**StarContests.com**

***If it’s not here it’s not happening.***

***Software Design Document v1.0***

# **Team 8**

**Instructor - Prof. Asim Banerjee**

## GROUP MEMBERS

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Name** | **ID** |
| 1. | HARDIK BELADIYA | 201201064 |
| 2. | ARCHIT GAJJAR | 201201066 |
| 3. | SOHAM DARJI | 201201070 |
| 4. | KRUPAL BAROT | 201201074 |
| 5. | DHAVAL CHAUDHARY | 201201075 |
| 6. | PRACHI KOTHARI | 201201077 |
| 7. | YASH KUMAR JAIN | 201201080 |
| 8. | RACHIT MISHRA | 201201092 |
| 9. | SHIVANI THAKKER | 201201108 |

## REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Document | Date | Version | Created By | Reviewed By |
| Software Design Document | 4 March, 2015 | 1.0 | All members | Prachi, Rachit, Shivani |

## DOCUMENT DESCRIPTION

The purpose of this document is to formalize the design and implementation of

StarContests.com website. The primary purpose of the document is to Provide the High and

Low Level Design of the website. To document different modules and components to be implemented. To document the algorithm used in building each module or group of modules and show how the various components are integrated. This would be serve as an Entry document for Coding Phase.

## CONTENTS

1. Introduction.............................................................................................3
   1. Purpose and Scope....................................................................3
   2. Design Overview........................................................................4
      1. Approach....................................................................4
      2. Guiding Principles.......................................................4
      3. Developing environment..............................................5
      4. Programming paradigm................................................5
      5. Reference Documents..................................................5
2. High Level Design......................................................................................5
   1. Architectural Design Diagram....................................................5
   2. Database Design....................................................................10
      1. ERD..............................................................................11
      2. Relational Model.........................................................11
      3. Data dictionary............................................................11
      4. Sequence Diagrams....................................................17
      5. Use case diagram.......................................................26

2.2.5 Data flow diagram.......................................................26

1. Low Level Design......................................................................................29
   1. Activity Diagram........................................................................29

## INTRODUCTION

The software design document (SDD) is a written description of the software product, that a software designer writes in order to give a software development team an overall guidance of the architecture of the software project.

1. Purpose

The purpose of this document is to outline the technical design of a contest hosting platform ‘StarContests.com’ for encouraging local talents and provide an overview for its usage and management by the users. Its main purpose is to – Provide high level and low level design of our system.

Document the functionality provided by each module or group of modules and show how the various components interact in the design. This document is intended to help the coding team to build our system.

1. Scope

This document contains the low level design which shows the various modules and how they interact with each other thus enabling our system to work. The Design for various modules outlined in this document builds upon the scope defined in the Requirements phase. This document will serve as a link between design team and coding team and will be frequently referred by coding team to build our system.

1. Definitions, Acronyms and Abbreviations
   * PHP – Hypertext Processor scripting language
   * MySQL – Relational database management system (RDBMS)
2. Tools used in creating this document
   * draw.io
   * Microsoft Word

## DESIGN OVERVIEW

1. Approach

This document is created when the team moves from the problem domain to the solution domain, from designing to coding domain. This document describes the functions of various modules and how the various modules interact with each other. A design methodology is a systematic approach to creating a design by applying a particular set of techniques and following a particular set of guidelines. This document also shows ER Diagram, Data Dictionary, Data Model diagram, Data flow diagram, Sequence Diagram, Architectural Design flow, Use case diagram and the Activity Diagram.

2.Guiding Principles

Guiding principles provide a foundation upon which to develop the target architecture for our system. These in turn drive design principles that can be used to validate the design. Following are some of the guiding principles that will be followed.

* Scalable

Scalability is the ability of the platform to scale both up and down to support varying numbers of users. The application should be able to scale horizontally (by adding more servers) or vertically (by increasing hardware capacity or software efficiency).

* Flexible

Flexibility is the ability of the application to adapt and evolve to accommodate new requirements without affecting the existing operations. This relies on a modular architecture which isolates one layer from other. Coupling among various modules addressing different functionality must be as low as possible and cohesion among the group of modules to provide functionality must be as high as possible.

* Maintainable

Modules shall be designed with the view for future optimizations and for adding extra functionalities.

* Portable

Modules designed should be portable or platform independent.

* Security and Access Privileges

Modules shall make sure that the correct interface and functionalities are available according to the user type that is using it and the system is secure.

1. Developing environment

We are developing a website in HTML, PHP, CSS and JavaScript. We will use simple text editor Sublime-2 Text Editor to write the code. To run Apache servers and MySQL databases, we will use the software- XAMPP. We will also use Bootstrap js libraries.We have decided to follow procedural programming to implement.

1. Programming paradigm

We would be following procedural programming to implement the project as we are developing a web application which would require a systematic flow of all the features .We would be following different routines, subroutines which would execute all the features required in our website. ​**Procedural programming**​ is a programming paradigm, derived from structured programming is based upon the concept of the procedure call. Procedures, also known as routines, subroutines, methods, or functions, simply contain a series of computational steps to be carried out.

5.Reference Documents

* + Prof. Asim Banerjee, IT-314 Software Engineering, Lecture Slides, Winter 2015, DA-IICT.
  + SRS, team 8 SEN project
  + Requirements traceability matrix, team 8 SEN project

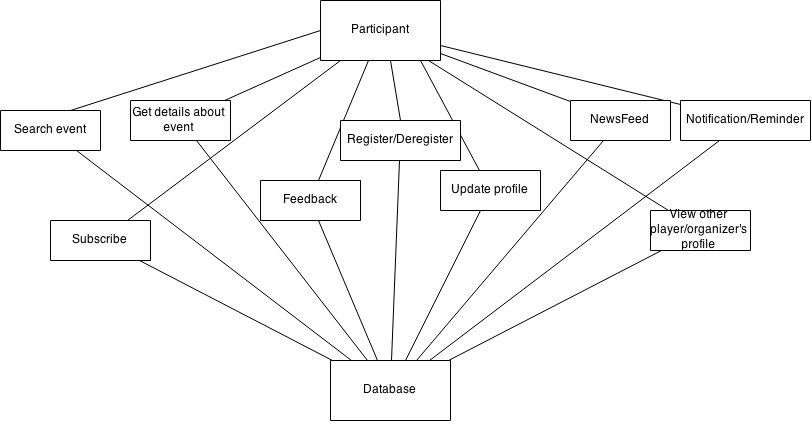
## HIGH LEVEL DESIGN

We are following ​**two tier architecture**​ model for our software development as our aim is to create a web application which the client will access on a browser and the server will have all the database and the processes. A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture.

