers hired by them and the members of their organization.
Figure 18 – Implementation 1

```
public class VIASystem
{
   public static MemberList members = new MemberList();
   public static LecturerList lecturers = new LecturerList();
   public static EventList events = new EventList();


}
```

The GUI for the system was created using a Java Swing GUI builder provided by the NetBeans IDE.
When the user runs the application, the system will check to see if there are any files that store previous data and will read them in order to initialize the lists. The system will instantiate the required objects and use a FileController class to access and call the *readFile* methods for each *member, lecturer* or *event* object.
Figure 19 – Implementation 2

```
try
{
FileController.readEvents();
}
catch(NullPointerException e)
{
   e.printStackTrace();
}
```

```
try
{
FileController.readLecturers();
}
catch(NullPointerException e)
{
   e.printStackTrace();
}
```

```
try
{
FileController.readMembers();
}
catch(NullPointerException e)
{
   e.printStackTrace();
}
```

As stated before, the methods called on the FileController class instantiates as many objects as needed and then calls the required method for that object:

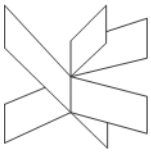


Figure 20 – Implementation 3

```
public static void readLecturers() throws IOException
{
   File file = new File("./lecturers");
   File[] files = file.listFiles();
   for(int i = 0; i < files.length; i++)
   {
      Lecturer lecturer = new Lecturer(1);

      lecturer.setFile(files[i]);
      lecturer.readFile();
      VIASystem.lecturers.addLecturer(lecturer);
   }
}
```

Then the user will be presented with three different choices: Events, Lecturers or Members. If he clicks on any of those buttons he will be taken to another frame of the UI that will display all the objects as a list. The team used a *jList* to achieve this in which the programmers stored objects and then used a custom made ListCellRenderer class in order to control what information will be displayed.

Figure 21 – Implementation 4

```
// getting events

Event[] eventsArr = new Event[VIASystem.events.getAllEvents().size()];
for (int i = 0; i < VIASystem.events.getAllEvents().size(); i++)
{
   eventsArr[i] = VIASystem.events.getAllEvents().get(i);
}

jList1.setListData(eventsArr);
jList1.setCellRenderer(new EventListCellRenderer());
```

Figure 22 – Implementation 5

```
public class EventListCellRenderer extends DefaultListCellRenderer
{
   public Component getListCellRendererComponent(JList list, Object value,
         int index, boolean isSelected, boolean cellHasFocus)
   {
      if (value instanceof Event)
      {
         value = ((Event) value).getTitle();
      }
      super.getListCellRendererComponent(list, value, index, isSelected,
            cellHasFocus);
      return this;
   }
}
```

In order to create another event, the user only needs to click on the add button which will take him to a new frame in which he will have to select what type of event he wishes to create (lecture, seminar, workshop or trip). After clicking on one of the four buttons a new frame will open in which he will enter the data required to create that type of event (the *Lecture, Seminar, Workshop* and *Trip* classes are all children of the *Event* class). After finishing writing the date the system will instantiate a new object and add it to the *events* list. When the user chooses to close the system the FileController class will be used again to write everything that has been created up until now to files.

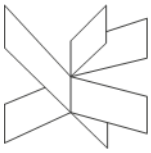
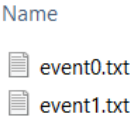


Figure 23 – Implementation 6

```
FileController.writeEvents();
FileController.writeLecturers();
FileController.writeMembers();
System.exit(0);
```



The system also allows the user to search the list by several parameters. Depending on what parameter the user chose the system uses specific code to search the list and display each element.

Figure 24 – Implementation 7

```
else if(jComboBox1.getSelectedItem() == "Start date")
{
   try
   {
      String[] numbers = jTextField1.getText().split("\\.");
      int[] results = new int[numbers.length];
   for (int i = 0; i < numbers.length; i++)
   {
      results[i] = Integer.parseInt(numbers[i]);
   }

     Date date = new Date(results[0], results[1], results[2]);
     eventsArr = VIASystem.events.getEvent(date);
     jList1.setListData(eventsArr);
   }
   catch(NumberFormatException e)
   {
      JOptionPane.showMessageDialog(null,
            "Please enter a valid input (use only numbers and dots) \nE.g: 15.12.2017");
   }
}
```

Furthermore, the system generates a unique ID for every event, lecturer or member and uses them in the naming of their files and for easily distinguishing each object from another.

