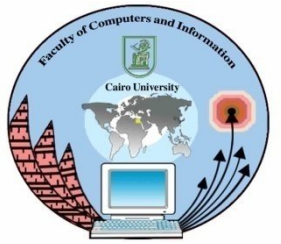


Course: Software Engineering -1

TA: Sara Tarek

Academic Year: 2014

Project: Evently

Software Requirements Specification

Leader: Nouran Atef 20120434

Contents

Team

ID	Name	Email	Mobile
20120434	Nouran Atef (leader)	noura.fcian@hotmail.com	01113187126
20120489	Reem Mohamed	Reemsheeha.rs@gmail.com	01129126166
20120396	Malak Medhat	Malak.medhat.mahmoud@hotmail.com	01005108718
20120460	Yussef Saleh	yussuf_saleh@outlook.com	01112510251
20120389	Mostafa Nasser	mos_man90@yahoo.com	01110562439

Document purpose and audience

This document is to describe our project (Evently) and to decide the purpose and the requirements for the program.

The audience for this document is the Product owner.

Introduction

Software Purpose:

This software will make creating and scheduling events easier, and will reduce effort and time in inviting people to the event. Also it will document all the previous events which will make searching for the events easier than before.

Software Scope:

Evently is an event organizing website which targets two types of users:

- User A who is called "Main user", is the user who creates events, puts descriptions for them, determines the place, date and the prices of the tickets and makes the tickets available online. Also can invite his/her friends and can answer all the questions of attendees.

- User B who is called "attendee", is the user who can discover all events even old events, search by categories, can invite his/her friends, and ask anything he/she wants about the event. Also can buy the tickets online.

Definitions , acronyms and abbreviations

Word	Definition
Evently	It's the name of the software system that aims to easier the users' life by allowing them to create and join different events according to their interests.
Main user	He is the user who is responsible for creating, managing, inviting and deleting events of his interests and preferences.
Attendee	He is the user who attends any event he likes and buys the online tickets.

Requirements

Functional Requirements:

"Evently" is a Software System where it allows different users to:

1- **Create** personal accounts with their own information and preferences.

* The user can create an event by clicking on Create Button, where he will fill some of his personal information like: Name - Age - Gender - Preferences (interests). Then he clicks on Submit Button such that his account is created with his information.

2- **Edit** their accounts by changing their privacy settings or edit in their own information.

* The user can change in his own information or his privacy settings by clicking on Edit Button in his profile.

3- **Create** an event and provide main information like: name and category of event -place - date - day - tickets and their prices.

* The user can create an event of his own by clicking on Event_Creator Button in his profile where he will fill the event's details like: Name - Category - Day - Date- Time - Place - if there were Tickets or not and their prices - Attendees to invite. The event can has mix of different categories as sub options to the category.

4- **Edit** an event by changing its date, day, place or privacy settings.

* The user or the creator of the event can change in the details of the event:

Name - Date - Day - Time - Privacy settings by clicking on Edit_Event Button and Submitting Button.

5- **Manage** an event and organize it by inviting users (attendees).

* The creator of the event can manage the event by inviting different users (attendees) or rejecting users' requests of joining the event by clicking on either Invite Button or Decline Button.

6- **Delete** an event.

* The creator of the event can remove the event from the system and by default it would be removed from his stream or the attendees' stream by clicking on Delete_Event Button.

7- **Join** an event.

* The user can request to join a certain event if the event's privacy wasn't public otherwise join directly by clicking on Join_Event Button.

8- **Buy** tickets for joined (attended) events.

* The user can reserve any number of tickets and choose the way of online paying.

9- **Leave** an event.

* The user (attendee) can leave the event by clicking on Leave_Event Button.

10- **Create** their own streams where they can add different events.

* The user can create a stream full of his created events or joined events and organize them either according to his preferences or the date of the event by clicking on Create_My_Stream Button.

11- **Follow** different users.

* The user can request to follow different users according to their preferences by clicking on Follow Button.