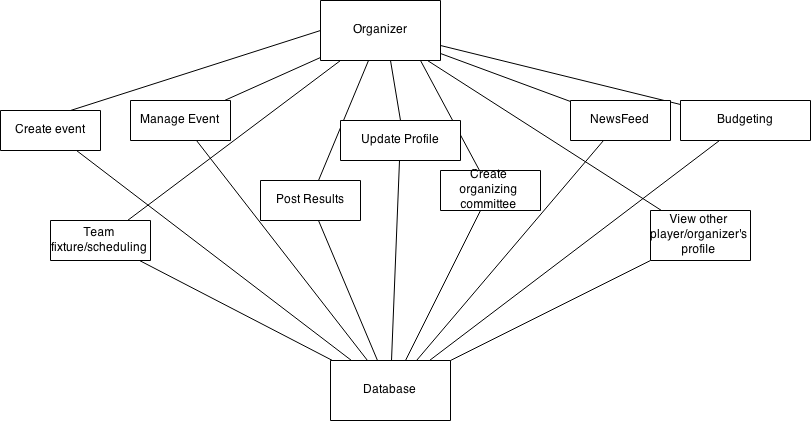
1. Architectural Design Diagram

The Architectural Design Diagrams are explained from the viewpoints of the types of users using the system namely, ​**Guests**​, ​**Participants**​, ​**Organizers, and Sub organizers.**

