



SWIM-Suite

A Swim Meet and Swimmer Management Software

Requirements Analysis Document

Submitted on: September 30th, 2010

Team Members (in alphabetical order):

Abhishek Bindiganavile

Robert Greenberg

Sedat Ozer

Jay Takle

Contents

1 Introduction	2
1.1 Purpose of the System	2
1.2 Scope of the System	2
1.3 Objectives and Success Criteria of the Project.....	3
1.4 References	3
2 Current Available System	4
3 Proposed System.....	5
3.1 Overview.....	5
3.2 Functional Requirements.....	5
3.3 Nonfunctional Requirements.....	6
3.3.1 Usability.....	6
3.3.2 Reliability.....	6
3.3.3 Performance	6
3.3.4 Supportability.....	7
3.3.5 Implementation	7
3.3.6 Interface	7
3.3.7 Packaging.....	7
3.3.8 Legal	7
3.4 System models.....	7
3.4.1 Scenarios.....	7
3.4.2 Use case model	9

1 INTRODUCTION

1.1 Purpose of the system

Organizing a swim meet is a time-consuming and difficult task that requires numerous managerial skills. Organizers have to pay attention to every detail of accepting entries from each swimmer, adding swimmers to the races for which they would like to be registered, seeding them according to their most recent timing and entering each race's results. Special care has to be taken to avoid human made mistakes and typos while accepting swimmer entries as well as publishing the results, when the process is handled manually.

The most traditional way among the organizers to process all abovementioned details and the information is using paper forms. Organizers provide swimmers with paper forms where they have to enter information such as swimmer name, age, race type and timing. Once the swimmers have the paper forms, they fill their details in, and submit their entries to the organizers. Organizers then refer these forms and put swimmers under the requested race. At this stage of the organization, the seeding also is handled manually and all this information is stored on sheets of papers.

Software developers have worked with organizers to develop software that helps the organizers during this process to enter, process and store all the swimmers information in a more secure, fast and convenient way. Meet Manager is such software that envelopes the features essential for handling each swimmer's data. Although this software is widely being used by the organizers, there are some missing features on it which would be useful to the organizers during the races. To include such features, we have met with swim meet organizers and identified key features missing in the current software.

Accepting the shortcomings of the existing software as a challenge, we propose a system that will make the swimmer registration process more flexible. The system will allow the swimmers to enter their own data and their own registration requests to the system without requiring an authorized person to enter such data. The system will also provide swimmers with a tool to assess their progress.

1.2 Scope of the system

The goal of developing SWIM Suite is to provide software which would make organizing and participating in a swim meet easier. SWIM Suite will have functions for both the organizers and the swimmers. The swimmers will be able to register for races without the need of

waiting for an approval of an organizer authority. Moreover, they will be able to check their previous swim meet results, thus will have the opportunity to assess how their race timings are progressing.

The organizers can setup a new swim meet in the database for the swimmers to register, seed every race, which is basically allotting lanes to all the swimmers in a particular race. After the meet is over, the organizers can enter the results of each race in the system, so that the swimmers can come back and review their progress.

The system is currently to be implemented as a stand-alone software package. After successful deployment in swimming pools and clubs, it may also be implemented as a web based system, through which both the organizers and the swimmers to access the system remotely.

1.3 Objectives and Success Criteria of the Project

After a brief discussion with the swim meet organizers, following are the objectives of our system:

1. To help organizers setup a detailed database for an entire swim meet.
2. To allow organizers to manage races in a meet.
3. To allow swimmers to create their profile.
4. To allow swimmers participate in a swim meet.
5. To seed/allot lane number to swimmers according to their timings.
6. To save results of a race, so that they are accessible to swimmers.
7. To display a swimmer's progress over a period of various meets, with respect to his/her race timings.

