DBIS Systems Requirements Specification

Generic Event Manager (GEM)

Submitted in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

by

Tejas Iyer	$< tejas@cse >$	98005004
Satyen Kale	$< satyen@cse >$	98005009
B. Aditya	$< baditya@cse >$	98005033
Vijay D'Silva	$< vjvictor@cse >$	98005102

under the guidance of

Prof. S. Sudarshan

Department of Computer Science and Engineering
Indian Institute of Technology
Bombay

November 14, 2000

Contents

List of Figures

Chapter 1

Introduction

1.1 Overall Description

Every year, IIT-Bombay and many other institutions conduct many events in the cultural and sporting spheres. These include Mood Indigo, the Swimming Gala, Inter-IIT and many more. Other colleges have their respective college festivals, technical weeks and others. At a smaller level, we have inter-hostel events.

One problem - that increases with the magnitude of the event, is the handling of data and maintaining of records - with respect to registration, scheduling, participant information and the like. Every event handles large amounts of such data and most of this is still done manually.

Our product is a generic event manager. It is designed to be a one-step solution for creating and managing the administration behind large events - especially cultural festivals, sporting events and the like. The software allows the sculpting of the database based on the requirements of the specific event and then provides all this as well as an easy interface to implement and access/modify registration details, schedules, results and the like.

1.2 Definitions

1. `FAA` : The filter and authorization level in our software.
2. `SysAd` : The System Administrator
3. `Event` : A large "event" - e.g. MoodIndigo, Techfest, Swimming Gala.
4. `Event Manager/Event Organizer` : Manager of an event.
5. `Sub-Event` : A component of an event - e.g. Conundrums in MI, 100m freestyle for men in the Swimming Gala.

6. `Sub-Event Manager/Sub-Event Organizer` : Manager of a sub-event.
7. `Participant` : A generic term used for both individuals and teams of people who wish to participate in the event.
8. `Contingent` : Like a college at an inter college festival or a swim club at a swimming gala - these contain members who then participate in the sub-events, either as individuals or teams.
9. `Team` : A group of individuals participating together in a sub-event. All team members have to belong to the same contingent.
10. `E1` $\leftarrow$ `E2` : Entity `E1` forms a `"IS A"` relationship with `E2`.
11. `(A1, A2)` $\Rightarrow$ `R` : Key `(A1, A2)` is a foreign key referencing relation `R`.

1.3 References

1. Abraham Silberschatz, Henry F. Korth and S. Sudarshan. *Database System Concepts.* McGrawHill, 1996

1.4 Overview of Developer's Responsibilities

The following responsibilities come under the developers purview:

1. *Development*: The software shall be developed entirely by us. Anyone who wishes to add further modules to the code to tailor it for their own needs is welcome to do so.
2. *Installation*: The software shall be designed to install on any system that follows the requirements given in Chapter 5.
3. *Training*: The software is easy to use and participant users should be able to use the software with ease. The same follows for higher level users, but these should be careful while making certain modifications. For eg. deleting attributes of an event could discard all the corresponding data from the database.
4. *Maintenance*: As this software shall aim to encompass most of what its original scope prescribed, we do not see extensibility as our responsibility. As stated earlier, users who wish to add modules are welcome to do so and should be able to do it with ease.

Chapter 2

General Description

2.1 Product Perspective

This product interfaces with the Oracle Database using JDBC and is otherwise stand alone. It has the following basic interfaces..

1. The Login/Register Interface - to provide new users with logins and also login authorized users.

2. The SysAd Interface - to alow the SysAd to create, view and modify events - add constraints and the like.

3. Event Manager Interface - For the event manager → to create sub-event schema, add sub-events, assign sub-event managers, view,modify details, set constraints and so on.

4. Sub-event Manager Interface - For the sub-event manager → to insert sub-event details, view/modify details, approve participation, insert judges etc.

5. Individual Interface - For events where individuals participate, this is the primary interface for participants - here she is allowed to view,edit her personal profile, register for events, view schedules and results.

6. Group Interface - For events where groups participate - here, a group representative can register teams (incl. single participants) for events, view schedules and so on.

7. In addition, the information page of the event should have links to view the schedule and provide access to the login/register pages.

2.2 Product Functions Overview

1. Login and Verification
2. Registration of new users
3. Event Creation/Population
4. Sub-event/Participant Schema Design
5. Sub-event Creation/Population
6. Schedule Generation
7. Result Generation
8. Registration for sub-events
9. Registration moderation and intimation of approval of application

2.3 User Characteristics

Our users are of three basic types.

1. `System Administrator` : The head of the database, only he can add events and other event constraints. He should be well-versed in using the system and also know about the database.

2. `Event Manager/Sub-event Manager` : Since these two classes of users are the basic controllers of all the data in the event database, they are to be trained in designing proper schema (over and above the default one provided), entering meaningful data, proper moderation, optimal use of the database (not allowing junk to fill the DB)

3. `Participants` : In both the case of the individual and the group participant, we assume the user is computer-literate, has a valid email id and also types in the right kind of data i.e. we do not do type checks on all data entered. Further, email ids are primary keys and are assumed to be entered correctly, even if one has to do so repeatedly.

All other users are not registered and only visit the website to glean information. They are inconsequential and we require that they only know how to use a browser!

2.4 General Constraints

This project is to be completed, including coding, testing and loading of the database, by Nov 15, 2000. A prototype demo is due on Oct 22, 2000.

Chapter 3

Information Description

3.1 Entities and Relationships

See figures 3.1, 3.2, 3.3 for ER-diagrams. (at the end of the chapter)

3.1.1 Entities

1. `User`
   Attributes:

   - `user_name`
   - `email_id`

   The entity `User` contains any user in the database. This is the smallest atom in our universe. All other entities which denote human beings in real life are built up from this point. As can be seen from the attributes, all users must have an e-mail address.

2. `Loggable` ← `User`
   Attributes:

   - `login_name`
   - `password`

   This entity contains all users who possess a login and password on the system. This distinction is made because there will arise cases where an individual user is not the smallest participating unit in the Event/Sub-Event and therefore will not have a distinct login.

3. `Participant` ← `User`
   Attribute:

- `PID` : the participant ID.

This entity contains participants in the Event/Sub-Event. They can be individuals or teams.

4. `Team_Member` ← `User`
   A `Team_Member` is a `User` who is a member of a team. This is a used in instances where the individual does not matter in the greater scheme of things!

5. `Judge` ← `User`
   Attributes:

   - `phone_number` Contact number.
   - `address_1`
   - `address_2` An alternate address if it exists.
   - `fax_no` Fax No if it exists.

   A judge for one or more Sub-Events. The attributes store information regarding ways of contacting the person.

6. `Registered User` ← `Loggable`
   Registered Users are those who possess a login and have registered for the Event. This is because users on the organising committee will also possess a login, but will not register for any event.

7. `Manager` ← `Loggable`
   Attributes: A `Manager` is a member of the organising committee.