12- **Receive** notifications and emails about the upcoming events and recently added events.

* The user will receive notifications either from the system or his email.

13- **Suggest** events for users according to their interests (preferences).

* The user will get many suggestions of newly created public events according to his preferences.

14- **Search** for different events of specific category (criteria).

* The user can search for different events to attend by clicking on Search Button and choose the category.

15- **Add** photos or videos about attended events and share them to different users.

* The user or creator of the event can add photos and videos on his profile or the event page.

Non-Functional Requirements

1- Usability:

a- The system will be simple user-friendly with simple interface.

b- The buttons will have significant name to be understandable and there will be definition of the functionality of each button.

c- There will be “Tip Button” to explain how to use the system in the right way.

2- Reliability:

a- The system will have many security measures to guarantee the system’s effectiveness.

b- The system will work more than 90% of the time without crashing but if it crashes that will be because of the high usage of the system by many users at the same time performing the same function.

c- The system will be working 24 hours per day for at least 3 years before it could crash.

3- Performance:

a- The loading time for the system is less than 10 seconds else if there is a problem in the connection then it can take up to 20 seconds maximum.

b- The response of the buttons won’t take more than 3 seconds to apply the request.

c- The performance of the system will be of high quality to guarantee high degree of user's satisfaction.

4- Supportability:

a- The system and its functionalities will be supported 24 hours for different actions from the same user.

b- The system can support any event with any mix of categories for the same event as sub-option.

c- The system will be supported by different languages if needed as an update in the system software.

5- Implementation:

a- The system will be developed and implemented by Java Language and can be implemented by different programming language as an update in the software system.

6- Interface:

a- The system will be simple user-friendly with simple interface.

b- The user interface is based on a main window with a light colored background, where at the upper left most part of the window there will be The system's logo and name, at the upper part of the window there will be a bar full of different buttons like: log_in - sign_up - create an event - profile, at the middle of the window there will be the most upcoming events and the attended events, at the right part there will be search_textfield for searching for a certain event and list of categories and at the bottom there will be a message stating: " Welcome The user's name".

c- There will be pop-ups for the notifications.

7- Legal:

a- The development of the system software shouldn’t cost more than 1000 LE, so many users can reach it. Also the cost of maintainability shouldn’t cost a lot.

b- The system will cover its expenses through advertisements.

8- Availability:

a- The system should be available 24 hours for easily use the system at any time of the day.

b- The system can’t be down for more than 5 minutes once every 10 to 15 years.

9- Maintainability:

a- The maintenance of the system should be regularly every 2 months to check if there are any bugs or problems facing the user.

b- There should be monthly survey to be filled by the users to check if the users are satisfied by the quality of the system’s level of service and if they need any updates.

c- There should be developers and testers to check the system once a week.

d- There should be a notification for the user if there will be any maintenance of the system so the user could take into his consideration that the system could be stop for a certain period and the notification should be a week before the maintenance everyday.