Following are the success criteria for our system:

1. System meets all objectives.
2. System guarantees all data is secure.
3. System is reliable and easily usable.

1.4 References

<http://www.hy-tek ltd.com/swim/mm/index.html>

2 CURRENTLY AVAILABLE SYSTEM

Currently the organizers either use the traditional paper forms for managing the whole meet or use “Meet Manager” software. First developed in 1985, “Meet Manager” has been upgraded over the years and is now available as “Meet Manager 3.0”. It is a stand-alone application storing the swim meet data in Microsoft Access format. The software’s price, which starts at \$239, is beyond what many swim clubs could afford and that is why many swim clubs still use the traditional paper method.

Since there is already software available in the market, we consider its most current version as a reference and try to build a superior system.

Functionalities of the current software:

- a) Create a database for each race
- b) Create a profile for each swimmer
- c) Populate race database with swimmers
- d) Seed each race
- e) Store results

As seen from the above functionalities, Meet Manager is made solely for the organizer, there are no features for the swimmer. Also, the swimmers still have to use a paper form provided by the organizer, fill their details, races and send it back to the organizers. The organizer then fills these details manually into the Meet Manager system.

Shortcomings of the current system:

- a) Paper work is necessary in the initial stages of the meet.
- b) Once the entries are submitted, the swimmer cannot edit any information, even if the information has been entered incorrectly.
- c) Organizers have to input each swimmer’s information manually into the system, which is inefficient and frequently results in errors.
- d) Swimmers do not have a unique profile connecting swimmers past races.
- e) Swimmers cannot assess their progress for the past races through this system.

3 PROPOSED SYSTEM

3.1 Overview

As mentioned in the earlier section, the earlier phase of managing the meet, which is mainly the process of transferring entries from the swimmers to the organizers, is still done through a paper form. After which, the entries are populated into the Microsoft Access database by the organizer using the Meet Manager software. This process has a potential to create errors while the swimmers and their timings are being entered into the system.

To make our system comparable to this commercial system, we have met the swim meet organizers of Rutgers swimming pool to discuss their requirements for software which would help them organize a swim meet efficiently. On the basis of their requirements and new ideas put forward by our development team, we propose a standalone system which will completely remove the process of 'paper form' submission, envelope the features necessary for swimmer database management and present the swimmer with a tool to assess their own progress.

3.2 Functional Requirements

As per the requirements of the organizers, there are two types of users who will be using this system:

Firstly, the organizer must be able to:

1. Manage a swim meet by:
 - (a) creating a database for a meet with detailed information about the meet viz. Meet Name, Location, Start date, End date, Age-up date, Entry deadline, Type of course, Type of meet.
 - (b) editing the meet details.
 - (c) deleting the whole meet.
2. Manage different races in the meet by,
 - (a) creating a race with information such as, Event number, Gender, Number of Lanes, Age group, Stroke type and Distance.
 - (b) deleting a race.
 - (c) editing details of a race.
3. Seed every race according to the timings provided by the swimmers.
4. Organize all the race's to form a schedule that can be distributed amongst the swimmers on race day.
5. Enter race results and store them.

Secondly, the swimmers should each be able to:

1. Create their profile with details, including:
 - (a) Swimmer's Last Name, First Name
 - (b) Date of Birth
 - (c) Gender, and
 - (d) Team.
2. Receive a unique identification number.
3. Login to the system.
4. Edit their profile.
5. Direct the system to display a list of past meets in which they have participated (if any).
6. Participate in a meet by:
 - (a) Requesting the system to show a list of races for which they are eligible, according to their profile.
 - (b) Participate in races of their choice.
 - (c) Receive a confirmation of their participation.
 - (d) Edit the races in which they have participated .
7. Request the system to display a progress graph for a specific race, on the basis of their timings in the earlier meets (if any).

3.3 Non-Functional Requirements

3.3.1 Usability

1. The interface for swimmers should be operable by people of all age groups beyond 10 years of age to, and focusing mostly on, the teenage group, which can be verified once the software is ready.
2. Menus on the system must have clear directions on how to request a specific feature.
3. Feature buttons on the menu should have words/icons pertaining to the feature.
4. The system should have a document to help organizers use the system.