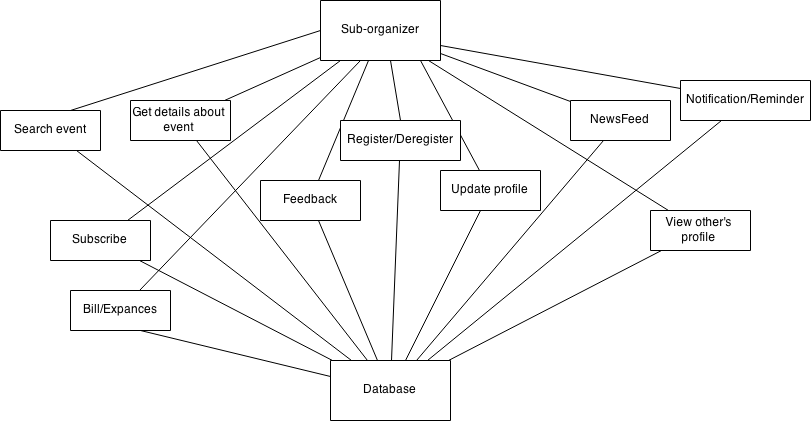
### ➢ Participant View​



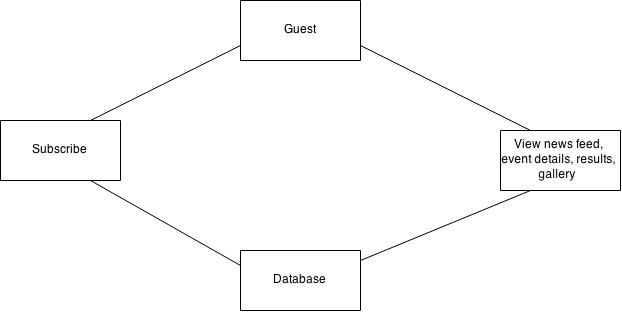
### ➢ Organizer View



### ➢ Sub organizer View​



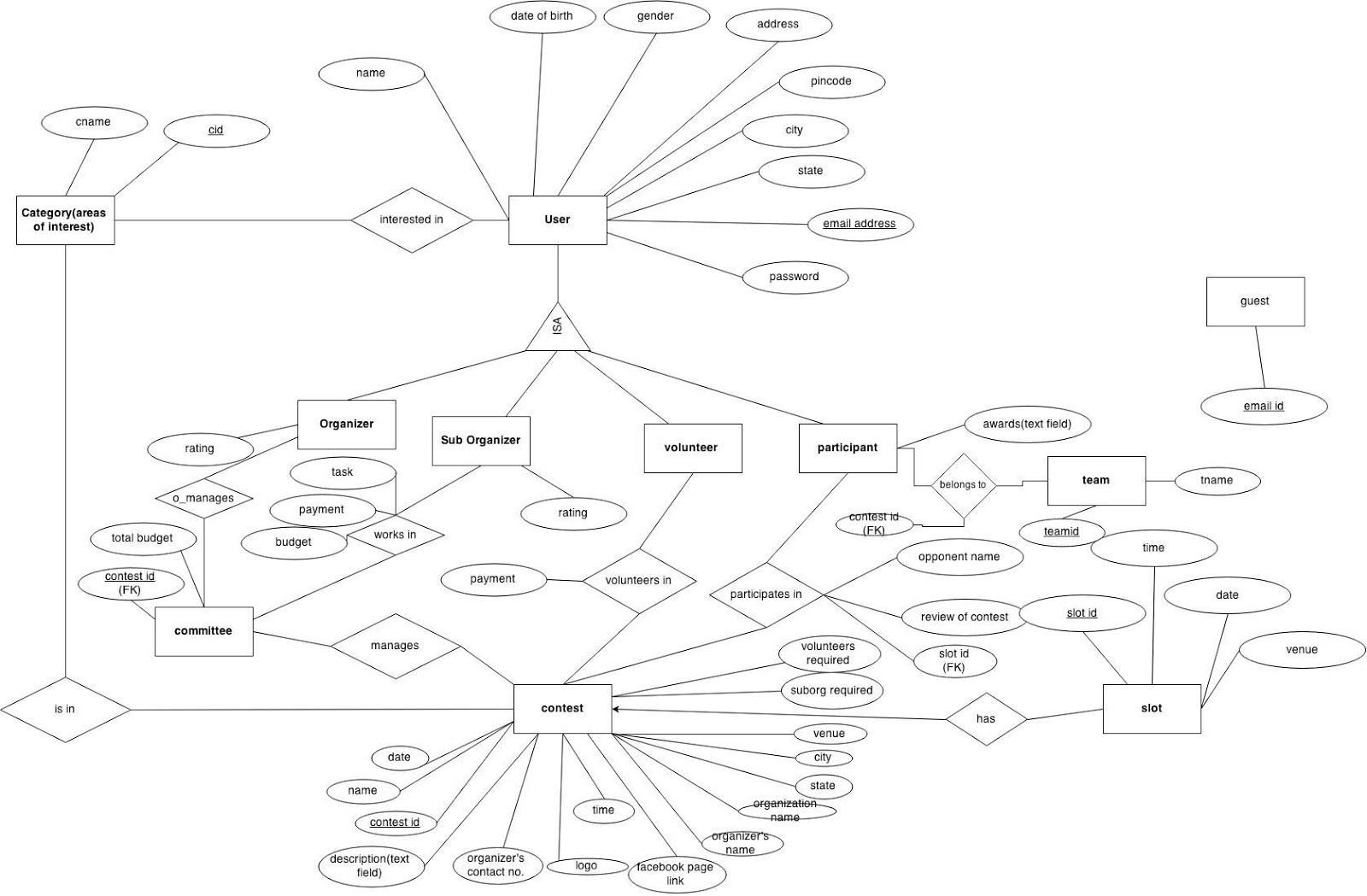
### ➢ Guest View



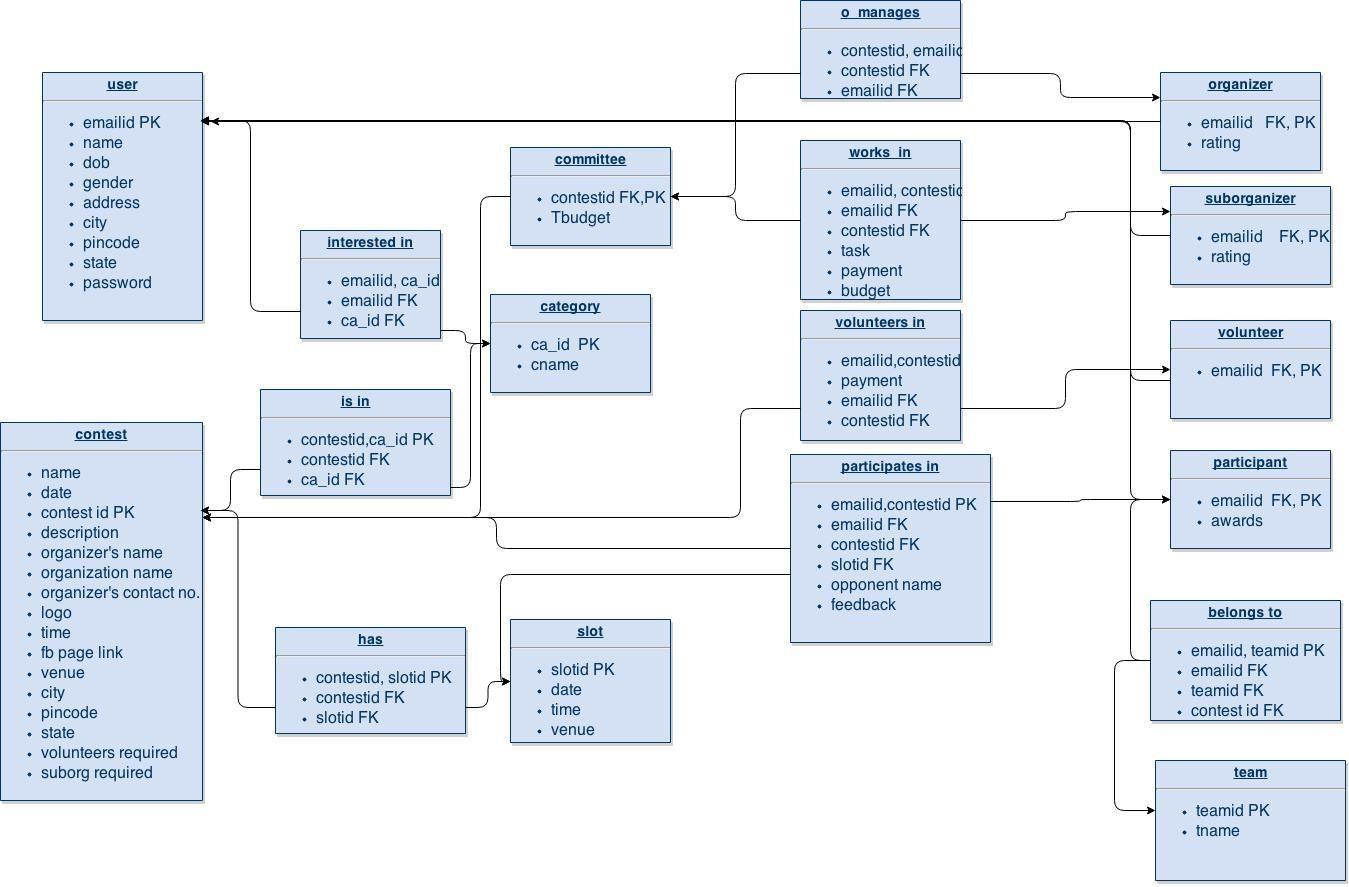
## DATABASE DESIGN

### 1. Entity relation diagram(ERD)​

Entity Relationship Diagram i.e. the ERD describes the database implementation in an abstract way, which is really for the end user or client to understand the working of the system without actually getting into technicalities of the databases.