Figure 25 – Implementation 8

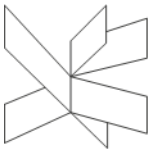
```
public class UID
{
   private static int member_id = 0;
   private static int lecturer_id = 0;
   private static int event_id = 0;

   public static int generateMemberId()
   {
      return member_id++;
   }

   public static int generateLecturerId()
   {
      return lecturer_id++;
   }

   public static int generateEventId()
   {
      return event_id++;
   }
```

An example for the usage of these IDs is a *member's* list of attended events. Instead of creating a list of *event* objects the system instead stores the ID of the event. When creating a member for example the user chooses which events the member attended



and the *member* object is being attributed and ArrayList consisiting only of the ID of each event.

Figure 26 – Implementation 9

```
// getting attended events
boolean[] values = new boolean[table.getRowCount()];
TableModel tm = table.getModel();
ArrayList<String> attendedEvs = new ArrayList<String>();
for (int i = 0; i < tm.getRowCount(); i++)
{
   Object o = tm.getValueAt(i, 0);
   values[i] = (boolean) o;
   if (values[i] == true)
   {
      Object obj = table.getValueAt(i, 1);
      Event event = (Event) obj;
      for (int j = 0; j < VIASystem.events.getAllEvents().size(); j++)
      {
         if (VIASystem.events.getAllEvents().get(j).getTitle()
               .equals(event.getTitle())
               && VIASystem.events.getAllEvents().get(j).getId() == event
                     .getId())
         {
            attendedEvs.add(String.valueOf(
                  VIASystem.events.getAllEvents().get(j).getId()));
```

VIII. Test

The system was tested according to the requirements that the group wrote in order to assure the system's functionality. The GUI was tested as well to ensure that it follows what was specified in the Use Case Descriptions.

GUI

I. Store event

1. The user first chooses the type of event he wishes to add
2. The user enters the required data into the system
3. The system verifies the data and stores it

PASSED

II. Search events

1. The user enters a search parameter
2. The system checks and lists every event that meets the requirements
3. If there is no event with that requirement, the list displayed will be empty

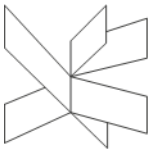
PASSED

III. Modify event

1. The user selects the desired event
2. The user enters new data
3. The system verifies the data and stores it

PASSED

IV. Delete event



1. The user selects the event that needs to be deleted
2. The system deletes the event

PASSED

V. Store member information

1. The user enters the required data into the system
2. The system verifies the data and stores it

PASSED

VI. Search members

1. The user enters a search parameter
2. The system checks and lists every member that meets the requirements
3. If there is no member with that requirement, the list displayed will be empty

PASSED

VII. Modify member information

1. The user selects the desired member
2. The user enters new data
3. The system verifies the data and stores it

PASSED

VIII. Delete member

1. The user selects the member whose information needs to be deleted
2. The system deletes the member's information

PASSED

IX. Store lecturer information

1. The user enters the required data into the system
2. The system verifies the data and stores it

PASSED

X. Search lecturers

1. The user enters a search parameter
2. The system checks and lists every lecturer that meets the requirements
3. If there is no lecturer with that requirement, the list displayed will be empty