3.3.2 Reliability

1. The privacy of the swimmers and the organizer should be guaranteed.
2. The system should ask the organizer once, before deleting a race or meet.
3. The system should notify the swimmer and organizer if they modify any data.

3.3.3 Performance

1. Eligible races for a swimmer should be calculated using swimmer profile.
2. Swimmers should be seeded, according to the timings they have provided.

3.3.4 Supportability

1. The system should be able to accept updates to any of its existing modules.
2. The system should be able to accept changes to any conventions being used.

3.3.5 Implementation

1. The system should run on any version of a Microsoft Windows operating system.
2. The system should be developed and implemented using JAVA and Eclipse IDE.

3.3.6 Interface

1. The system should be independent of any existing meet managing systems.

3.3.7 Packaging

1. The system should be first installed on the computer to access it.
2. No other software should be required to access the system.

3.3.8 Legal

1. The system and its operations comply with all local and federal laws.

3.4 SYSTEM MODEL

3.4.1Scenarios

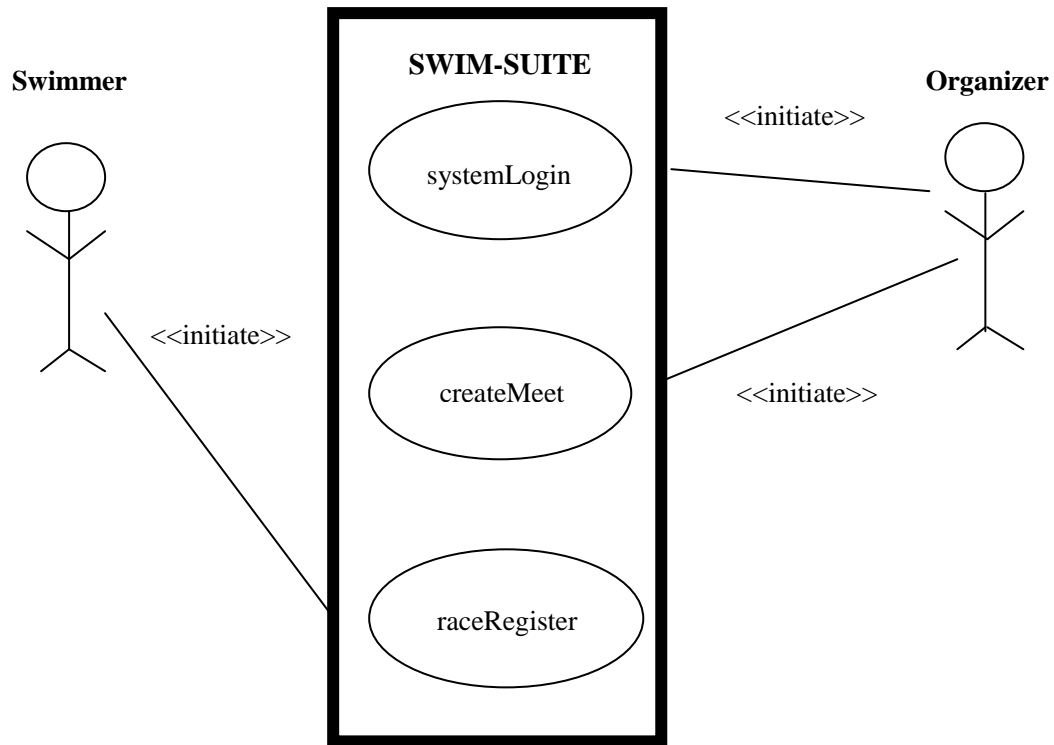
Scenario name	SwimmerRegister
Participating actor instances	Tom : Swimmer
Flow of events	<ol style="list-style-type: none">1. Tom arrives at the Recreation Center, and requests for the registration screen2. Tom fills in the registration screen with his/her information like name, date of birth, gender, club affiliation.3. The input data is saved in a form on the system and an acknowledgment is displayed to Tom.4. The confirmation is sent as an email to Tom.5. Tom is given a unique identifier.6. Tom saves the unique ID and exits the registration process.