8. `Contingent` ← `Registered User`
   Attributes:

   - `strength` Size of the contingent.
   - `CL_name` Contingent Leader's name.

   A `Contingent` which will register for the `Event`. Members of the contingent can then register for Sub-Events. The `name` attribute in the entity `User` will be the Contingent's name and the `email_id` attribute is the email address of the Contingent Leader(CL).

9. `Registered User` → `Individual` ← `Participant`
   The entity `Individual` is a specialisation of two different entities as can be seen above. The individual has to be registered for the event, but need not participate in any `SubEvent`.

10. `Team` ← `Participant`
    A team which will participate in a Sub-Event.

11. `Event_Manager` ← `Manager`
    An organiser of the overall event. This is derived from the entity `Manager`.

12. `SubEvent_Manager` ← `Manager`
    An organiser of a SubEvent. This is also derived from the entity Manager.

13. `Attribute`
    Attributes:

    - `attribute_name` take.

    Ironic though it may seem, the entity `Attribute` allows for attributes to be added to different entities and enhance the generic nature of this software.

14. `SubEvent_Attribute` ← `Attribute`
    Specialisation of the entity `Attribute` for Sub-Events.

15. `Participant_Attribute` ← `Attribute`
    Specialisation of the entity `Attribute` for participants.

16. `Event`
    Attributes:

    - `Event_name`
    - `Event_id`
    - `Location` The general location of the event.
    - `Capacity` Maximum capacity allowed in terms of participation.
    - `Private_flag` Indicates whether the Event information is public
    - `IC_flag` Indicates whether the event is contingent or team based. or private.

    The Event as a whole.

17. `Sub Event`
    Attributes:

    - `SubEvent_name`
    - `SubEvent_id`
    - `Venue` The venue within the location of the Event.
    - `Capacity` Constraint on the number of participants.
    - `Private_flag` Indicates whether the `SubEvent` information is public or private.
    - `Moderation_Flag` Indicates whether registeration is open or not.
    - `Team_Flag` Indicates whether the event is Team/Individual based.

A Sub Event. It will occur under some Event.

18. Date
    Attributes:
    - date

    A date within the duration of the Event on which some sub-event can occur.

3.1.2 Relationships

1. Contingent_Member
   Indicates which Contingent a Team participating in a sub-event belongs to.

2. Team_Member_Of
   Indicates that a Team Member is part of a Team which will participate in a sub-event.

3. Registered_For
   A relationship between a Registered_User and an Event which indicates that this Registered_User has registered for this Event.

4. Participates_In
   Attributes:
   • Result The result, if any.
   • Order The order in which the results are to be displayed.
   When an Participant (Individual or Team) participates in a Sub_Event.

5. EM_manages
   The relationship between an Event_Manager and the Event he manages.

6. SEM_manages
   The relationship between an SubEvent_Manager and the SubEvent he manages.

7. Judges
   Relationship between a Judge /Master of Ceremonies and the sub-event that he will be judging.

8. Part_Of
   For a Sub_event which is a part of an Event.

9. Occurs_On
   Attributes:
   • Start_Time
   • End_Time
   Relationship between the entities Date and Sub_Event which shows that a particular sub-event occurs on a given date, and also shows what period of time it is going to happen.

10. EM_Spec_SubEvent_Attrib
    A relationship between Event and SubEvent_Attrib which indicates that this SubEvent_Attrib is specified by the Event_Manager and is thus part of the information common to all SubEvents and is to be filled in by the SubEvent_Manager.

11. EM_Spec_Participant_Attrib
    A relationship between Event and Participant_Attrib which indicates that this Participant_Attrib is specified by the Event_Manager and is thus part of the information common to all Participants over all SubEvents and is to be filled in by the Participant.

12. SEM_Spec_SubEvent_Attrib
    Attribute:
    • Value The value for this SubEvent_Attrib
    A relationship between SubEvent and SubEvent_Attrib which indicates that this SubEvent_Attrib is specified by the SubEvent_Manager and is thus part of the information specific to this SubEvent, and is to be filled in by the SubEvent_Manager.

13. SEM_Spec_Participant_Attrib
    A relationship between Event and Participant_Attrib which indicates that this Participant_Attrib is specified by the SubEvent_Manager and is thus part of the information common to all Participants for this SubEvent and is to be filled in by the Participant.

14. EM_Spec_SubEvent_Attrib_Val
    Attribute:
    • Value The value for this EM_Spec_Participant_Attrib_Val
    Here, we have an aggregation of Event, EM_Spec_SubEvent_Attrib, and SubEvent_Attrib. The relationship is between this aggregation and SubEvent. This relationship gives the values for the EM_Spec_SubEvent_Attrib filled in by the SubEvent_Manager for this Event.

15. EM_Spec_Participant_Attrib_Val
    Attribute:
    • Value The value for this EM_Spec_Participant_Attrib_Val
    Here, we have an aggregation of Event, EM_Spec_Participant_Attrib, and Participant_Attrib. The relationship is between this aggregation and Participant. This relationship gives the values for the EM_Spec_Participant_Attrib filled in in by the Participant for this Event.

16. SEM_Spec_Participant_Attrib_Val
    Attribute:
    • Value The value for this SEM_Spec_Participant_Attrib_Val
    Here, we have an aggregation of SubEvent, SEM_Spec_Participant_Attrib, and Participant_Attrib. The relationship is between this aggregation and Participant.

This relationship gives the values for the `SEM_Spec_Participant_Attrib` filled in in by the `Participant` for this `SubEvent`.

3.2 Data Dictionary

3.2.1 Passwords

Attribute	Data Type	Constraints	Description
login_id	varchar(20)	Primary Key	Login Name
password	varchar(8)	not null	Password

This is the basic entity which represents users who can log onto the system. There might be logins for teams or contingents as well.

3.2.2 Participants

Attribute	Data Type	Constraints	Description
PID	auto-numbered	Primary Key	Participant ID
name	varchar(50)	not null	Participant Name

The set of all participants for all the events. This is why the autonumbering is required.

3.2.3 Event_Manager

Attribute	Data Type	Constraints	Description
login_id	varchar(20)	Primary Key	Login Name
event_id	varchar(15)	⇒ *Event*	Event ID
name	varchar(50)	not null	Event Manager's Name
email_id	varchar(50)	not null	Email Address

Table containing all basic information pertaining to an `Event_Manager`.

3.2.4 SubEvent_Manager

Attribute	Data Type	Constraints	Description
login_id	varchar(20)	Primary Key	Login Name
event_id	varchar(15)	$\Rightarrow$ *Event*	Event ID
subevent_id	varchar(15)	$\Rightarrow$ *Event*	SubEvent ID
name	varchar(50)	not null	Event Manager's Name
email_id	varchar(50)	not null	Email Address

Table containing all basic information pertaining to an `SubEvent_Manager`.

3.2.5 Individual

Attribute	Data Type	Constraints	Description
PID	auto_numbering	$\Rightarrow$ *Participants*	Individual's PID
email_id	varchar(50)	not null	Email Address
login_id	varchar(20)	$\Rightarrow$ *Passwords*	Login ID
name	varchar(50)	$\Rightarrow$ *Participants*	Individual's name

An individual who can participate in a subevent.

