W-eXpert

FREELANCERS

INITIAL DESIGN REPORT

1394659 Serhat ALYURT

1395508 Ahmet Kutlu ŞAHİN

1448869 Caner KAVAKOĞLU

1448851 Yağız KARGIN

TABLE OF CONTENTS

1.0 INTRODUCTION

Throughout the analysis phase of our senior project development cycle, we examined what are the functional and non-functional requirements of W-eXpert. In addition, we have searched the required technologies we will use. After this period, we studied to design of the W-eXpert and initial design report was created.

While studying to prepare this report, we saw what we will face and deal with. Also preparing the diagrams has drew us a solid picture of our project. They helped us to find answers to many of the questions in our minds. With the help of this and final design report the implementation of the project will be much more easier. Because the process of implementation will be systematic and easy to separate according design concepts.

In order to take advantages of structural design we have used detailed Data Flow Diagrams, Class Diagrams, State Transition Diagrams, Sequence Diagrams which are expressed in the following pages.

1.1Project Title

Our project is called *W-eXpert*.

1.2 Problem Definition

In this information era where we are living today, reaching information is getting more and more important. In this point, getting people who demand and supply information together, is the main problem. Our aim is to take humanity one step forward to the future through our philosophy of information sharing in this common platform. Even more we will take this service and insert it into a social environment.

Past life is nostalgia for people. Especially finding old friends, teachers, family who had lost contact with you, surprises you and brings back memories. To contact with these people, learning their conditions and their life is something charming. Even sharing life makes it unavoidable to join into this environment.

Everybody is fed up from lots of pages in result of search engines. We read and read lots of documents. Most searches will probably come out with new searches even about unrelated topics. Time is everything and getting information from a search engine will spend our precious time with garbage information. Forums are also something people do not one to use because of detailed registration. Sometimes we really get lost in web pages reading to find our answer. There is no database or artificial intelligence that can give our specific answer other than human mind/knowledge. On the other hand, there will be questions which require human experience to be answered appropriately. Unless one to one conversation with a person is managed, we cannot find our answers with most details.

With the requirements analysis report new problems aroused with our design. From our previous observations there is only one solution to getting the exact answer to a question is getting in contact with some experienced person. We have to lure these people to use services. Why will people sit by a computer and try to help people? Nowadays nobody is doing something without a gain. So we will provide this gain. Reputation, especially in a crowded environment is everything! If you are known by others, you gain respect, acceptance into certain groups, better job opportunities, better social platform etc. These are valuable gains everybody surely wants. With these in our hands we will surely encourage people to join and do their best and get known.

1.3 Project Scope and Goals

Our goal in this project is to find solution to the problems defined above. We will create a social environment with a human-to-human information delivering system, which saves time.

This will be an application that will come to your rescue in times when you seek for an expert that you may ask any information which you can’t reach through search engines. On the

other hand you can create your social environment or join an existing one. We will provide written communication by using technologies such as Jabber, Ajax, XMPP.

1.4 Design Objectives

In this part, we examine the main design objectives on the privacy,availability, reliability and usability perspectives.

Privacy

Like Facebook, our social platform will ensure users privacy. If they want they can hide their profile from anyone they want. Expert privacy is another topic for this section; they should not be distributed too much with users and their questions if they do not want. They must feel free to answer/spend time. Their privacy is more important than other users, so if they want they can use nicknames other than their profile names. If they do not want their names/information will not be shown in dictionary or instant message service. There will be only friend-to-friend instant messaging and expert-to-friend instant messaging, so people unknown to you cannot disturb you.

Availability

Our project's target is a huge population, so we will make this project reachable by everyone. Our only requirement for client side will be Java Runtime Environment and a mail address.