Scenario name	raceSignup
Participating actor instances	Tom: Swimmer
Flow of events	<ol style="list-style-type: none">1. Tom arrives at the Recreation Center and activates the login screen2. Tom logs into his/her profile using the unique identifier that he/she possesses3. The race page is then displayed on the screen based on the physical profile of the Swimmer with all races that Tom is eligible for4. An option is given to Tom to register at the moment or later.5. If register later is chosen, proceed to the log out option.6. If register now is chosen, the Tom chooses the races that he/she is eligible for and the ones that he/she is likely to participate in.7. Tom receives a confirmation page detailing every race event that he/she registers for, apart from an email for the same to Tom's address.8. Tom logs out.

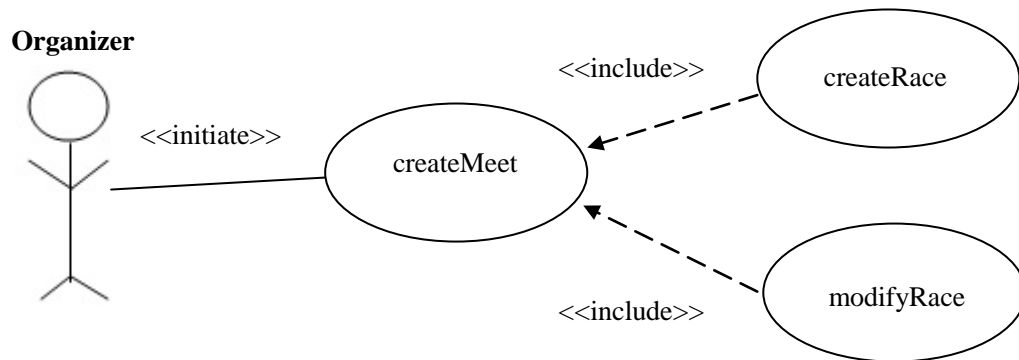
Scenario name	viewStatistics
Participating actor instances	Tom: Swimmer
Flow of events	<ol style="list-style-type: none"> 1. Tom arrives at the recreation center and requests for the login page. 2. Tom logs into the system with his unique identifier. 3. Tom selects the view results page where his race timing from his first registration to the present are stored and he can get a fair comparison from his lowest timed one to his fastest race timing. 4. Also, Tom is given the option of viewing the pictorial representation of his/her progress over a certain period of time. 5. Tom logs out of the system.

Scenario name	meetManage
Participating actor instances	Bob: Organizer
Flow of events	<ol style="list-style-type: none"> 1. Bob logs into the system using his administrator id and password. 2. Bob first sets up a meet with the following requirements in terms of age group, gender and club affiliation. 3. Bob decides the date and location of the meet and enters it into the system for displaying it to the Swimmers. 4. Bob decides on an age-up date and calculates the age of all the Swimmers who have registered for a particular meet. 5. Based on the number of registrations received, Bob, now, can add/create heats for a particular race in the meet. 6. Through the duration of the meet and registration limit, Bob can change/alter any option in the meet and have it show up on the Swimmer's side 7. Bob maintains a database of the results of the races in the meet in order to facilitate easier operation when the Swimmers want to view their respective results. 8. At the end of the registration period, Bob stops the system from accepting new registrations and proceeds to the age-up period. 9. Bob has all the applicants sent out an email confirming their registration for a race in the meet.