3.2.6 Contingent

Attribute	Data Type	Constraints	Description
login_id	varchar(20)	$\Rightarrow$ *Passwords*	Login Name of CL
name	varchar(50)	not null	Contingent Name
email_id	varchar(50)	not null	E-mail Address of (CL)
CL_name	varchar(50)	not null	Contingent Leader's (CL)Name

The contingent name is assumed to be unique for a given event. The contingent leader provides the event organisers with a means of contacting the contingent.

3.2.7 Team

Attribute	**Data Type**	**Constraints**	**Description**
PID	auto_numbering	⇒ *Participants*	Team's PID
login_id	varchar(20)	⇒ *Contingent*	Login name of CL
team_name	varchar(50)	⇒ *Participants*	Team Name
name	varchar(50)	not null	Team Name
email_id	varchar(50)	not null	Contact Email ID

A Team can also participate in a SubEvent. Team Name should be unique for a given sub event.

3.2.8 Team_Member

Attribute	**Data Type**	**Constraints**	**Description**
email_id	varchar(50)	Primary Key	Email address
name	varchar(50)	not null	Team Member's name

3.2.9 Team_Member_Of

Attribute	**Data Type**	**Constraints**	**Description**
PID	auto_numbering	⇒ *Team*	Team's PID
email_id	varchar(50)	⇒ *Team_Member*	Email address of CL

This table denotes team-members and their teams.

3.2.10 Judge

Attribute	**Data Type**	**Constraints**	**Description**
email_id	varchar(50)	Primary Key	Email address
name	varchar(50)	not null	Judge's name
phone	integer	not null	Phone Number
address1	varchar(100)	-	Address
address2	varchar(100)	-	Alternate address
fax	integer	-	Fax Number

The address of the judge mightnot always be required, but the phone number is a must because he/she will have to be notified in case of last minute changes.

3.2.11 Event

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	Primary Key	Event ID
event_name	varchar(50)	not null	Event Name(unique)
location	varchar(50)	not null	General Location
start_date	date	not null	Start Date
end_date	date	not null	End Date
capacity	integer	not null	Maximum Capacity
moderation_flag	integer	not null	Public/Private Flag
IC_flag	integer	not null	Individual/Contingent Flag

Table containing Event data.

3.2.12 SubEvent

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	⇒ *Event*	Event ID
subevent_id	varchar(15)	Primary Key	Sub Event ID
subevent_name	varchar(50)	not null	SubEvent Name
venue	varchar(50)	not null	Specific Location
capacity	integer	not null	Maximum Capacity for the Subevent
moderation_flag	integer	not null	Public/Private Flag
team_size	integer	not null	Team size

Table containing SubEvent data. Team size will be set to 1 for individual based events.

3.2.13 Registered_For

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	⇒ *Event*	Event ID
login_id	varchar(20)	⇒ *Passwords*	Login Name
approved	integer	not null	Approval Flag for participation

Indicates which users are registered for the event.

3.2.14 Participates_In

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
PID	integer	⇒ *Participant*	PID
approved	integer	not null	Approval Flag for participation

This relationship indicates which subevents an individual participates in and the result of that subevent.

3.2.15 Results

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
description	varchar(50)	-	Description of postion
result	varchar(50)	not null	Result in the Event.

The set of results for the Sub Event.

3.2.16 Judges

Attribute	**Data Type**	**Constraints**	**Description**
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
email_id	varchar(50)	⇒ *Judge*	Email address

Table which shows the subevent that a judge will be presiding over.

3.2.17 Occurs_On

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
day	date	Primary Key	A day within the event.
start_time	date	not null	Start time of the subevent.
end_time	date	not null	End time of the subevent.

The date is a day within the duration of the Event on which a subevent might be scheduled. It is a weak entity set and the value is constrained to lie within the values of `start_date` and `end_date` attributes of the `Event` entity.

3.2.18 EM_Spec_SubEvent_Attrib

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *Event*	Event ID
attrib	varchar(30)	Primary Key	Attribute name
mandatory_flag	integer	not null	Mandatory Attribute Flag

This relation represents the attributes specified by the `Event_Manager` for a `SubEvent`.

3.2.19 EM_Spec_SubEvent_Attrib_Val

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
attrib	varchar(30)	⇒ *EM_Spec_SubEvent_Attrib*	Attribute name
value	varchar(50)		Value of the Attribute

This relation represents the variable portion of the data for a `SubEvent`. It includes `EM_Spec_SubEvent_Attrib_Val` and `SEM_Spec_SubEvent_Attrib_Val`.

3.2.20 EM_Spec_Participant_Attrib

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *Event*	Event ID
attrib	varchar(30)	Primary Key	Attribute name
mandatory_flag	integer	not null	Mandatory Attribute Flag

This relation represents the attributes specified by the `Event_Manager` for a `Participant`.

3.2.21 SEM_Spec_Participant_Attrib

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
attrib	varchar(30)	Primary Key	Attribute name
mandatory_flag	integer	not null	Mandatory Attribute Flag

This relation represents the attributes specified by the `SubEvent_Manager` for a `Participant`.

3.2.22 EM_Spec_Participant_Attrib_Val

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *Event*	Event ID
login_id	varchar(20)	Primary Key	Login Name
attrib	varchar(30)	⇒ *EM_Spec_Participant_Attrib*	Attribute name
value	varchar(50)		Value of the Attribute

This relation gives the values for the `EM_Spec_Participant_Attrib` filled in in by the `Participant` for this `Event`.

3.2.23 SEM_Spec_Participant_Attrib_Vl

Attribute	Data Type	Constraints	Description
event_id	varchar(15)	⇒ *SubEvent*	Event ID
subevent_id	varchar(15)	⇒ *SubEvent*	SubEvent ID
PID	integer	⇒ *Participant*	PID
attrib	varchar(30)	⇒ *SEM_Spec_Participant_Attrib*	Attribute name
value	varchar(50)		Value of the Attribute

This relation gives the values for the `SEM_Spec_Participant_Attrib` filled in in by the `Participant` for this `SubEvent`. The table name was shortened with "Vl" at the end instead of "Val" due to naming restrictions in Oracle.

3.3 Data Flow

The Functional blocks in the GEM - a three-tier model

The GEM has been divided into three distinct tiers with clearly demarcated functions. (see figure 3.4 - at end of chapter)

3.3.1 The Database Layer

This layer consists of the backend database - in this case, oracle. The GEM stores event information in the form of tables as described by the ER diagram. (see figure 3.1, 3.3 - at end of chapter)

All events stored in the database have a fixed schema and are hence stored in the table `Event`.