PASSED

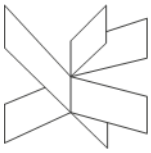
XI. Modify lecturer information

1. The user selects the desired lecturer
2. The user enters new data
3. The system verifies the data and stores it

PASSED

XII. Delete lecturer

1. The user selects the lecturer whose information needs to be deleted



2. The system deletes the lecturer's information
   PASSED

SYSTEM

1. The user can create a new event (**PASSED**)
2. Events can be of four different types (**PASSED**)
3. The system should store about lectures: a title, start date, start time, duration, lecturer, 1 subject, sponsor name, price, finalized or not, total number of tickets, discount (**PASSED**)
4. The system should store about seminars: a title, start date, start time, duration, lecturers, subjects, sponsor name, price, finalized or not, total number of tickets, discount (**PASSED**)
5. The system should store about workshops: a title, start date, start time, duration, lecturers, food included (vegan or not), price, finalized or not, total number of tickets, discount (**PASSED**)
6. The system should store about trips: a title, start date, start time, duration, locations, price, finalized or not, total number of tickets, discount (**PASSED**)
7. The user should be able to search events by: finalized or not, start date, subject, price, lecturers, sponsors (**PASSED**)
8. The user should be able to modify every aspect an event (**PASSED**)
9. The user can store a member's information (name, email, address, phone, payment year, date of registration, newsletter subscription, attended events) (**PASSED**)
10. The user should be able to search members by name, payment year, date of registration, attended events (**PASSED**)
11. The user should be able to update the information of each member (**PASSED**)
12. The user can store a lecturer's information (name, email, phone, sponsor or not, subject) (**PASSED**)
13. The user should be able to search lecturers by name, subject, email, phone, sponsor or not (**PASSED**)
14. The user should be able to update the information of each lecturer (**PASSED**)

Besides using the final version of the system to test its functionality there is a tester class created in order to check for more specific errors.

Figure 27 – Implementation 10

```
public class VIATester
{
   public static void main(String[] args)
   {

      Lecture lecture = new Lecture();
      Date date = new Date(15, 2, 2018);
      Time time = new Time(18, 15);
      Time duration = new Time(2, 30);
      Lecturer lecturer = new Lecturer(1);
      Lecturer lecturer2 = new Lecturer(1);
      Lecturer lecturer3 = new Lecturer(1);
      Lecturer lecturer4 = new Lecturer(1);
      Lecturer lecturer5 = new Lecturer(1);
```

```
System.out.println();
System.out.println(seminar.toString());

for(int i = 0 ; i < lecturers.getAllLecturers().size(); i++)
{
   System.out.println(lecturers.getAllLecturers().get(i).getId());
   System.out.println();
}

System.out.println("//////////////////////");
System.out.println();

for(int i = 0; i < events.getAllEvents().size(); i++)
{
   System.out.println(events.getAllEvents().get(i).getId());
   System.out.println();
}
```

```
//////////////////////

0

1

2

3
```

```
2
Hara3333
15 02 2018
18 15
10120
310
1000
true
About Hara
Hara
Hara Sponsor
02 30

0
Hara
haraasdasdasdhara
0722DejaImiSunaCunoscut
true
Doin a Hfadfasdara


0
Hara
hadagahgahaharamail.hara
0722DejaImiSunaCunoscut
true
Doin a Hfad3123ara
```

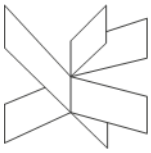
IX. Results and Discussion

The team feels that the results provided by the current analysis based on VIA's requirements and the overall design and implementation are satisfactory not only towards themselves but the stakeholder as well.

The system has been developed and delivered in time as well as the website itself.

X. Conclusion

All functional and non-functional requirements have been met and as it stands, the system is working flawlessly and achieves everything that it needs to do. Everything has bene implemented cleanly to assure maximum maintainability throughout the usage of the system.



As for the website, based on VIA's initial sketches, the team has managed to create a simple, clean and responsive website that would suit any company such as the stakeholder.

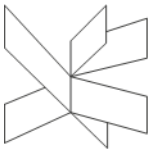
Overall the project has yielded powerful results for both employer and developer. The employer received a stable and robust system accompanied by a clean website while the developer received tremendous experience In how to tackle projects like these.

XI. Project Future

The group is overall satisfied with the project implementation. From a technical viewpoint everything is done correctly and all the requirements have been met, however, there are a few more improvements the team could make regarding the current version of the application. The user input validation could be improved to prevent incorrect data input and offer more stability to the system. Also the developers could create a utility controller class, however, it is not an essential function for the system to have beacause of it's simplicity.

Apart from that, there are a lot more improvements that could be made in order to make the project ready for some kind of deployment stage. The first step would be connecting the application with the website. The group could add a login system to the website, so that users can create accounts by themselves and it would be automatically created on the server without any administrator's manual input. Furthermore the team could supplement the website with more functionality for users such as ticket reservation, event marking, and a private message system.

As for the application itself, it could benefit from a history list, so the administrator cansee all of the changes that were previously made. Moreover, an essential feature would be the administrator's ability to restore recently deleted files, to achieve this the group would use the simplest functions of hiding data for specific periods of time before deleting it completely, after the administrator actually calls the delete method.



XII. List of Appendices

Note: [Clicky, clicky…]