### 2. Relational Model​



### 3. Data dictionary​

1. User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK | not null | string |  |
| name |  | not null | string |  |
| dob |  | not null | date |  |
| address |  | not null | string |  |
| gender |  | not null | char |  |
| city |  | not null | string |  |
| pincode |  | not null | int |  |
| state |  | not null | string |  |
| password |  | not null | string(hashed) |  |

1. Organizer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK,FK | not null | string | references ‘user’ |
| rating |  |  | int |  |

1. Sub-organizer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK,FK | not null | string | references ‘user’ |
| rating |  |  | int |  |

1. Participant

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK,FK | not null | string | references ‘user’ |
| awards |  |  | string |  |

1. Volunteer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK,FK | not null | string | references ‘user’ |

1. Guest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid | PK | not null | string(check for validation) |  |

1. Contest

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| cid | PK | not null | int |  |
| name |  | not null | string |  |
| date(from) |  | not null | date |  |
| date(to) |  | not null | date |  |
| description |  | not null | string |  |
| organizer’s name |  | not null | string |  |
| organization name |  | not null | string |  |
| organizer’s contact no. |  | not null | int |  |
| time |  | not null | time |  |
| logo |  | not null | jpg |  |
| venue(address) |  | not null | string |  |
| city |  | not null | string |  |
| state |  | not null | string |  |
| pincode |  | not null | int |  |
| volunteers required |  | not null | int |  |
| suborg required |  | not null | int |  |

1. Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| ca\_id | PK | not null | int |  |
| c\_name |  | not null | string |  |

9.Committee

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| contest id | PK, FK | not null | int | references ‘contest’ |
| total budget |  | not null | int |  |

1. Slot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| slotid | PK | not null | int |  |
| time |  | not null | time |  |
| date |  | not null | date |  |

1. participates\_in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid, contestid | PK | not null | string, int | emailid- participant,contestid- contest |
| slotid | FK | not null | int | references ‘slot’ |
| opponent name(if sports) |  |  | string |  |
| feedback |  |  | string |  |

1. volunteers\_in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid, contestid | PK | not null | string,int | emailid- volunteer, contestid- contest |
| payment |  |  | int |  |

1. contest\_has\_slots

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| contestid, slotid | PK | not null | int,int | contestid- contest, slotid-slot |

14.is\_in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| contestid, ca\_id | PK | not null | int,int | contest, category |

15.interested\_in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid, ca\_id | PK | not null | string,int | user, category |

16.o\_manages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid, contestid | PK | not null | string,int | organizer, committee |

1. works\_in

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid,contestid | PK | not null | string,int | suborg, committee |
| task |  |  | string |  |
| payment |  |  | int |  |
| budget |  |  | int |  |

1. team

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| team id | PK | not null | int |  |
| tname |  |  | string |  |

1. belongs to

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **key constraints** | **null constraints** | **domain constraints** | **referential constraints** |
| emailid,teamid | PK | not null | string,int | participants, team |
| contestid | FK | not null | int | references contest |

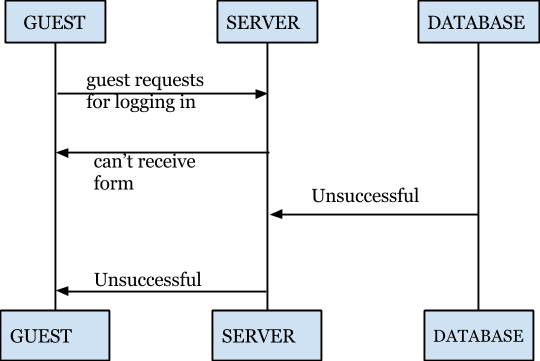
### 4. Sequence Diagrams​

These are interaction diagrams that show how processes operate with one another and in what order. It shows object interactions on a timeline.

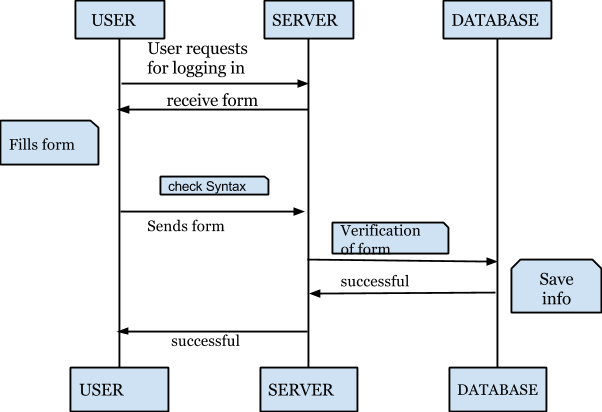
In software engineering, sequence diagrams are diagrams which show for particular use cases, a sequence of events which external actors(who perform roles) will generate in accordance with the project. In our scenario, there are various use cases as can be deduced from the traceability matrix and at the maximum, six entities who are entitled to either successfully or unsuccessfully perform that use case. After a thorough study and elimination of all the redundant use cases, we came up with 8 different sequence diagrams which follow in the next section.

**S.D 1:**

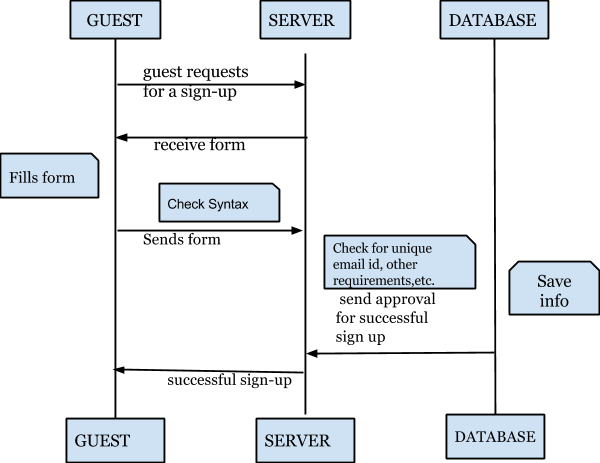
LOG-IN (For a guest) : ​A guest can’t log in to the website unless he/she has an account created. So this is how the sequence diagram will follow.