3.3.2 The Servlet-FAA-JDBC layer

This layer is split into three functional sub-layers. Here is a broad overview of the functions handled by the the three sub-layers:

The JDBC Query Engine

This is the most basic sub-layer in this layer and handles the connection with the oracle database and the query process. It has the following main functions:

1. Query handling
2. Schedule generation
3. Results generation
4. Schedule Clash detection (has not been implemented)

The JDBC Query Engine hands to the next layer a neatly packaged object which holds the results of the query executed. For example, a scheduling request to the JDBC Query Engine will result in a Schedule object being passed back - a sort of a wrapper object, which holds formatting info for the Servlet layer.

The FAA layer - Filter and Authorization

This layer serves as a middleman - serving requests by the Servlet layer to the JDBC Query Engine to execute, after suitable filtering and formatting of the request. The term *filter* here refers to the transformations applied to the request to make it a formal request to the JDBC Query Engine.

Furthermore, this layer also serves as an authorizing layer - keeping track of what data different users are allowed to modify. This is necessary because the basic database will be stored under a single username.

Additionally, this layer receives results of a request made to the JDBC Query Engine and makes it available to the Servlet layer.

The Servlet layer

This is the topmost sub-layer in the middle tier. This directly generates the HTML files for the user interface and passes on requests made by users to the FAA layer to handle.

3.3.3 The Web Client - The Actual HTML interface

This forms the third tier of the GEM, and it is this that is used to use the GEM for all purposes: from creation of events to registration of users.