1.1.1 Use case model



High level use case diagram for system Swim-Suite



Detailed use case diagram refining the createMeet use case

2) System-level use case description

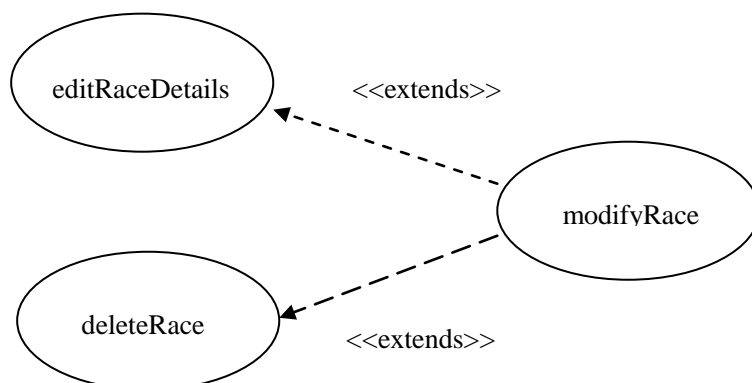
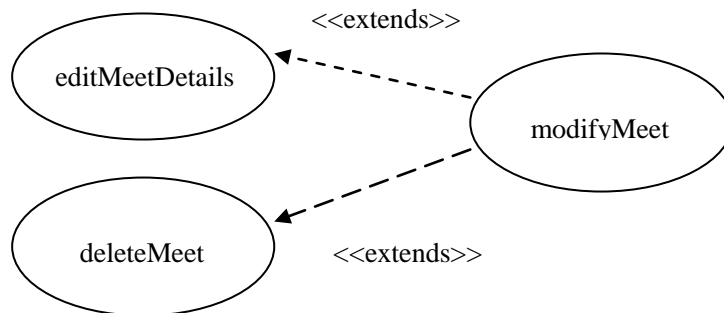
Use case name	SystemLogin/Register
Participating	Swimmer Organizer
Flow of events	<ol style="list-style-type: none"> 1. The Swimmer or Organizer requests the registration/login page and fills in his/her information. 2. User is given an unique identifier to access his/her profile. The Organizer is given administrative authority over the system. 3. Once the register form is filled, the user is displayed a confirmation of registration page complete with a unique identifier. 4. The user exits this page.
Entry condition	The user is on the homepage of the swimming event manager.
Exit conditions	The system stored the profile details in the database and sent a confirmation to the user.

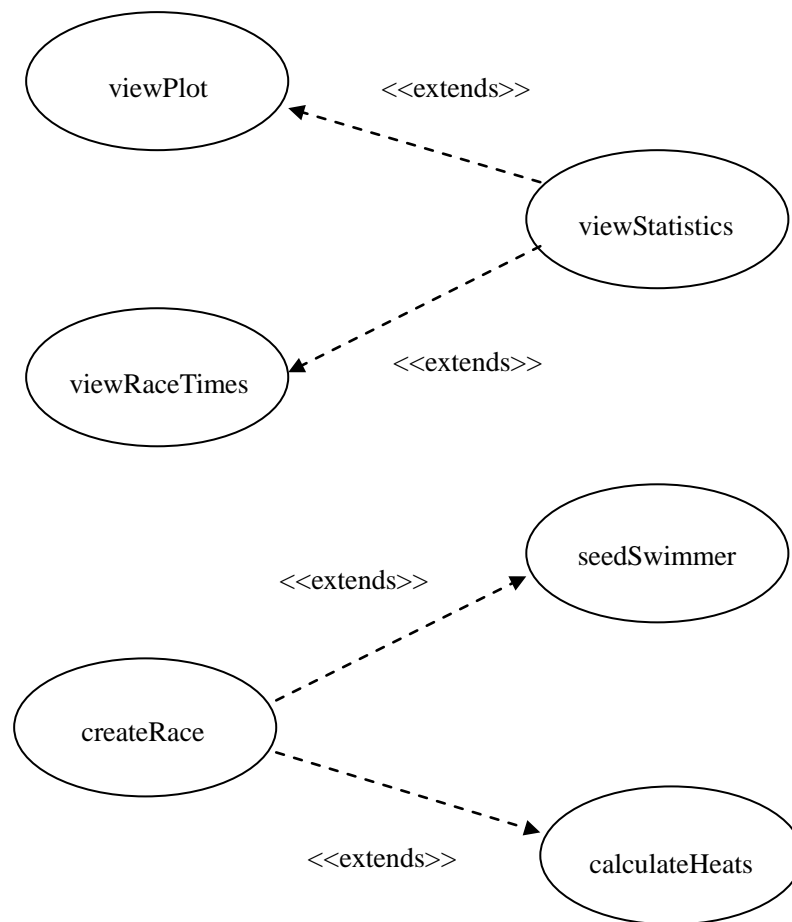
Use case name	CreateMeet
Participating	Initiated by Organizer
Flow of events	<ol style="list-style-type: none"> 1. Organizer logs into the system with his admin details 2. Organizer inputs the name of the meet, location, start and end date of the meet, deadline for registration and the age-up date. 3. With these details and the type of meet, the organizer completes the creation of the meet. 4. The confirmation for the creation of the meet is displayed on the screen 5. Organizer logs out of the system
Entry condition	Organizer logs into the system with his admin details.
Exit condition	Organizer logs out of the system upon receiving confirmation

