Abdelrahman gamal	20120225
Abdelrahman maher	20120229
Abdelrahman mohamed ibrahim	20120230
Abdelrahman mohamed mustafa	20120233

BumpersRequirementsAnalysis Document (RAD)

Document Purpose and Audience

- The introduction of the Software Requirements Specification (SRS) provides an overview of the entire SRS with purpose, scope, function Requirements ,NonfunctionRequirements and overview of the SRS. The aim of this document analyze and give an in-depth insight of the complete **API** by defining the problem statement in detail
- **the audience to read this document client , project manager , system analysis and system designer**

1. Introduction

1.1 Purpose of the system

This document purpose is to make a preview for our new API system which will be a social network, showing the goal of the project, functions that the program will be able to do , the design and how will the program will look like , who will be our audience .

Note: some of our functions may be modified while developing the system.

1.2 Scope of the system

Our project will be an API for a social network .this API will be a simple social network which can be developed later to service a small community like a school.

Our system will be as follow Users which is the main element in this system and it can make a group , page ,post ,like ,massage ,hashtag or adding another user and has many attributes like name ,age ,email ….etc. users can be normal or premier . Groups are one of what user can make and it can be private or public and managed by many users. Users create fan pages which managed by one or many users and it’s attribute is the number of likes. Users make like to the posts they

admire. User make hashtag# to organize a specific content of posts. Users send massages to other users in their friend list. Users send friend add request to other users to add them in their friend list.

2.1. Functional Requirements

FUNICTION REQUIRMENTS

2.1.1. USER :

Anyone can be a user by sing up write his name and password if the name was exist before program ask him to enter again

2.1.2.GROUP :

Group is a set of users collected together can share their needs in, group can be secret no one see posts accept them and could be not secret (public && private group) group contain 2 types of users

admin user and ordinary user

2.1.3.Post :

user can write what he want , post picture and share photos and

share it with his friends and they show it and can comment on it and do like

2.1.4.Message :

Message is a private text from user to another no one can see it except them

2.1.5.Hashtag :

Is and ordinary post but contain '#' and must contain '_' that make it shared with any one write this hashtag also can share a specific subject using it

2.2. Nonfunctional Requirements

- 2.2.1. Performance
- *Short response time*. The loading time of the API software must be smaller than 30 seconds
- 2.2.2. Reliability

- ***Crash safe***. The API software should be crash safe in 85 % of its runtime.

2.2.3. Usability

- Simple to use . software must be simple as possible that make user treat with it so easy

2.2.4. Supportability

- Users people who use API
- Login API must support login
- Sign up API must support login if user do not have account can make account and then can log in
- Posts , page , group, hashtag
- Also must support like posts , comment on posts ,like page , join groups

  And using hashtag

2.2.5. Implementation

- ***Programming language***. Bumpers must be implemented in Java.

2.2.6. Interface

- ***Simple user interface***. The user interface of API should be understandable tointerface . The user interface is based on a main window, which includes a text which write E_mailor username and button for login and other for sign up

2.2.7. Legal

- ***Cheap API*** . The development of API must be done with a budget <= 1000 $. As the software should reach as many players as possible and it should be financed over advertisement, it cannot be expensive

2.2.8. Packaging

- The API developer must be available over the web within one download. All needed files

Use case

Use Case ID:	1	
Use Case Name:	User	
Actor:	System	
Pre-conditions:	User sign up, sign in	
Post-conditions:	User already exist so sign in or new so register to create new user	
Flow of events:	**User Action**	**System interaction**
	Sign in	Check on name and password if correct sign in else ask user to resign in or register
	Register at it by enter his data name ,pass ,other ordered information	Check on name and pass and create account
Exceptions:	None	
Includes:	None	
Notes and Issues:	None	

Use Case ID:	2	
Use Case Name:	Group	
Actor:	System	
Pre-conditions:	User Create Group	
Post-conditions:	Group be Available for users	
Flow of events:	**User Action**	**System interaction**
	User create group	Ask user for name, secret or public and description and some other info
	User enter all data ordered by system	System check all data and create group
Exceptions:	none	
Includes:	None	
Notes and Issues:	This case owner for case 1 who create it	

Use Case ID:	3	
Use Case Name:	Post	
Actor:	System	
Pre-conditions:	User write post	
Post-conditions:	The Post arise and be available to anyone who has the authority to see it	
Flow of events:	**User Action**	**System interaction**
	User write post and share it	
		System make it available to anyone who has the authority to see it
		If any one make like or comment appear on the post
Exceptions:	None	
Includes:	None	
Notes and Issues:	None	

Use Case ID:	4	
Use Case Name:	Message	
Actosr:	system	
Pre-conditions:	User write message and send it to other user	
Post-conditions:	Gets the message to the other user	
Flow of events:	**User Action**	**System interaction**
	User write message and select other user to send this message to it	
		Send message and let the other user see it
Exceptions:	None	
Includes:	None	
Notes and Issues:	No one can see message Except them	

Use Case ID:	5	
Use Case Name:	Hash Tag	
Actors:	System	
Pre-conditions:	User write post and write mark of hash tag '#'	
Post-conditions:	Post add to the hash tag written	
Flow of events:	**User Action**	**System interaction**
	User write post and write hash tag	
		Check if the hash tag exist if yes add post to it
		If no create hash tag then add post to it
Exceptions:	None	
Includes:	None	
Notes and Issues:	None	