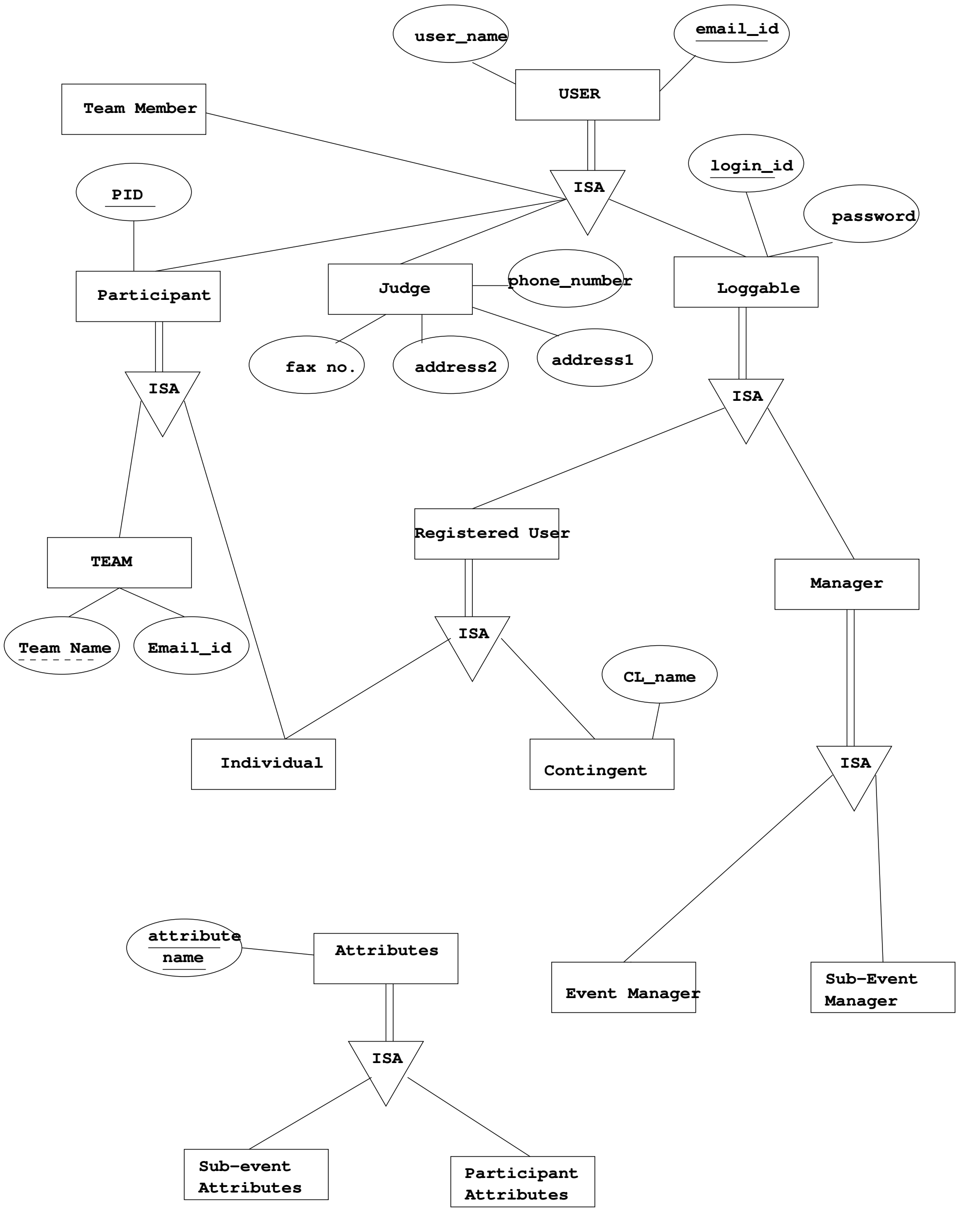

Figure 3.1: Hierarchy

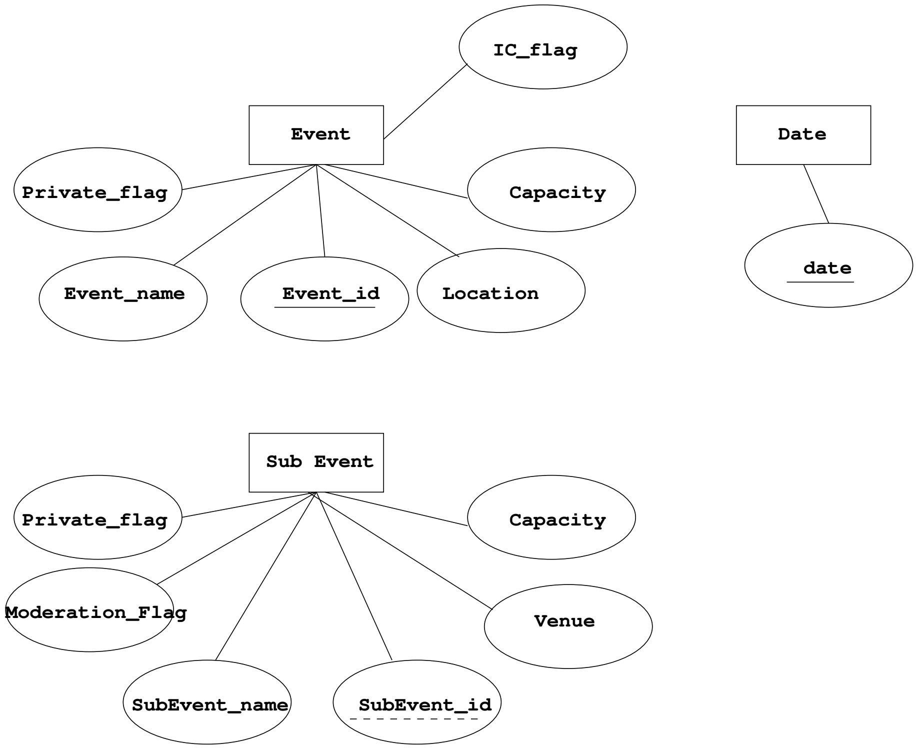

Figure 3.2: Entities

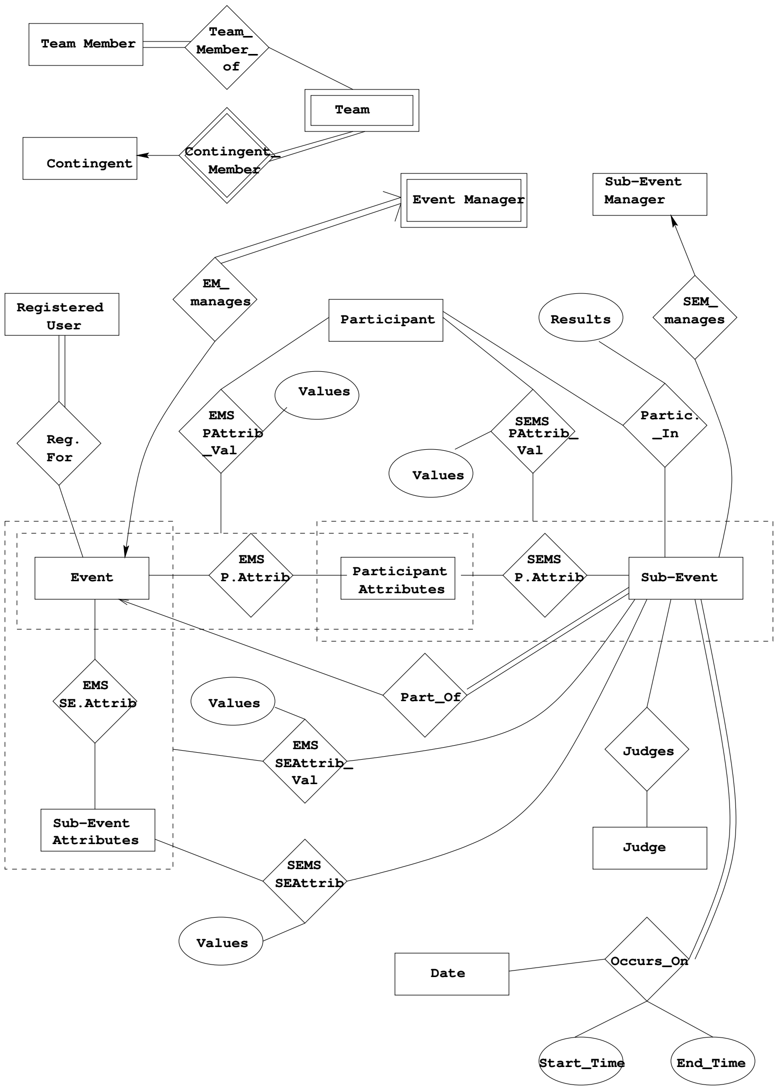

Figure 3.3: Our main ER diagram

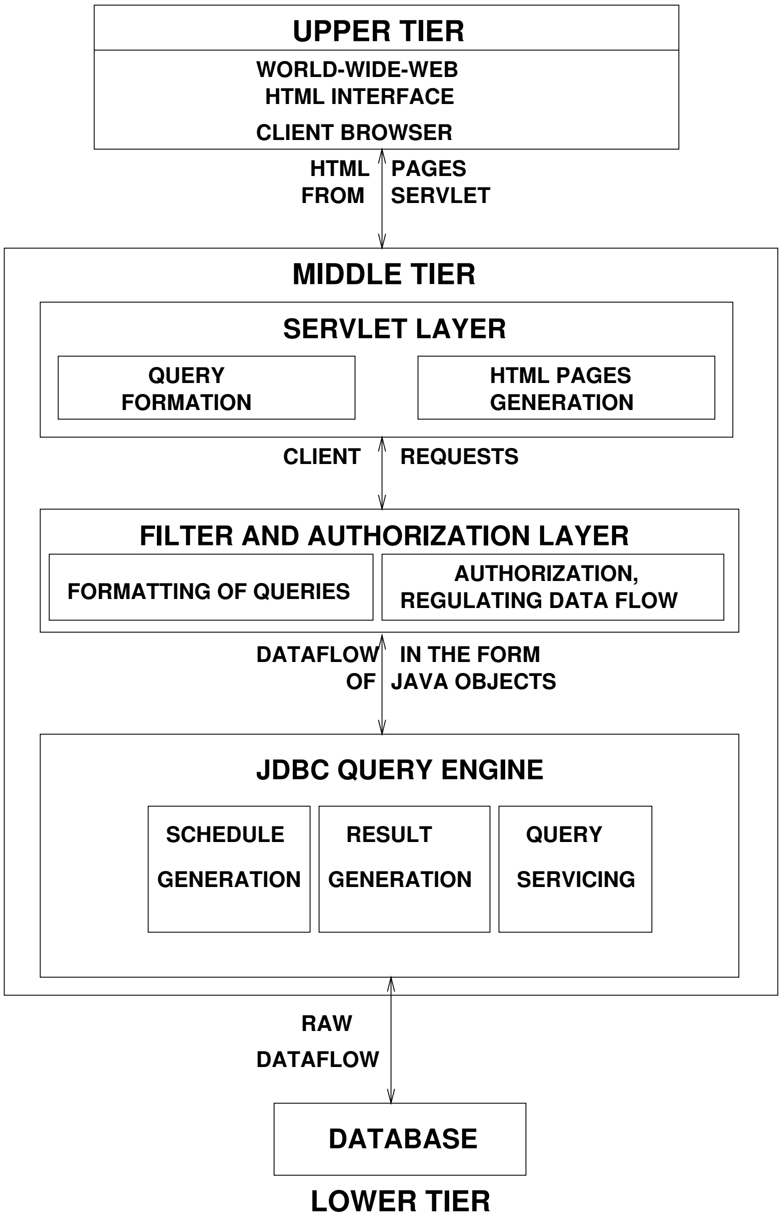

Figure 3.4: 3-Tier Model for Data Flow

Chapter 4

Functional Requirements

4.1 User Interfaces

The first screen for managers/participants is the login screen. If a participant has not already registered, he can then call up the register screen instead. Links to these two pages should be provided from the information page of the event. New users can thus register and then proceed to the login screen. Managers are automatically regsistered when their events/subevents are created. The SysAd has a separate static login page.

Subsequent data flow is different depending upon the type of the user using the system. Here is a brief overview of the data flow when the user in consideration logs in:

1. **The SysAd**

   (a) He is given a choice to create an event or view/remove the event catalog.(see fig 4.1)

   (b) If he chooses to create an event, then a form is served which allows him to fill in event details like event name, organizer login, organizer password, capacity constraints, and so on. These values, once set, are immuatble. Once this is done, an email is sent to the event organizer informing him of his login id and password, which he can use to now log in.