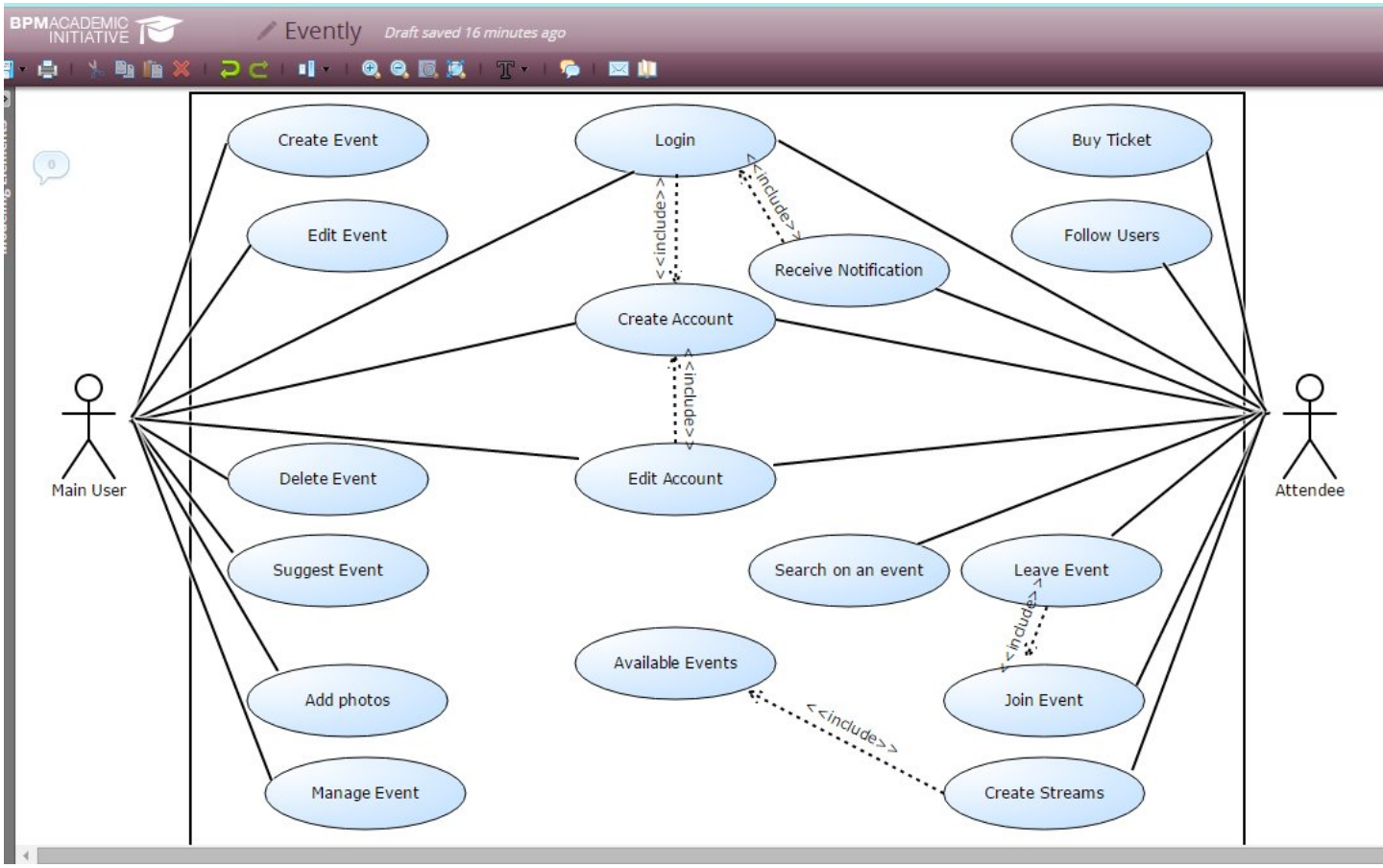
10- Recovery:

a- The system shouldn’t take too long, max 3 seconds to recover from any crash to guarantee high quality of service.

b- There should be a lot of back-up plans for recovery to guarantee that the time of recovery don’t exceed its time limit.

System Models

Use Case Model



Use Case Tables

Use Case ID:	1	
Use Case Name:	Create Account	
Actors:	Main User	
Pre-conditions:	The user can create an account (sign up) and fill some of his personal information: name, age, gender (interests).	
Post-conditions:	The user can sign in with his account by enter email and password , and can create an events.	
Flow of events:	**User Action**	**System Action**
	1- User can sign up to create account by enter his personal information and click submit.	
		2- System saves information of the user to verify this account.
	3- User can sign in by enter his username and password.	
		4- System will show profile page of the user.
	5- User can create an event by click on create event button.	
		6- System will show the event of information.
	7- User will enter the information of this event (name of the event, time, date, place ...) and click save event.	
		8- System will save the event to upcoming events in the list.

