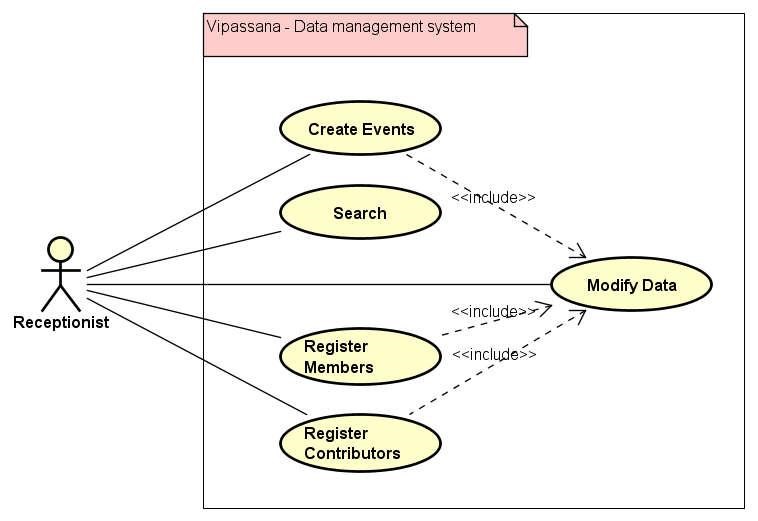
Requirements

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 | 1. The user should be able to create events (Workshops, seminars, lectures, journeys) 2. The user should be able to register member my name, email address, phone number and payment status. 3. The user should be able to register employees by name, field of work, email address and phone number. 4. The user should be able to search for events, members and contributors. 5. The user should be able to modify the data stored for events, members and contributors. |
| Non-priority Requirements | 1. The system should facilitate searches with a filter. |
| **Non-functional Requirements** | |
| 1. The system must be programmed in Java. 2. The system must be destined for a single user. | |

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.



Use Case Descriptions

Below is a list of different use case descriptions, which in depth explains how each 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, almost like some guidelines or user manual.

|  |  |
| --- | --- |
| Use Case | Search |
| Summary | Search for a specific piece of data stored within the program. |
| Actor | The receptionist/the employee |
| Pre-condition | None |
| Post-condition | A filter for type of data has been chosen |
| Base sequence | 1. The user chooses the type of data to search for. 2. The system opens the file that contains the specific data. 3. The system displays the data in a list format. 4. The user filters the list of displayed data based on specific attributes. 5. The system displays the filtered results. |
| Exception sequence | If the user chooses to search for Sponsors:  1-2   * 1. The system displays Sponsors in a list format, without filters. |

|  |  |
| --- | --- |
| 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 | 1. User chooses the type of event to create (lecture, journey, workshop, seminar). 2. User titles the new event. 3. User enters event subject/theme. 4. User chooses date and time for the event. 5. User enters event duration. 6. User chooses a contributor, from a list, to be assigned for this event. 7. User sets a member limit for attendance. 8. User enters the price of the event. 9. User enters location for event. 10. User clicks “save” button. 11. System validates data. 12. System creates new event in memory. |
| Exception sequence | If the user chooses Journey event type:  1-3   1. User enters date, time and ending date for event. 2. No contributor will be assigned.   6-12  If the user chooses Workshop event type:  1-3   1. User enters date, time and ending date for event.   5-12  If data is invalid:  1-12 |

|  |  |
| --- | --- |
| Use Case | Modify Data |
| Summary | Modifies currently existing data |
| Actor | The receptionist/the employee |
| Pre-condition | There must be available data in order to modify it. |
| Post-condition | The changes made to the data are stored. |
| Base sequence | 1. User chooses the type of data to modify (events, contributors, members). 2. The system deletes the chosen instance. 3. The system opens a window for the specific chosen data. 4. User changes the properties. 5. User clicks “save” button. 6. System saves new instance. |
| Exception sequence |  |

|  |  |
| --- | --- |
| Use Case | Register Contributor |
| Summary | The receptionist/employee adds a new contributor to the system |
| Actor | The receptionist/the employee |
| Pre-condition | None |
| Post-condition | A new contributor is added to the system |
| Base sequence | 1. User chooses type of contributor (lecturer, sponsor). 2. User enters contributor’s name. 3. User enters contributor’s phone number, email address. 4. Uses clicks “save” button. 5. System stores data to its specific file. |
| Exception sequence | If user chooses Lecturer data type:  1   1. User enters contributor’s name and specialization.   3-5  If user chooses Sponsor data type:  1  2. User enters contributor’s name and money contributed.  3-5 |

|  |  |
| --- | --- |
| Use Case | Register a member |
| Summary | The receptionist/employee add a new member to the system |
| Actor | The receptionist/the employee |
| Pre-condition | None |
| Post-condition | A new member is added to the system |
| Base sequence | 1. User enters member’s name. 2. User enters member’s phone number and email address and date of membership. 3. User clicks “save” button. 4. System validates data. 5. System saves member to its specific data file. |
| Exception sequence | If data is invalid: 1-3  If member is of type paid member, user should also enter payment year. |