   Apart from this, an entry is made in the `Event` catalog.

   (c) If he chooses to view the event catalog, then a table of the events is shown. A link is provided so that he can view any specific event details. Alongside every event,a link is also provided to remove the event.

2. **An Event Manager**

   (a) The first time an event organizer logs in, he is asked to fill extra event details - such as start and end date, whether moderation required, event specific attributes for subevents, event specific participant attributes and so on. (see fig 4.3 - at end of chapter)

   (b) Once this is done, a control page with links to view/modify event details, create/view/delete subevent, process pending approvals, view participant details and view schedule is provided.

   (c) If he chooses to create a new subevent, he is then asked to insert basic attributes like the subevent manager's emailid, loginid, name, passwd etc.

   (d) If he chooses to process pending approvals, then he is served a table with these and is allowed to select those to include. Links will also be provided to view participant info.

   (e) If he chooses to view subevents, a list is generated with links to view all details. Additional links to remove a subevent are also provided.

   (f) If the schedule view is chosen, the standard schedule servlet is invoked to spew out the schedule.

3. **A Sub-event Manager**

   (a) The first time a subevent organizer logs in, he is asked to fill extra sub event details - such as start and end date(s), whether moderation required, venue, values of attribs specified by the EM, and also any attributes for participants who will register for that subevent.(see fig 4.4 - at end of chapter). He is also allowed to have the subevent span over more than a day. Suitable entries will be made in the `Occurs_On` table accordingly. *Again, these participant attributes are taken to be of type varchar(30).* Each extra participant attribute can be chosen to be mandatory or not, and participant-filled or sem-filled.

   (b) Next, he is shown a page which provides links to view/update subevent details, view participant details, pending approvals (if moderation is selected), view/update judge info, enter subevent results, or view schedule.

   (c) If he chooses to view/modify judge details, then he is served a list of all the current judges and allows him to view details, remove the judge and add judges. Functionality has also been provided to mail the judges from here.

   (d) If he chooses to view pending approvals, then he is served a table with these and is allowed to select those to include. Links will also be provided to view participant info.

   (e) If he chooses to insert results, then a page will be served to allow him to do so depending upon the participant-subevent tuple selected. The results will be displayed finally in the order they were entered.

4. **A new user**

(a) A new user can create a login using the register servlet. He is prompted for basic details like a preferred emailid, passwd, name and the like. Any event specific data is also asked for.

(b) This done, his registration is processed: if the event is not moderated, then the user is directly registered for the event, subject to capacity constraints, and he is informed of this on the webpage. If the event is moderated, his name is added to the list of people whose registration is pending approval. Once the event organizer approves of the registration, the user is sent a mail informing him of this. After this, the user can log in using the newly created login id and password.

5. **Individual participant registration**

   (a) Once a new user has obtained a login id, he can log in to register himself for various subevents.

   (b) To this end, he is shown a page with links to edit his profile, register for a subevent, and to view the event schedule.

   (c) If he chooses to edit his profile, he is allowed to change only *non-mandatory, participant-filled* values.

   (d) If he chooses to register for a subevent, a table of subevents *for which he has NOT registered* is shown. He can select whichever ones he wants and submit the form.*Once registered for an event, he cannot un-register for it now.*

6. **Contingent registration**

   (a) If an event is specified to allow only contingent registrations, then the users who log in now abstract to a whole contingent. *Any event which involves team participation for any subevent MUST be declared to be contingent based.* In this case, subevents which involve only individual registrations must register them as *single-member teams.* This is done to facilitate a uniform and efficient storage of participant data in the underlying database.

   (b) Furthermore, each contingent member must have a unique email id which will be used to register him as a member of a team to various subevents.

   (c) As before, once a contingent representative logs in, he is shown a page with links to edit the contingent profile, register a team for a subevent, and to view the event schedule.

   (d) If he chooses to register a team for a subevent, a table of subevents *for which a team has NOT been registered* is shown. To register, a link is given against each event. Clicking on this link leads him to a form with as many textboxes as the number of members in a team for that subevent. These should be filled by the email ids of the team members for the event.*Once registered for an event, a team cannot un-register for it now.*

4.2 Login and Verification

Participants who have created their accounts, the event and sub-event organisers and the system administrator can login to the database. Thus upon authentication, they will be presented with a certain set of permissions to modify and view the various features of the database. This, as mentioned earlier, is handled by the FAA layer.

4.3 Event Creation

The SysAd may create new events as they come. He will then also create accounts for the event organiser and the sub-event organisers, and allocate them a password. He will also set constraints at this time on the size of the event database.

4.4 Sub-event / Participant Schema Design

The event organiser must be able to prepare schemas for the sub-events well as the participants for his event. By this, we mean that the various fields of each sub-event that must be stored in the database are specified. Since the information required of each sub-event and participant differ from event to event, we must provide for this flexibility.

4.5 Sub-event Creation

The event organiser must be able to create new sub-events after designing their schemas. An option for deleting these sub-events is also provided.

4.6 Schedule Generation and Clash Indication

Schedules are generated for each event, according to the schedules of each sub-event. A participant having two sub-events whose schedules clash must be informed in advance.(Informing of clashes is not done)

4.7 Result Generation

The result for an event and its sub-events must be displayed. In addition, the result for each group of students must be displayed so that totalling of points may be done.(This functionality has not been provided)

4.8 Registration for sub-events

The participant must be able to register and un-register himself for new events.