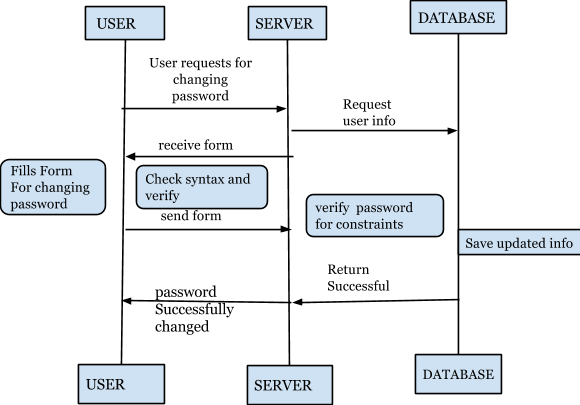
**S.D 2:** ​**LOG-IN (For other entities/users, namely a participant or an organizer or a sub-organizer or a volunteer or an admin)** ​**those who have an account already.**



**S.D 3:** ​**If a guest needs to sign up**​**, the following sequence diagram shall follow:**

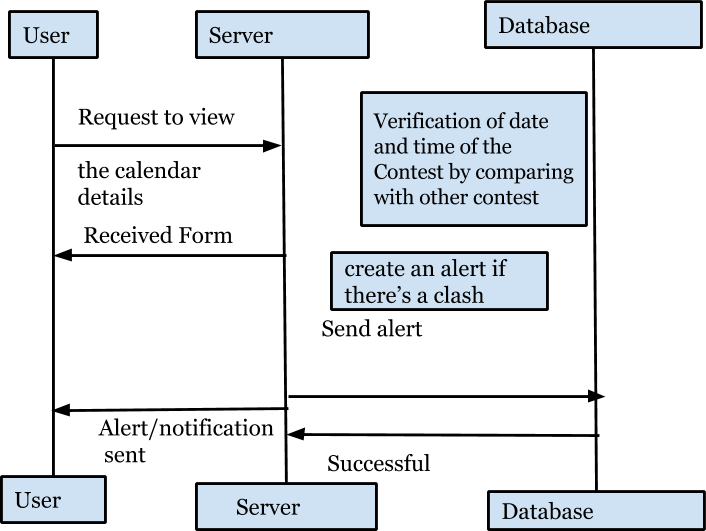


**S.D 4:** ​**If a user wishes to change password,**



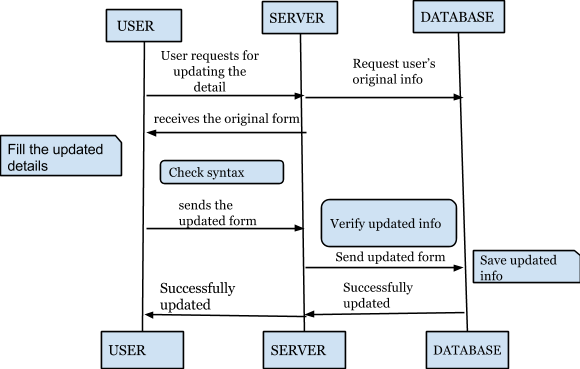
**NOTE : If a user wishes to view the budget, then that particular user entity must either be an organizer or a sub-organizer. Any other entity won’t even have this functionality to view the budget. So there’s no need to draw a sequence diagram for that exclusively.**

**S.D 5:**​ **If a user wishes to view the calendar of events**



**S.D 6: If a user wishes to update his/her profile/event section.**

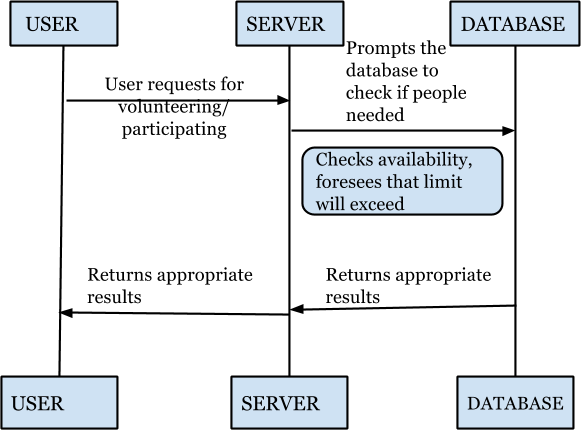
The detail which needs to be updated can be anything ranging from an updated profile to updating an event’s date or any other attribute of an entity.



**S.D 7: If a user wishes to volunteer/participate/sub-organize:** ​**The functionality which leads to this sequence diagram can be further divided into two cases:**

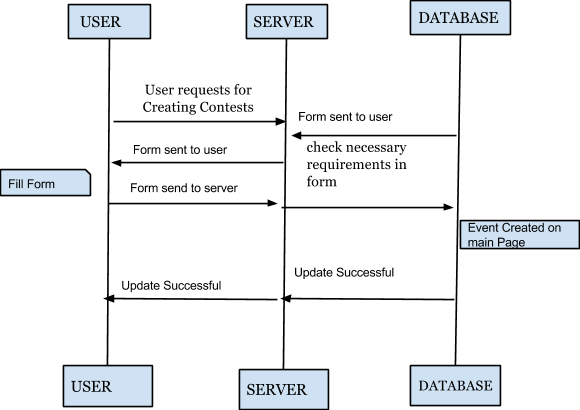
Case 1: If the organizer has already specified the number of participants in an

event or the no. of volunteers he/she must be needing in an event, then ​**no further requests** ​ ​**of participating or volunteering will be entertained** ​ by the organizer.