Exceptions:	User can't sign up with the email address two times. When the user sign in with wrong password or email the System will request from the user to re-enter the correct username and password .
Includes:	Sign up function includes sign in .
Notes and Issues:	

Use Case ID:	2	
Use Case Name:	Edit Account	
Actors:	Main User -Attendee	
Pre-conditions:	The Main user and the attendee should have an account first.	
Post-conditions:	The attendee can't see the old version of the account after editing the account.	
Flow of events:	**User Action**	**System Action**
	1-The main user will click on "Edit account" button	
		2- System will show list of choices he/she can edit **(name, Birthday, change password)**
	3- main user chooses the choice	
		4- System will open windows of that choice
	5- main user edit the choice then click on "save changes" button	
		6- System will save the new edited version.
Exceptions:	The account doesn't exist	

	Solution: System will show a message “Create an account”.
Includes:	Create Account
Notes and Issues:	The user should first create a personal information for himself first then he can edit the account as he like.

Use Case ID:	3	
Use Case Name:	Create event	
Actors:	Main User	
Pre-conditions:	The person must login as Main user.	
Post-conditions:	Attendees can join the event and also invite their friends to join also.	
Flow of events:	**User Action**	**System Action**
	1- User click in button Create Event	
		2- System will ask the user about the name, the description, and the date for the event.
	3- User invite friends by clicking invite beside each friend’s name	
		4- System will send the request to this name’s account.
	5- Friends will click on button “Join”	

		6- System will add people who joind the event to the list.
Exceptions:	The event has been already created before Soln: Send a notification from the system" this event name is already exists , please try another name" .	
Includes:	none	
Notes and Issues:		

Use Case ID:	4	
Use Case Name:	Edit an event	
Actors:	Main User	
Pre-conditions:	The event should be created first by the main user, then he can edit by changing its date, day, place or privacy settings.	
Post-conditions:	The attendee can't see the old version of data.	
Flow of events:	**User Action**	**System Action**
	1- main user will click on "edit this event" button	
		2- System will show list of choices he/she can edit **(date, day, place or privacy settings.)**
	3- main user chooses the choice	
		4- System will open windows of that choice
	5- main user edit the choice then click on "submit" button	
		6- System will save the new edited version.
Exceptions:	The event isn't exist.	

	Solution: System Sends a message “The event doesn’t exist”.
Includes:	none
Notes and Issues:	

Use Case ID:	5	
Use Case Name:	Manage an event and organize it.	
Actors:	Main User	
Pre-conditions:	The main user should register and login in , the attendee should register in the website , the event should be created at first .	
Post-conditions:	Attendee should be online to confirm requests. Updating stream of attendees.	
Flow of events:	**User Action**	**System Action**
	1- User click in button Invite friends	
		2- System will show to the user his friend list.
	3- User invite friends by clicking invite beside each friend’s name	
		4- System will send the request to this name’s account.
	5- User will search for a name that is not friend with him to invite.	
		6- System will add this name to the user’s list and send request to the name’s account.
	7- User click in button Decline button.	
		8- System will show to the main user all the attendees

	9- User will choose which attendee will be declined by clicking decline beside each name in the list.	
		10- System will remove the declined name from the attendees list.
Exceptions:	- If the User invite friend and he is already invited : 1) Message to the user to till him that he is already invited . - If the User wrote wrong name in the search to invite him . 1) Message to the user to till him that this name is not found . 2) The user will write the name again correctly . 3) System will search to find this account name and invite him .	
Includes:	- Function to add friends in the User Profile . - Function to Display Events .	
Notes and Issues:		

Use Case ID:	6	
Use Case Name:	Delete an event	
Actors:	Main User.	
Pre-conditions:	The creator of the event can remove the event from the system.	
Post-conditions:	System will remove the event and attendees will be removed.	
Flow of events:	**User Action**	**System Action**
	1- User will click Delete event.	
		2- System will remove the event from the list of events.
Exceptions:	the attendees cant delete any event or edit in it	
Includes:	No includes function needed to delete an event.	

