Software Requirements Specification

Version 1.0

October 28, 2014

Nirmaan Activity Manager

|  |  |
| --- | --- |
| Devashish Deshpande | 2013A7PS122G |
| Hemanshu Sethi | 2012B4A7420G |
| Manav Mishra | 2013A7PS123G |
| Mohit Menon | 2013A7PS110G |
| Mrunmayee Nasery | 2013A7PS087G |
| Sumit Bhatia | 2013A7PS031G |
| Krishnan G | 2012A4A7207G |

Submitted in partial fulfillment

Of the requirements of

CS F213 : Object Oriented Programming

Table of Contents

[List of Figures iii](#_Toc402305716)

[1.0. Introduction 4](#_Toc402305717)

[1.1. Purpose 4](#_Toc402305718)

[1.2. Scope of Project 4](#_Toc402305719)

[1.3. Glossary 4](#_Toc402305720)

[1.4. References 5](#_Toc402305721)

[1.5. Overview of Document 5](#_Toc402305722)

[2.0. Overall Description 6](#_Toc402305723)

[2.1 System Environment 6](#_Toc402305724)

[2.2 Functional Requirements Specification 7](#_Toc402305725)

[2.2.1 Event Head Use Case 7](#_Toc402305726)

[2.2.2 Volunteer Use Case 9](#_Toc402305727)

[2.3 User Characteristics 10](#_Toc402305728)

[2.4 Non-Functional Requirements 10](#_Toc402305729)

[3.0. Requirements Specification 12](#_Toc402305730)

[3.1 External Interface Requirements 12](#_Toc402305731)

[3.2 Functional Requirements 12](#_Toc402305732)

[3.2.1 Create event 12](#_Toc402305733)

[3.2.2 Create activity 12](#_Toc402305734)

[3.2.3 Post event update 13](#_Toc402305735)

[3.2.4 Schedule meeting 13](#_Toc402305736)

[3.2.5 Post discussion 14](#_Toc402305737)

[3.2.6 Confirm availability 14](#_Toc402305738)

[3.3 Detailed Non-Functional Requirements 16](#_Toc402305739)

[3.3.1 Logical Structure of the Data 16](#_Toc402305740)

[3.3.2 Security 19](#_Toc402305741)

# List of Figures

[Figure 1 - System Environment 4](#__RefHeading___Toc77487669)

[Figure 2 - Logical Structure of the Activity manager Data](#__RefHeading___Toc77487672) 6

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to present a detailed description of the Nirmaan activity manager application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

## 1.2. Scope of Project

This software system is intended to simplify the task of coordinating activities of the campus chapter of Nirmaan organization. This system will be designed to reduce the work involved in coordinating activities by providing a platform to plan activities and communicate the details of the same to all volunteers while remaining simple enough to be used by all members of the organization.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Activity | A single activity carried out as part of an *Event* |
| Database | Collection of all the information monitored by this system. |
| Event | A long term event in which the organization participates. An event may consist of multiple *activities* |
| Event Head | The volunteer responsible for coordinating activities related to that event. |
| Meeting | A meeting between certain members to discuss about events |
| Notice | A message that needs to be brought to the attention of members of the organization through the system |
| TimeTable | The college timetable of the volunteer |
| User | The person using the system |
| Vertical | A team in the organization that works on projects related to a certain field. |
| Volunteer | A non-administrative member of the organization. |

## 1.4. References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

## 1.5. Overview of Document

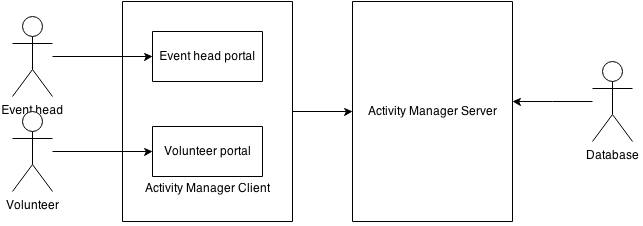
The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

# 2.0. Overall Description

## 2.1 System Environment

Figure - System Environment

The Activity manager has 2 active actors. The manager has a client application and a server as cooperating systems.

The Project head and volunteers access the system through a client application. The event heads will have options that help coordinate the events he/she is responsible for as well as appoint new event heads. To use the system as either volunteer or project head requires proper authentication. All clients communicate to the server via a network.

## 2.2 Functional Requirements Specification

This section outlines the use cases for the Event heads and Volunteers separately.

### 2.2.1 Event Head Use Case

#### Use case: Create event

**Diagram:**

Event head

Create event

**Brief Description**

An Event head creates an event and sets the vertical and head that are responsible for it.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Event head must be logged in to the system.

1. The event head creates an event and enters a name and description.
2. The system prompts the user to select the concerned vertical and appoint the event head.
3. The user fills in the choices and publishes the event.
4. The system notifies the event head and grants him/her administrative privileges over the new event.

**Xref:** Section 3.2.1, Create Event

#### Use case: Create Activity

**Diagram:**

Event head

Create activity

**Brief Description**

An Event head creates an activity as part of an event and assigns volunteers who are to be involved in it.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Event head must be logged in to the system and the parent event must have been created.