Reliability

Our question answering system will be reliable (especially in future). We make the question answering system as people can rate experts. Another rating system will be held by our system (such as response time). In a huge populated environment a natural selection will be applied over

users who want to be experts. So the experts' rating and their information, which is kept in our database, will be criteria for people to choose their experts. With this option we lead experts to be gentle and trustful to others. We will also create a triangle form for the experts according to their ratings and experience within the system. To walk up in the triangle form, experts will want to behave polite.

Usability

Our project will be a user-friendly service. First of all everything must be as simple as it can be. People do not deal with hardly understandable options. Search, ask, invite, create a group etc. should be very easy to comprehend.

2.0 Design Constraints

In this part of the report we examine what the constraints affect us and discuss them into three subtitle, namely, time constraints, performance constraints and API constraints.

2.1 Time constraints

As all other groups, we too have really little time to finish this project. Gantt Chart of the project is given in the appendix. As you can see in Gantt Chart everybody has lots of work to do and time is really limited to do these. Even if we have strictly stick with the schedule, there will be still lots of things to do. Because our project topic is really expandable and there will be always new things to be added. However we think in the remaining 5-6 months we will achieve the main goals of the project and produce every thing we have mentioned in these reports.. So every team member agreed on "We will really continuously improve our project by every means even after graduation".

2.2 Performance Constraints

As a social platform we'll have lots of members. So managing all these people will require good servers and databases. Now we have limited hardware for servers and database for storage. In the end of the project we will provide users with document share opportunity and a huge dictionary. Guess the big picture, millions of people, sharing lots of documents (now limited around 15MB) and everybody has lots of entries in dictionary. We have divided our database into three for this purpose, if we can have three good servers running on three good machines with (really) huge storage area, we can improve our performance.

2.3 API Constraints

Our project hopefully support application programing interface. However due time limitations and workload of the whole project we can only provide this for certain languages. In the end of second term our API service will provide only PHP support.

3.0 Process

In this part of report we will show the team organization and the Process Model that we will use in project. And also the tools we will use are included in this part of the report.

3.1 Team Organization

In our project group, Serhat ALYURT, Ahmet Kutlu ŞAHİN, Yağız KARGIN, Caner KAVAKOĞLU, we will adopt in Democratic Decentralized principle. For the task distribution and scheduling, see Gantt Chart at Appendix 9.1 and 9.2

3.2 Process Model

In this project we are using Spiral Process model. Our topic is very wide and as new modules are added, model will be revised and it will be easy to proceed.

3.3 Software Requirements

To implement our project we will use certain tools:

- Dreamweaver: For the design of the GUI.
- MySql: To establish and maintain databases.
- Eclipse: JSP Tool.
- TomCat Apache Server: Server.
- SVN : for revision of the project
- Jabber Server: for instant messaging

3.4 Hardware Requirements

Our Project will need following hardwares:

- Server Computer
- Internet connection (by any means)

4.0 Architectural Designs

In this part we will describe Graphical User Interface and Data Flow diagrams are shown and described. Also a dictionary for the elements of DFD's and the State Transition Diagram is provided in here