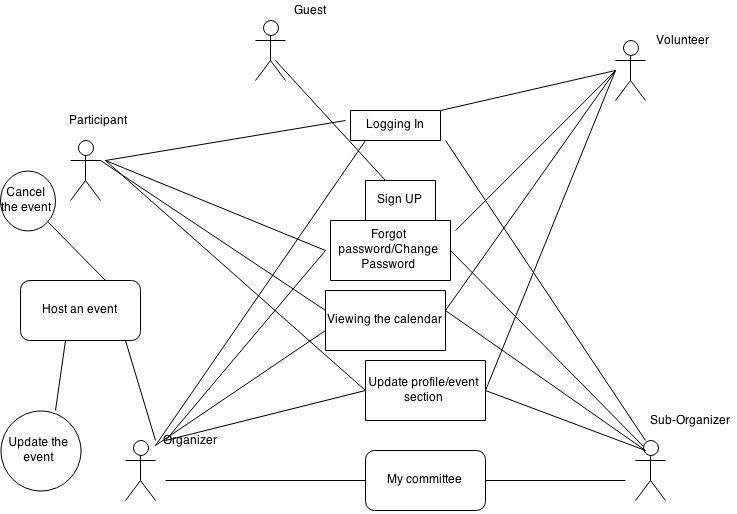


` Case 2: If there’s still some room for new participants or the volunteers, the server sends the request to the database and the database accepts the request after checking in in the slots. Accordingly, the result returned is ​**successful** ​from DB to the server and from server to the user.

**S.D 8: If a user(organizer) wishes to create an event following sequence diagram will be followed:**

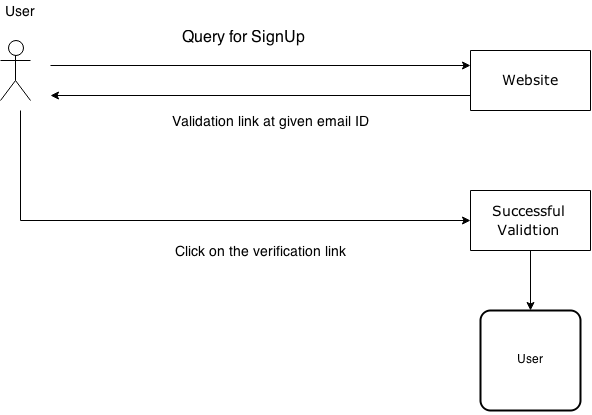


### 4. Use case diagram​

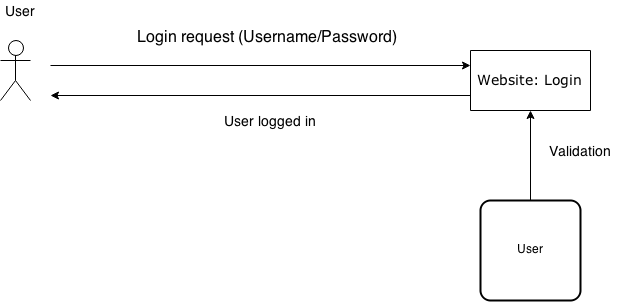


### 5. Data flow diagram​

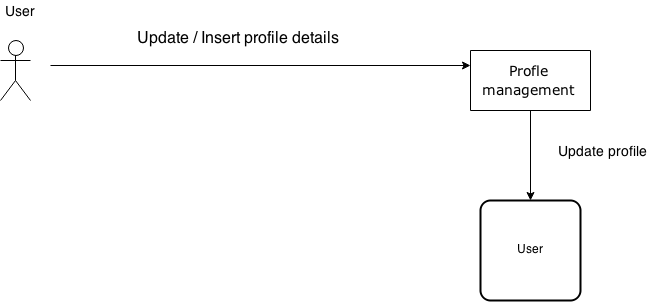
* Signup



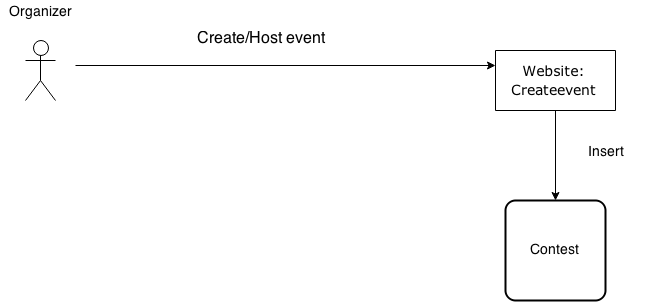
* Login



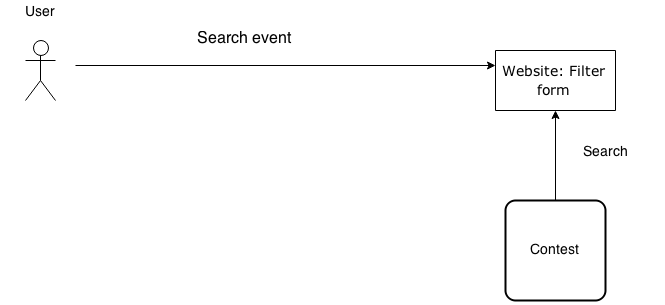
* Update Profile



* Host event



* Search event

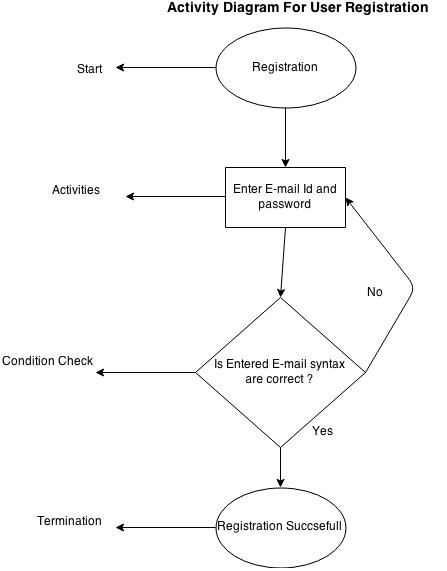


## LOW LEVEL DESIGN

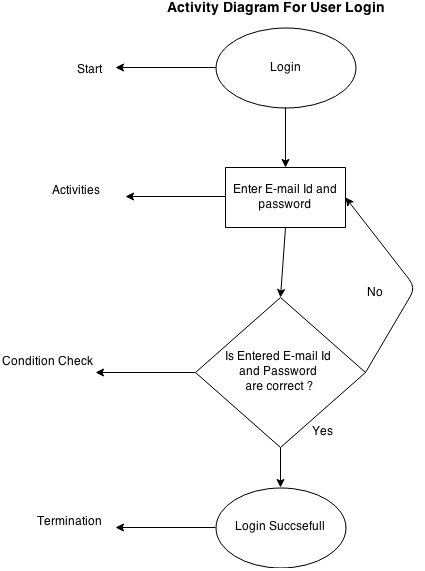
### 1. Activity Diagrams​

Activity diagrams are graphical representations of workflows of stepwise activities and actions. These are used to describe the operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