Notes and Issues:	

Use Case ID:	7	
Use Case Name:	JoinEvent	
Actors:	Attendee	
Pre-conditions:	First the event should be created.	
Post-conditions:	The attendee will receive notification from that event that he the main user accepted his request and then he can receive notifications from this event.	
Flow of events:	**User Action**	**System Action**
	1-The attendee should click on open button of the event he/she wants to join	
		2- System will display the event
	3- The attendee clicks on "join event"	
		4- System will add the attendee
	5- The main user will accept the attendee's request.	
		6- System will send notifications to this attendee.
Exceptions:	The event doesn't exist. Solution: System will send message "The requested event doesn't exist, please try again".	
Includes:	None	

Notes and Issues:	The main user can accept or refuse the attendee's request to join the event.

Use Case ID:	8	
Use Case Name:	Buy Tickets	
Actors:	Attendee	
Pre-conditions:	Attendee must join the event first before he can buy a ticket , and also he must has an account and login.	
Post-conditions:	System will register the attendee's name and add them in the list.	
Flow of events:	**User Action**	**System Action**
	1- User will click on the desired event.	
		2- System will display the event.
	3- User will click on button" Buy Ticket"	
		4- System will take the name, phone number, and the credit card number of the attendee.
Exceptions:	The tickets are sold out .or the amount of money isn't enough Soln: send a notification that tickets ar sold out Or , send a message " please recharge your account".	
Includes:	none	
Notes and Issues:		

Use Case ID:	9	
Use Case Name:	Leave an event.	
Actors:	Attendee	
Pre-conditions:	First the event should be created and the attendee joined it, then he can leave it.	
Post-conditions:	The attendee can't have notification from that event anymore.	
Flow of events:	**User Action**	**System Action**
	1-attendee should click on open button of the event he/she wants to leave	
		2- System will open the event
	3- attendee clicks on "leave this event"	
		4- System will make the attendee leave the event
		6- System will edit the choice
Exceptions:	The event has been already happened. Soln: System prints a message"The event is already happened.".	
Includes:	Join event function	
Notes and Issues:		

Use Case ID:	10

<table>
<tr><td>Use Case Name:</td><td colspan="2">Create streams to add different events.</td></tr>
<tr><td>Actors:</td><td colspan="2">Main User + Attendee = Users</td></tr>
<tr><td>Pre-conditions:</td><td colspan="2">Users should be online in their accounts; they should be joined or created three or more events.</td></tr>
<tr><td>Post-conditions:</td><td colspan="2">The user will have list with his upcoming and past events saved in his profile .</td></tr>
<tr><td rowspan="6">Flow of events:</td><td>User Action</td><td>System Action</td></tr>
<tr><td>1- User will click on Create My Stream Button .</td><td></td></tr>
<tr><td></td><td>2- System will show all the events that he joined or created.</td></tr>
<tr><td>3- User organize events according to his preferences in an array (index will represent the preference).</td><td></td></tr>
<tr><td></td><td>4- System will save this order.</td></tr>
<tr><td>and so on</td><td></td></tr>
<tr><td>Exceptions:</td><td colspan="2">- If the user have two events in the same time.
1) The system will show message to the user to till him to choose one of these two events to attend or create.
2) The user will choose the event.
3) System will save these changes.</td></tr>
<tr><td>Includes:</td><td colspan="2">Function to Display events.</td></tr>
<tr><td>Notes and Issues:</td><td colspan="2"></td></tr>
</table>

Use Case ID:	11
Use Case Name:	Follow different users.