1. The user selects the parent event and chooses to create an activity.
2. The user sets the time and location of the event.
3. The user selects volunteers to take part in the event.
   1. The system checks whether the volunteer is free at the scheduled time and adds.
4. The user confirms the details
5. Notices are sent to all the volunteers who are involved in the activity enquiring about their availability for the activity.

**Xref:** Section 3.2.2, Create activity

#### Use case: Post event update

**Diagram:**

Event head

Post event update

**Brief Description**

An Event head posts a message about the progress about of the event.

**Initial Step-By-Step Description**

Before this use case can be initiated, the user must be logged in to the system and navigate to the concerned event page.

1. The user chooses to post an update.
2. He then fills in the form with the required updates and submits it.
3. The system stores the update and updates the concerned event.

**Xref:** Section 3.2.3, Post event update

#### Use case: Schedule meeting

**Diagram:**

Event Head

Schedule meeting

**Brief Description**

An Event head posts a message about the progress about of the event.

**Initial Step-By-Step Description**

Before this use case can be initiated, the user must be logged in to the system and navigate to the concerned event page.

1. The user chooses to post an update.
2. He then fills in the form with the required updates and submits it.
3. The system stores the update and updates the concerned event.

**Xref:** Section 3.2.4, Schedule meeting

### 2.2.2 Volunteer Use Case

#### Use case: Post discussion

**Diagram:**

Volunteer

Post discussion

**Brief Description**

A volunteer either creates a new discussion thread or posts to an existing thread.

**Initial Step-By-Step Description**

Before this use case can be initiated, the User must be logged in.

1. The user navigates to the relevant discussion page.
2. He enters his message in the form and submits it.
3. The system adds the message to the thread and updates the thread.

**Xref:** Section 3.2.5, Post discussion

#### Use case: Confirm availability

Volunteer

Confirm availability

**Diagram:**

**Brief Description**

The user confirms his availability for an activity he has been assigned to.

**Initial Step-By-Step Description**

Before this use case can be initiated, the User must be logged in and have navigated to the page of the concerned activity.

1. The system prompts the user to confirm his availability for the activity.
2. The user enters whether or not he will be available.
3. The availability status is updated.
4. If the volunteer is unavailable, a notification is sent to the event head.

**Xref:** Section 3.2.6, Confirm availability

## 2.3 User Characteristics

The users are expected to be computer literate and be able to use applications.

## 2.4 Non-Functional Requirements

The activity manager will be deployed as 2 parts. All information will be stored on a database on a server with internet capabilities. Every user will have a client installed on his PC to interact with the server. The client’s pc must have a recent version of Java installed. The software developed here assumes the use of a tool such as Tomcat for connection between the Web scripts and the database. The speed of the user’s connection will depend on the hardware used rather than characteristics of this system.

# 3.0. Requirements Specification

## 3.1 External Interface Requirements

No link to any external system is required.

## 3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

### 3.2.1 Create event

|  |  |
| --- | --- |
| **Use Case Name** | Create event |
| **XRef** | Section 2.2.1 |
| **Trigger** | The user opts to create a new event |
| **Precondition** | A form containing all the required fields is displayed. |
| **Basic Path** | 1. The user enters a name and description of the event. 2. He chooses which vertical the event falls under. 3. He assigns a head for the event. 4. He submits the form. 5. The system creates an event 6. The user is taken to the event details page. |
| **Postcondition** | The event is created on the server. The user is taken to the event details page. |
| **Exception Paths** | The user may abandon the creation at any time. |
| **Other** | To choose the head of the event, The user is presented with a list of all volunteers |

### 3.2.2 Create activity

|  |  |
| --- | --- |
| **Use Case Name** | Create activity |
| **XRef** | Section 2.2.1, Create activity |
| **Trigger** | The user chooses to create an event from the event details page |
| **Precondition** | A form containing all the required fields is displayed. |
| **Basic Path** | 1. The user enters a name and description of the activity. 2. He chooses which event the activity falls under. 3. He sets the time and location for the event. 4. He assigns volunteers to participate in the activity. 5. He confirms the data entered and submits the form. 6. The system creates an activity 7. The user is taken to the activity details page. |
| **Postcondition** | The activity is created and the user is taken to the activity details page. |
| **Exception Paths** | A volunteer may not be available at a given time. The user will not be able to add him to the activity.  The user may abandon creation at any time. |
| **Other** | To choose the volunteers, The user is provided with a list of all the volunteers. |

### 3.2.3 Post event update

|  |  |
| --- | --- |
| **Use Case Name** | Post event update |
| **XRef** | Section 2.2.1, Post event update |
| **Trigger** | The user navigates to the event details page and chooses to post an update about the progress of the event |
| **Precondition** | A form containing all the required fields is displayed. |
| **Basic Path** | 1. The user enters a message stating the progress made. 2. He submits the form. 3. The system adds the update to the event. |
| **Postcondition** | The update is added to the activity. |