Use case name	RaceRegister
Participating	Initiated by the Swimmer
Flow of events	<ol style="list-style-type: none"> 1. The Swimmer logs into the system with his unique identifier. 2. Based on his profile, the Swimmer is prompted to view the races he/she is eligible for. 3. Upon race lookup, the Swimmer is prompted to register for the races. 4. Once registered, the Swimmer is requested to confirm his registration. 5. The confirmation is now displayed to the Swimmer and is prompted to save the schedule in his profile. 6. Swimmer saves the schedule and logs out of the system.
Entry condition	Swimmer is logged onto the system with is unique identifier
Exit condition	Swimmer confirms his/her registration and exits the system.

Detailed use case description

1)





Use case name	ViewStatistics
Participating	Initiated by Swimmer
Flow of events	<ol style="list-style-type: none"> 1. The Swimmer logs into the system with his unique identifier. 2. The Swimmer chooses a particular race from an earlier event. 3. The page containing the racetimes from the earlier race is displayed to the Swimmer. 4. The Swimmer is prompted for viewing a graphical representation of his/her performance over the earlier races/events/meets. 5. The Swimmer can request this page for any race/event that he/she has competed in, in the past. 6. Following the display of the stats page, the Swimmer is prompted to exit the system or choose another meet/race.
Entry condition	Swimmer logs into the system and requests for his/her statistics in the races/meets he/she competed in the past.
Exit condition	The Swimmer views the trend in his/her race timings and exits the system.

Use case name	SeedSwimmer
Participating	Initiated by Organizer.
Flow of events	<ol style="list-style-type: none"> 1. The organizer logs into the system past the age-up date and invokes the database for all the registrations for the meet. 2. The organizer then sorts the Swimmers based on the races registered for. 3. Organizer invokes the earlier racetimes, if any and proceeds to seed the Swimmers, with the fastest race timing in the center and the slower racetimings in the adjacent lanes 4. Once the seeds and the lane placements are decided, organizer creates a schedule with the lane placements. 5. The organizer confirms the seedings and exits the system.
Entry condition	Organizer logs into the system in order to seed the Swimmers for the individual races.
Exit condition	The organizer confirms the seeds and creates a schedule of the races and exits the system.

Use case name	ModifyMeet
Participating	Initiated by the Organizer
Flow of events	<ol style="list-style-type: none"> 1. The organizer logs into the system using his/her admin details. 2. The organizer chooses the meet to be modified. 3. The organizer then chooses to edit/add/remove the meet 4. A confirmation is obtained for the changes performed on the meet. 5. The Swimmers are notified of the changes made via a notification on their profiles. 6. The organizer exits the system.
Entry condition	Organizer logs into the system to modify the meet of his choice
Exit condition	Organizer logs out of the system following the confirmation and the notification to the Swimmers.