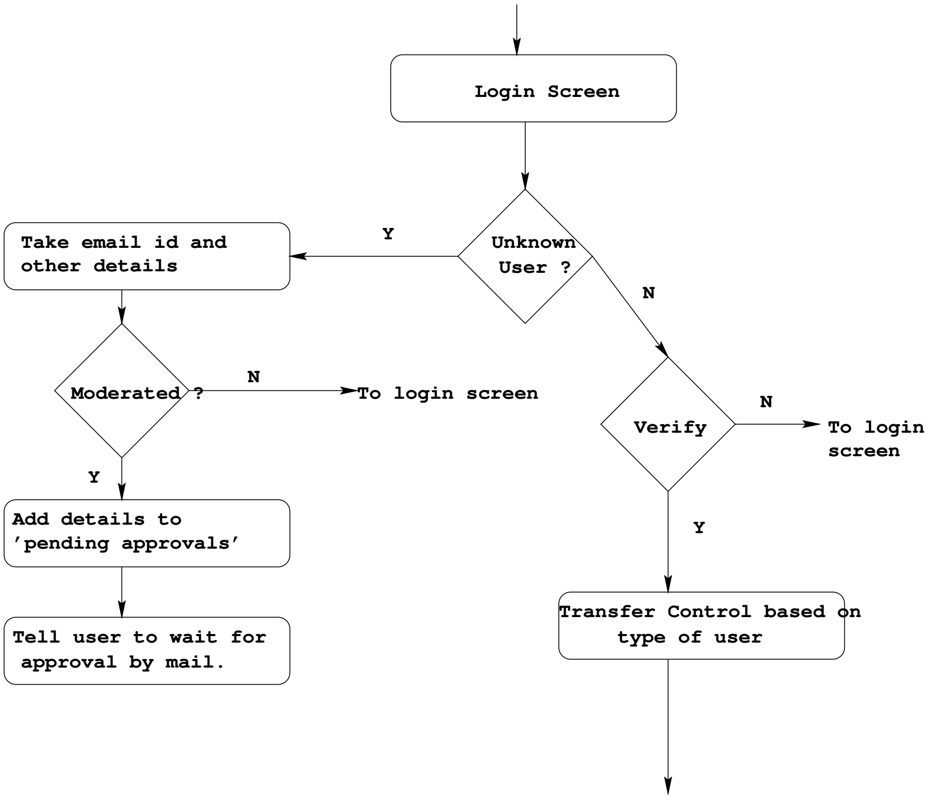

Figure 4.1: Login Interface

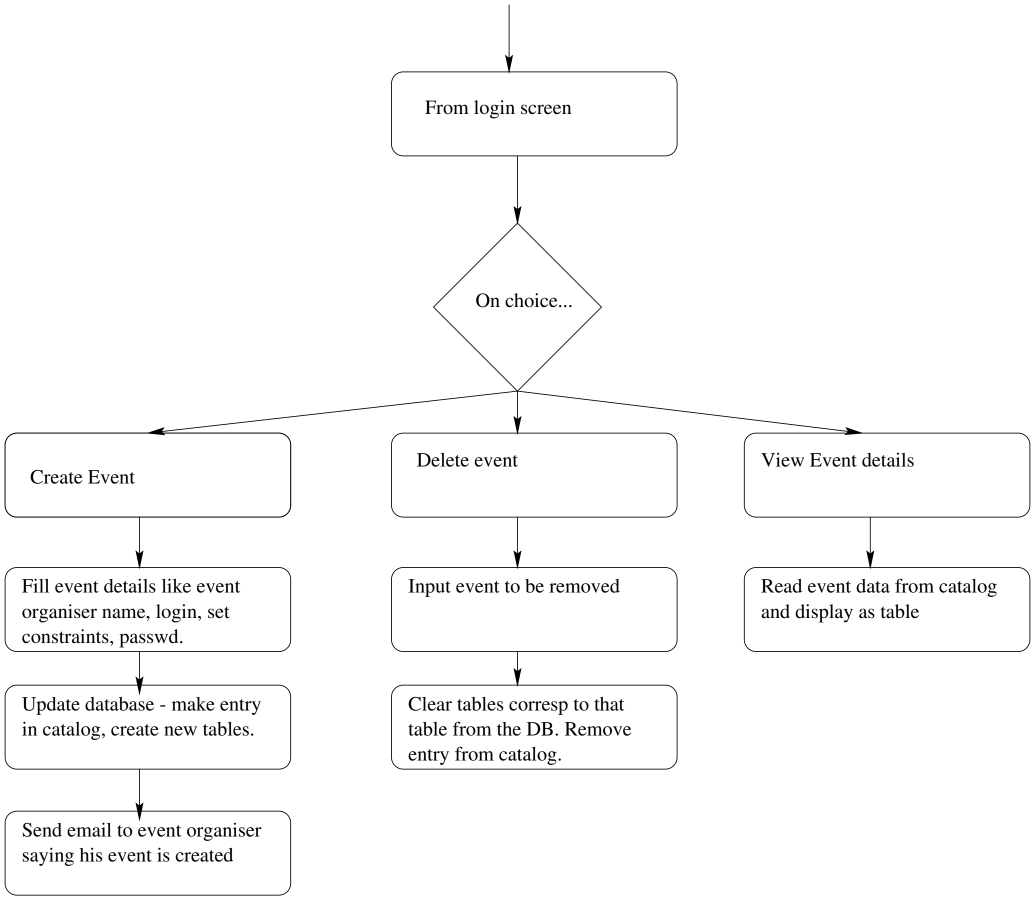

Figure 4.2: SysAd Interface

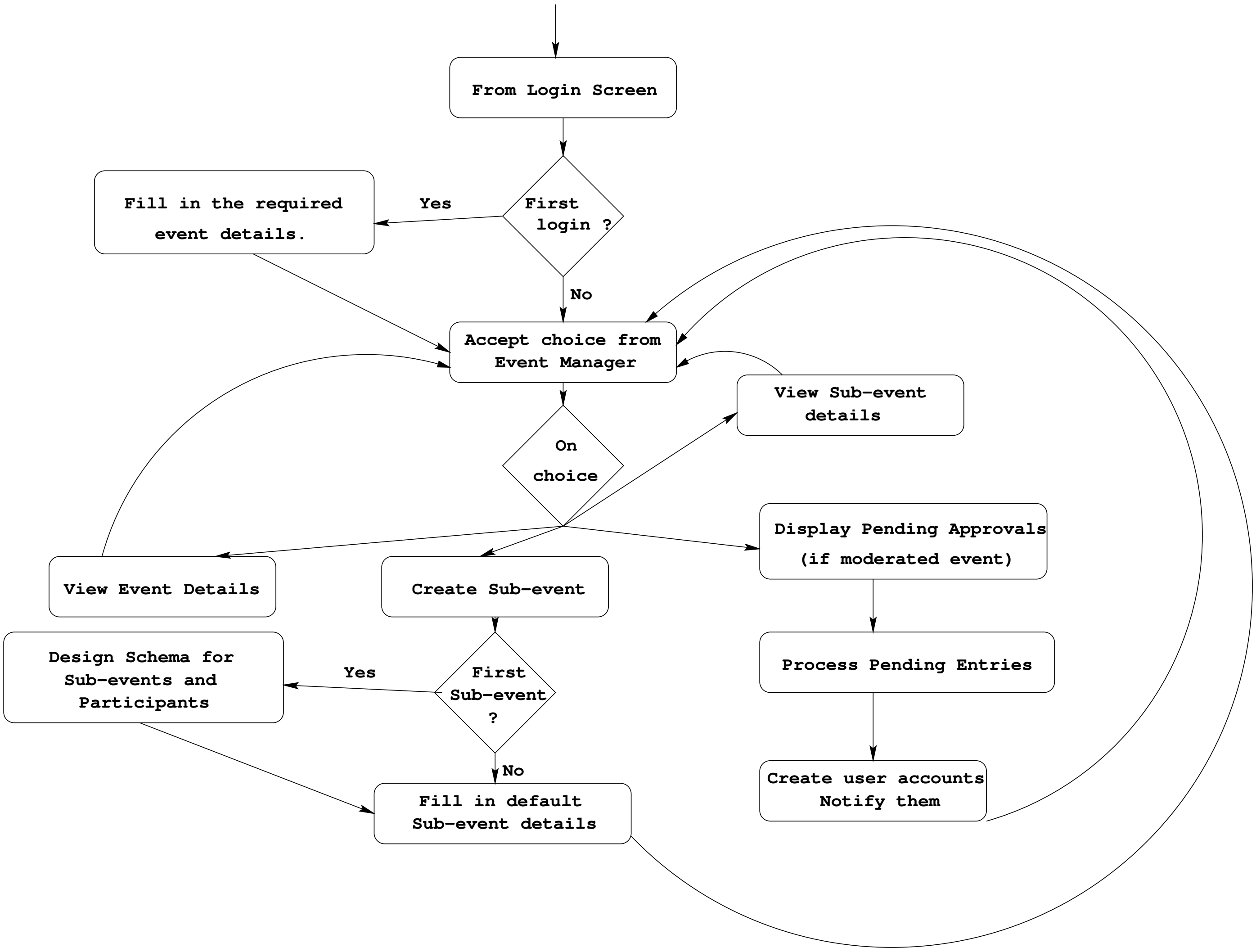

Figure 4.3: Event Manager Interface

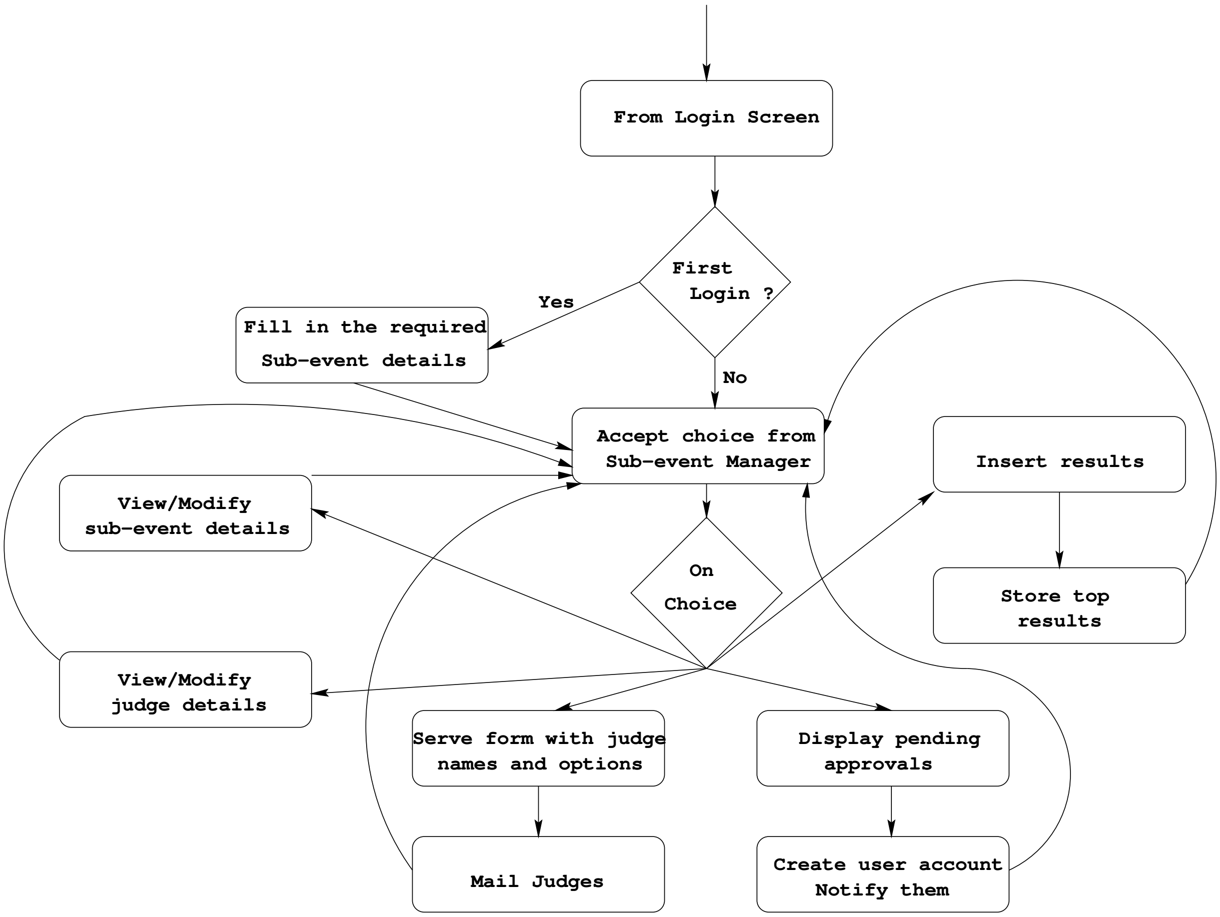

Figure 4.4: Subevent Manager Interface

Chapter 5

External Interface Requirements

5.1 User Interfaces

WWW will be used as the interface.

5.2 Hardware Interfaces

No Special Hardware Required.

5.3 Software Interfaces

5.3.1 Software Platforms

- Oracle8 Enterprise Edition Release 8.0.4.0.0

5.3.2 Software Packages

- Java 1.1.6
- Java Servlet Development Kit (JSDK) 2.0
- Oracle Thin JDBC Driver

Chapter 6

Performance Requirements

6.1 Capacity Requirements

6.1.1 Estimated number of users

Considering the events that are held in a institute like IIT Bombay and could be managed by our software, we have estimated the number of users in each category of users as follows:

User Type	Number of Users	Comments
System Administrator	1	The whole system is managed by a single system administrator.
Event Organiser	3-10	This equals the number of events that are managed by the database.
Sub-event Organiser	50-500	Events like the Annual Swimming Gala have upto 75 different sub-events, each possibly requiring a separate sub-event organiser while events like Mood Indigo have around 25 sub-events.
Participant	500-20000	A single quizzing event at Mood Indigo could have only 10 entries, while a single swimming sub-event could attract 40 participants in the elimination rounds.