### 3.2.4 Schedule meeting

|  |  |
| --- | --- |
| **Use Case Name** | Schedule meeting |
| **XRef** | Section 2.2.1, Schedule meeting |
| **Trigger** | The user chooses to schedule a meeting. |
| **Precondition** | A form containing all the required fields is displayed. |
| **Basic Path** | 1. The user enters a name and purpose of the meeting. 2. He chooses which volunteers / verticals are expected to attend. 3. He sets the time and location for the event. 4. He confirms the data entered and submits the form. 5. The system creates a meeting. 6. Notices are sent to all users expected to attend. |
| **Postcondition** | The meeting is scheduled and notices are sent to all members involved. |
| **Other** | To choose the volunteers / vertical, The user is provided with a list of all the volunteers and verticals. |

### 3.2.5 Post discussion

|  |  |
| --- | --- |
| **Use Case Name** | Post discussion |
| **XRef** | Section 2.2.2, Post discussion |
| **Trigger** | The user posts in a discussion thread |
| **Precondition** | The user navigates to the discussion page. A form to add a response is displayed. |
| **Basic Path** | 1. The user enters the message. 2. He submits the message. 3. The system updates the database accordingly. |
| **Postcondition** | The discussion thread is updated on the server and the new post is visible in the thread. |
| **Alternate paths** | The user may want to create a new discussion thread.  Precondition: The user navigates to the concerned event page. He chooses to start a new discussion thread using the provided button.  At step 3: The system creates a new thread. The post is added to this thread. |
| **Exception Paths** | The user may abandon the post at any time. |

### 3.2.6 Confirm availability

|  |  |
| --- | --- |
| **Use Case Name** | Confirm availability |
| **XRef** | Section 2.2.2, Confirm availability |
| **Trigger** | The volunteer clicks on a notice which prompts him to confirm his availability for an event he was assigned to.  The volunteer views an activity he has been assigned to and clicks the prompt. |
| **Precondition** | A prompt asking him to confirm availability is displayed. |
| **Basic Path** | 1. The user confirms whether he is available or not. 2. The system updates the database accordingly. 3. If the user is unavailable, The event head is notified. |
| **Postcondition** | The availability of the user is updated on the server. |

## 3.3 Detailed Non-Functional Requirements

### 3.3.1 Logical Structure of the Data

The logical structure of the data to be stored in the Activity Manager is given below.

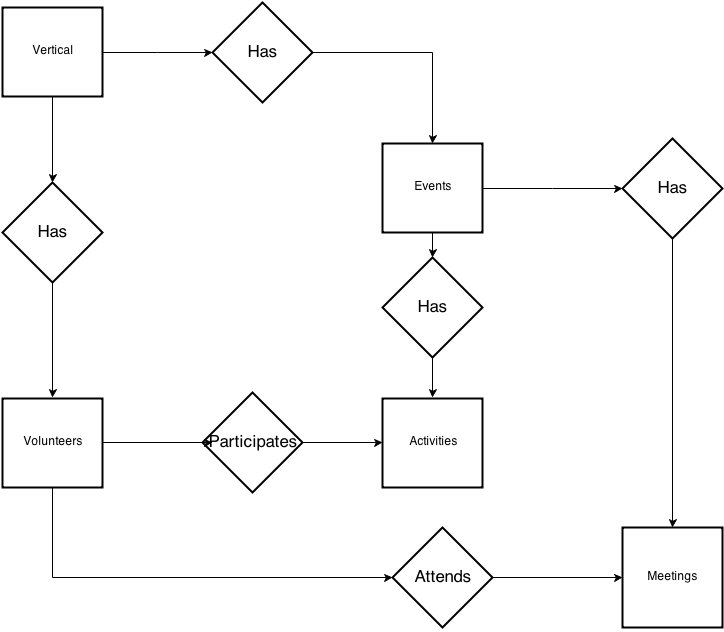


Figure 2 - Logical Structure of the Activity manager Data

The data descriptions of each of these data entities are as follows:

**Vertical Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of the vertical |  |
| Head | Pointer | Volunteer entity |  |

**Volunteer Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name |  |
| Vertical | Pointer | Vertical entity |  |
| Year | Number | Year of study |  |
| Email | Text | Email |  |
| Contact no | Text | Contact number |  |
| Timetable | Pointer | Timetable Entity |  |

**Event Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of the event |  |
| Head | Pointer | Volunteer entity |  |
| Vertical | Pointer | Vertical entity |  |

**Activity Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Descriptive name for the activity |  |
| Head | Pointer | Volunteer entity |  |
| Time | Number | Timestamp of the meeting |  |
| Location | Text | Location of the meeting |  |
| Participants | Pointer | Volunteer entity | Multiplicity |
| Report | Text | Report / Summary of the activity |  |

**Meeting Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Descriptive name for the meeting |  |
| Purpose | Text | Purpose of the meeting |  |
| Attendees | Pointer | Volunteer entity | Multiplicity allowed |

### 3.3.2 Security

The central server will have appropriate security to prevent access from unauthorized sources. The client will communicate to the server and access the database through an API. This will protect the system and the database from misuse.

Every user account will require a password for authentication.