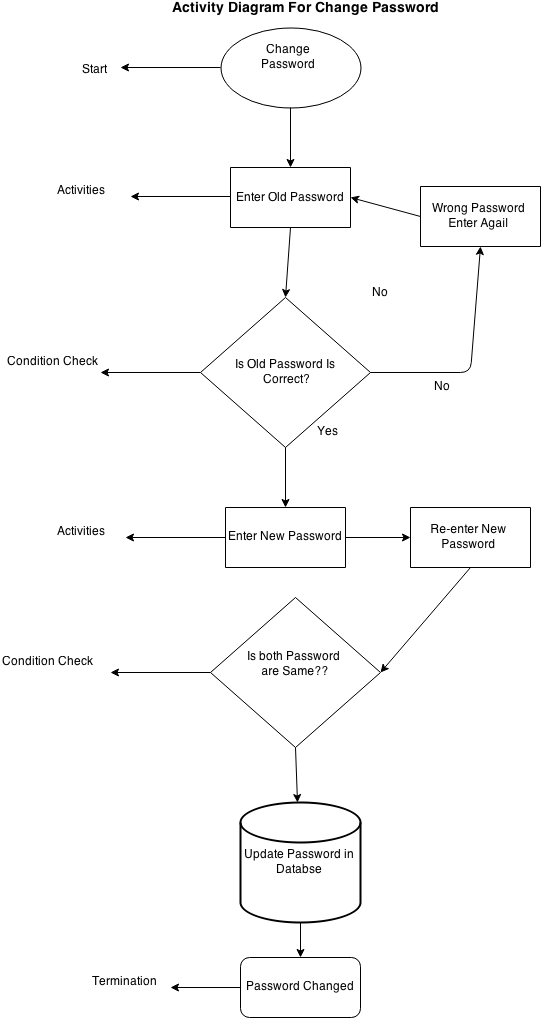
1.1 Register



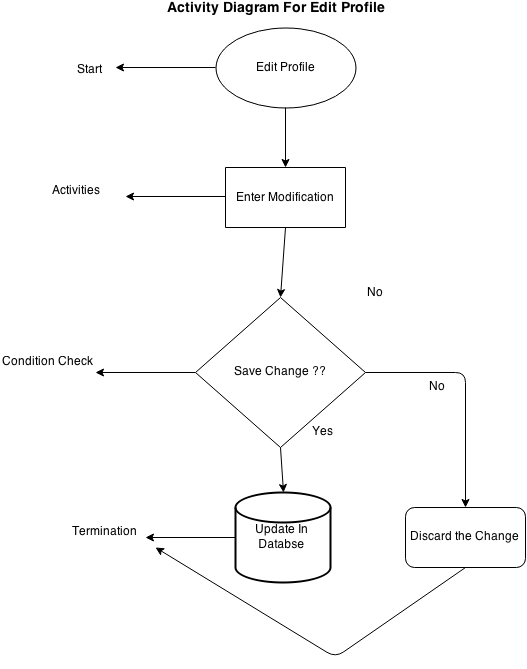
3.1.2 Login



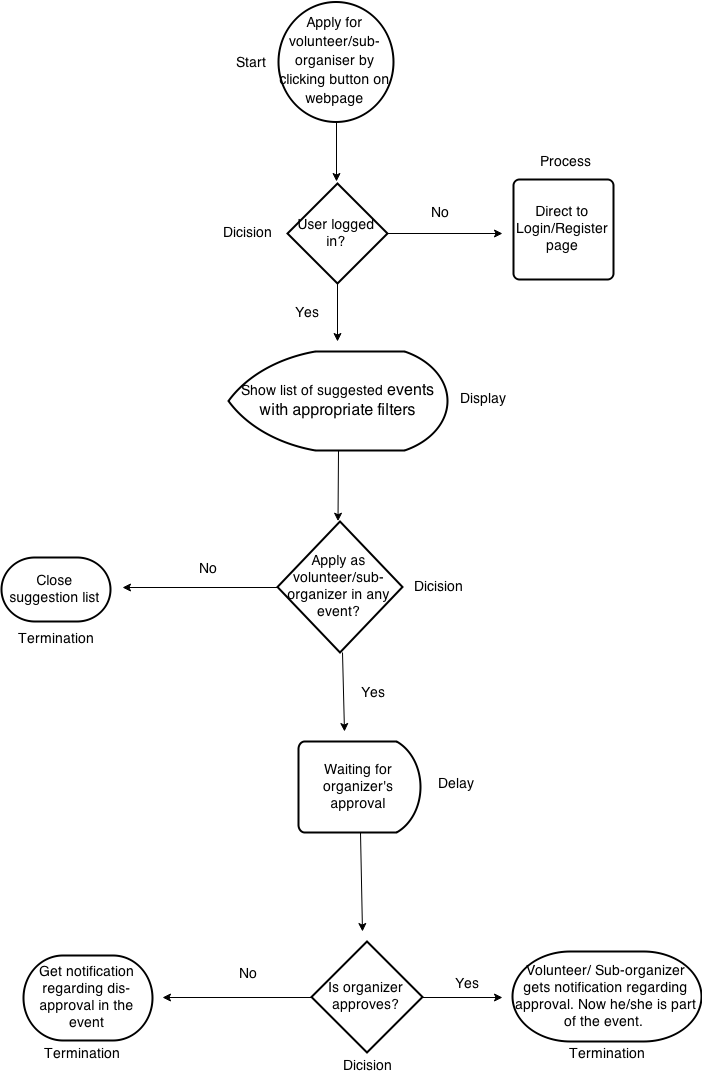
3.1. Change Password



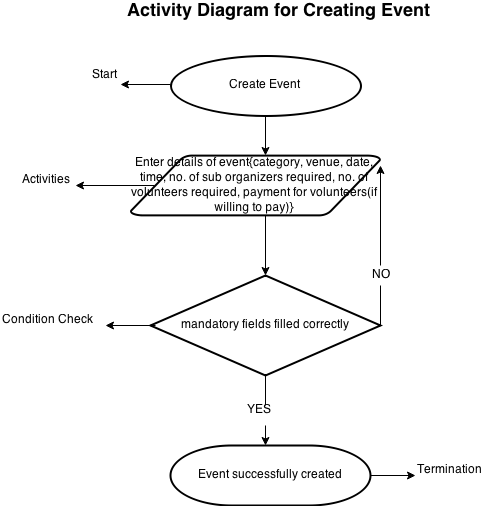
3.1.4 Edit Profile



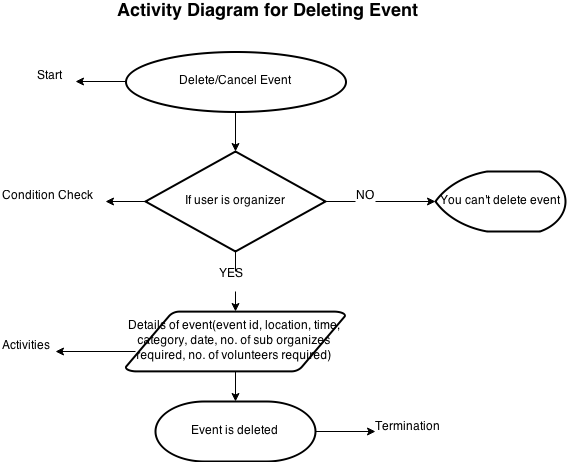
3.1.5 Be a Volunteer/suborganizer



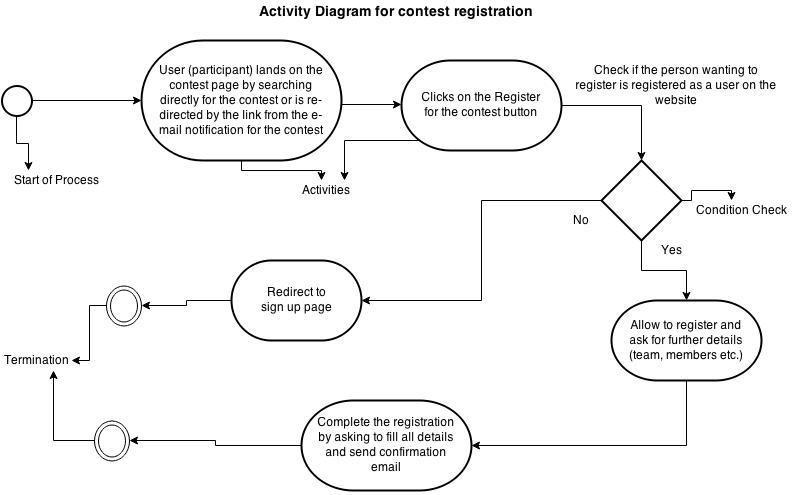
3.1.6 Create Event:



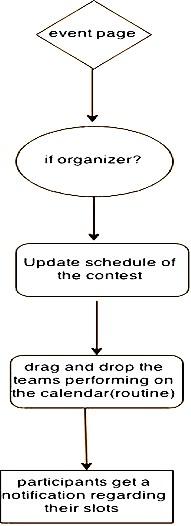
3.1.7 Delete/Cancel Event:



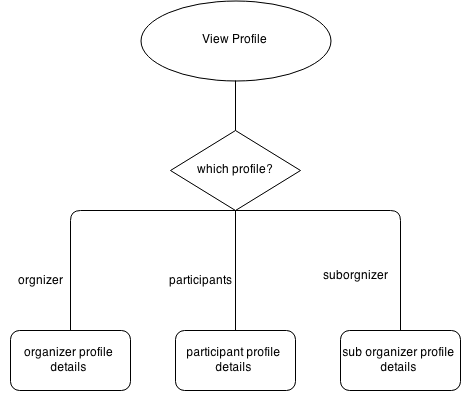
3.1.8 Register for a contest



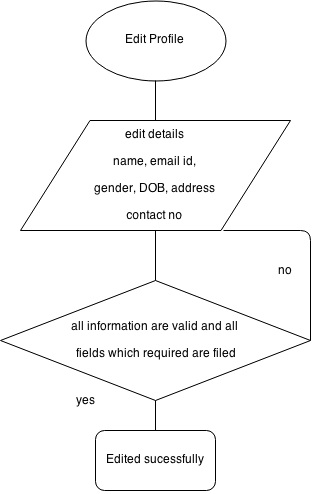
3.1.9 Update contest Schedule



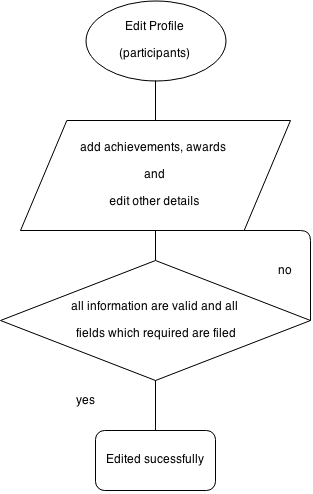
3.1.10 View Profile



3.1.12 Edit profile



3.1.13 Edit Profile (participants)



3.1.14 My committee