Actors:	Main User – other users.	
Pre-conditions:	User must be online to follow others.	
Post-conditions:	User can see different users' joined events.	
Flow of events:	**User Action**	**System Action**
	1- User will choose the friend that he wants to follow him .	
		2- System will send a notification to the follower to accept or reject the follow.
Exceptions:	The desired user to follow doesn't exist. Soln: System will send notification " the user doesn't exist , try again".	
Includes:	none	
Notes and Issues:		

Use Case ID:	12	
Use Case Name:	Receive Notifications	
Actors:	Main user - Attendee	
Pre-conditions:	The main user should at first accept the attendee's request to join the event and the attendee join the event.	
Post-conditions:	The main user and the attendee view the event's notifications.	
Flow of events:	**User Action**	**System Action**
	1-The main user or the attendee writes something on the event	
		2- The system sends notifications to

		the main user and the attendee
	3-The main user and the attendee view the notification	
		4- The system changes the notification to read notification.
Exceptions:	The main user didn't accept the attendee's request for joining the event. Solution: The system won't send any of the event's notifications to this requested attendee to the event.	
Includes:	Join Event.	
Notes and Issues:	The attendee should at first join the event to be able to receive notifications from the event and after the event ends or the main user deletes the event, the system won't send notifications from this event to the main user and the attendee anymore.	

Use Case ID:	13	
Use Case Name:	Suggest Event	
Actors:	Attendee - Main User	
Pre-conditions:	Attendee and Main user must log in to the system.	
Post-conditions:	Other people join to the specific event.	
Flow of events:	**User Action**	**System Action**
	1- User will click to the desired event to be suggested.	
		2- System will display the event.
	3- User suggest that event to another people to join	

		4- System will send a notification with that event to display to the other users.
Exceptions:	The suggested Event isn't exist Soln: Send a notification that the suggested event isn't exist and try to write the name of the event correctly.	
Includes:	none	
Notes and Issues:		

Use Case ID:	14	
Use Case Name:	Search for different events	
Actors:	Attendee	
Pre-conditions:	Attendee should have an account. Then he can Search for different events of specific category (criteria).	
Post-conditions:	Attendee can know anything about the event he/she chose also can join it	
Flow of events:	**User Action**	**System Action**
	1- Attendee writes the category of event he/she wants	
		2- System will show all the events on this category
	3- Attendee chooses the event he/she wants	
		4- System will open the event.
Exceptions:	The Event doesn't exist. Soln: The system prints a message " The requested event doesn't exist ,	

	please try again).
Includes:	none
Notes and Issues:	

Use Case ID:	15	
Use Case Name:	Add photos or videos about attended events .	
Actors:	Attendee - Main User	
Pre-conditions:	Having files with pictures in the PC .	
Post-conditions:	Saving the pictures in albums later on .	
Flow of events:	**User Action**	**System Action**
	1- User will click in add photo button.	
		2- System will ask if the user will add photo in his profile or in the event.
	3- User will choose.	
		4- System will ask for photo file path
	5- User will choose the file of the photo.	
		6- System will load the picture from the PC.
Exceptions:	- If the size of the picture is more than 5 megapixels. 1) Message to till the user or the attendee that the photo is too huge please chose another photo. 2) The user will chose another photo to be loaded.	

	The system will load the picture.
Includes:	None
Notes and Issues:	

Ownership Report

Item	Owners
-The Functional Requirements (Edit Account , join Event , Receive Notifications). - The non-Functional Requirements (Reliability, Supportability, Interface, Availability, Recovery). -Software Purpose. -Definitions, acronyms and abbreviations.	Malak Medhat
-The Functional Requirements (Create account , Delete Event , Follow user). -Use Case Model. - Software Scope	Mostafa Nasser
-The Functional Requirements (Create event , buy tickets , suggest events). - The non-Functional Requirements (Implementation). -Use Case Model. -Definitions, acronyms and abbreviations.	Nouran Atef
- The Functional Requirements (Edit event ,Leave Event , Search event). -The non-Functional requirement (Legal, Maintainability).	Yussuf Mohamed

- Document purpose and audience. - Software Scope.	
-The Functional Requirements (manage event, create stream , add photos/videos). - The non-Functional Requirements (Usability, Performance). -Use Case Model. -Software Scope.	Reem Sheeha