Use case name	ViewPlot
Participating	Initiated by the Swimmer
Flow of events	<ol style="list-style-type: none"> 1. The swimmer logs into the system and chooses the race event of his choice. 2. A dialog box prompts the user to choose to view either the racetimings for a particular race or the progress plot for the race over time in the past. 3. The swimmer chooses the progress plot. 4. The system shows the progress plot to the user on a page. 5. The swimmer logs out of the system
Entry condition	Swimmer logs in with his unique identifier
Exit condition	Swimmer logs out following the display of his/her progress plot.
Quality Requirements	The progress plot has to be shown for all races the swimmer has competed in, in the past in order to get a noticeable trend in his/her progress.

Use case name	ViewRaceTimes
Participating	Initiated by the Swimmer
Flow of events	<ol style="list-style-type: none"> 1. The swimmer logs into the system and chooses the race event of his choice. 2. A dialog box prompts the user to choose to view either the racetimings for a particular race or the progress plot for the race over time in the past. 3. The swimmer chooses the racetimings for all races competed in the past. 4. The system shows the racetimings of all the races the swimmer has competed in, as a list on a page. 5. The swimmer logs out of the system
Entry condition	Swimmer logs in with his unique identifier
Exit condition	Swimmer logs out following the display of his/her racetimings
Quality Requirements	The racetimings have to be listen with the presence of the winning racetime along with the difference in the two racetimes for each race, to enable easy performance analysis for the swimmer.

Use case name	EditMeetDetails
Participating	Initiated by the Organizer
Flow of events	<ol style="list-style-type: none"> 1. The organizer accesses the system with the admin details 2. The organizer is prompted with the message to edit the various parameters of the existing meet. 3. The organizer modifies the existing meet by changing the various parameters of the meet and confirms the changes. 4. The system saves these changes and displays a confirmation to the organizer. 5. The organizer logs out of the system.
Entry condition	Organizer logs in with the admin details in order to modify an existing meet.
Exit condition	Organizer saves the changes and exits the system.
Quality Requirements	The changes have to apply throughout the description of the meet and should throw up an exception in the event of a conflict of any functional parameter like location or time.

Use case name	DeleteMeet
Participating	Initiated by the Organizer
Flow of events	<ol style="list-style-type: none"> 1. The organizer accesses the system with the admin details 2. The organizer is prompted with the message to edit or delete any existing meet 3. The organizer selects a meet that is required to be removed at that point. 4. The system asks for confirmation of the deletion of the meet. 5. Upon confirmation, system deletes all entries belonging to that meet and saves the changes. 6. The system prompts a question to the organizer whether to notify the swimmers about the deletion of the meet or not. 7. Depending upon the response of the organizer, the system reacts accordingly. 8. Organizer logs out of the system
Entry condition	Organizer logs in with the admin details in order to delete an existing meet.
Exit condition	Organizer saves the changes and exits the system.

Quality Requirements	The system prompts a comments section where the organizer is required to enter the reason for deletion of meet.

Use case name	ModifyRace
Participating	Initiated by the Organizer
Flow of events	<ol style="list-style-type: none"> 1. The organizer accesses the system with the admin details 2. The organizer is prompted with the message to choose an existing meet 3. The organizer selects a meet that he/intends to modify races in. 4. The system prompts the list of races in that meet. 5. The organizer selects a particular race that he/she wants to modify. 6. Upon modification, the system prompts for confirmation. 7. After confirmation, the system prompts for modification of any more races or to add races. 8. If organizer wishes to add races to existing meet, he/she would choose that option and the system would direct the organizer to the createRace page. 9. After saving the changes, system displays a confirmation 10. The organizer exits the system
Entry condition	Organizer logs in with the admin details in order to modify an existing race.
Exit condition	Organizer saves the changes and exits the system.
Quality Requirements	The system should throw up an exception if there are conflicting parameters in races.