6.1.2 Database Size

Assuming that there are a 5000 users for every event, and a total of 5 events at a time, and around 4kB for every user, we estimate that 100 MB would be sufficient.

6.2 Throughput

Again we calculate the extent of activity on our database using the IIT Bombay atmosphere as a model. Of course we incorporate the cardinalities of the various user classes while doing this estimation.

User Type	Average Access Frequency
System Administrator	1/fortnight
Event Organiser	1/day
Sub-event organiser	5/day
Participants	10/hour

Chapter 7

Design Constraints

7.1 Standards Compliance

Not Applicable.

7.2 Hardware Limitations

Not Applicable.

Chapter 8

Validation Criteria

8.1 Testing

We will simulate a real event, like MI, with the following number of users (tentative and approximate)

1. `SysAd` : 1
2. `Event Manager` : 1
3. `SubEvent Manager` : 10
4. `Participants` : 50

8.2 Test Runs

The various classes of test runs that may be run are as follows:

- **Organization of different events**
  We will store essentially different genres of events - say, MI, Swimming Gala, Internet Workshop, some conference, an Inter-hostel event etc in order to demonstrate the versatility of our project.
- **Organization of events that are contingent/individual based**
- **Organization of events that have varying participant requirements for different subevents**
- **Organization of events that have varying subevent characteristics**