4.1 Class Diagrams

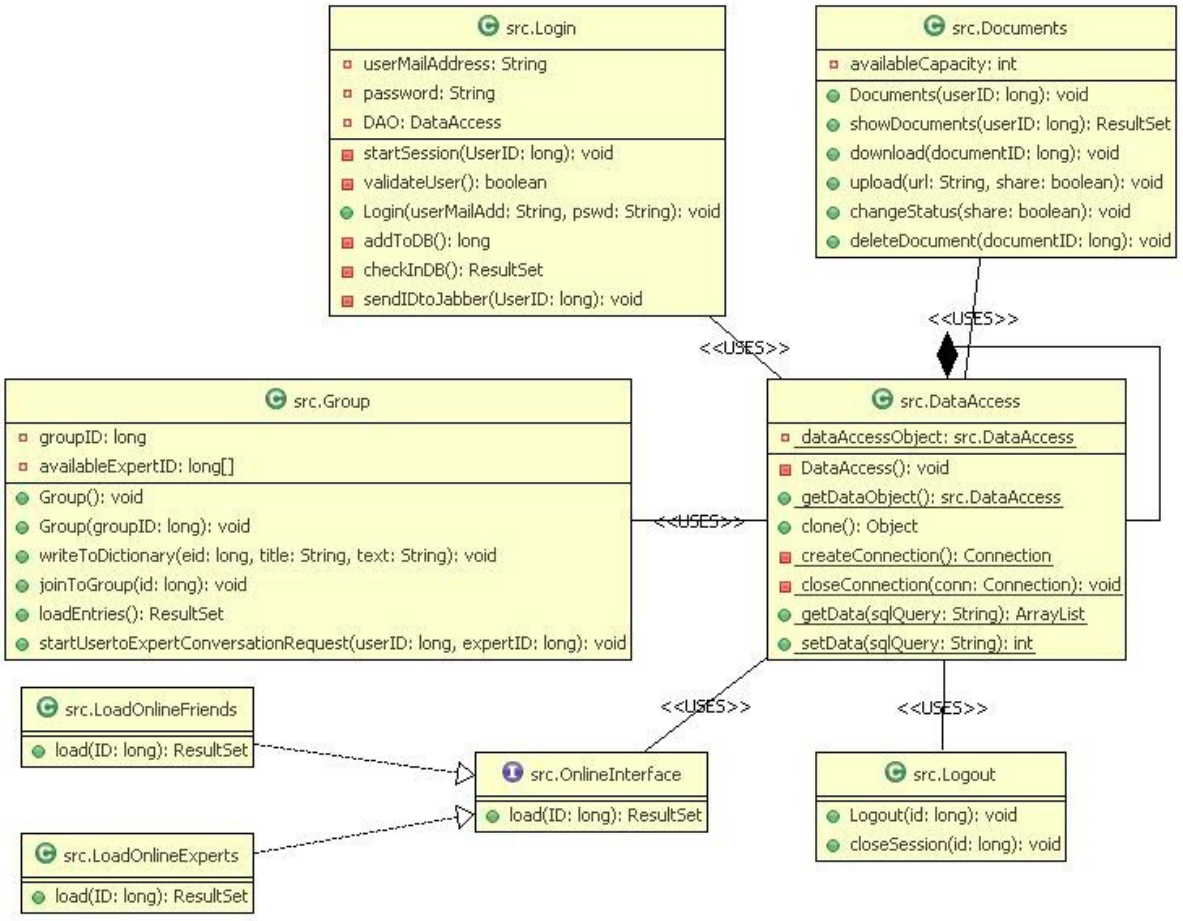
Classes of our project that we will use with Java bean in the Java Server Pages are given and described in this section.

Syntax used to describe class diagrams:

Symbol	It symbolizes
	Java class
	Java Interface
	Private Attribute
	Private Method
	Public Method
	Public Attribute

- Underlined texts means static.
- Italic text means Abstract

4.1.1 General Module



DataAccess:

Data transfers between the server and databases are the most frequent precesses on the system. Considering this fact, to maintain the efficiency we used singleton design pattern mentioned in GoF(Gang of Fours).

In all procedures which want to access databases, use the unique instance(DAO) of this class.

Login:

When the user enters system for the first time, a login instance i constructed. This instance gets the mail address and password from the user and connects to database to check if this account related with this mail address is exists. If the account exists, the profile of the user is loaded to the main page. If not, a default profile is created and loaded to the user database. Finally the user id is passed to the Jabber server and the main page is loaded. See Sequence diagram.....

Documents:

When a user wants to do any file operation (i.e. upload,download, share..) an instance of this class is created. And its methods is used. See Sequence diagram ...

Group:

When a user wants to do any group related operation (i.e. Create group, join a group, writing to dictionary as a high level expert, starting a conversation with an expert ..) an instance of this class is created. And its methods is used. See Sequence diagram ...

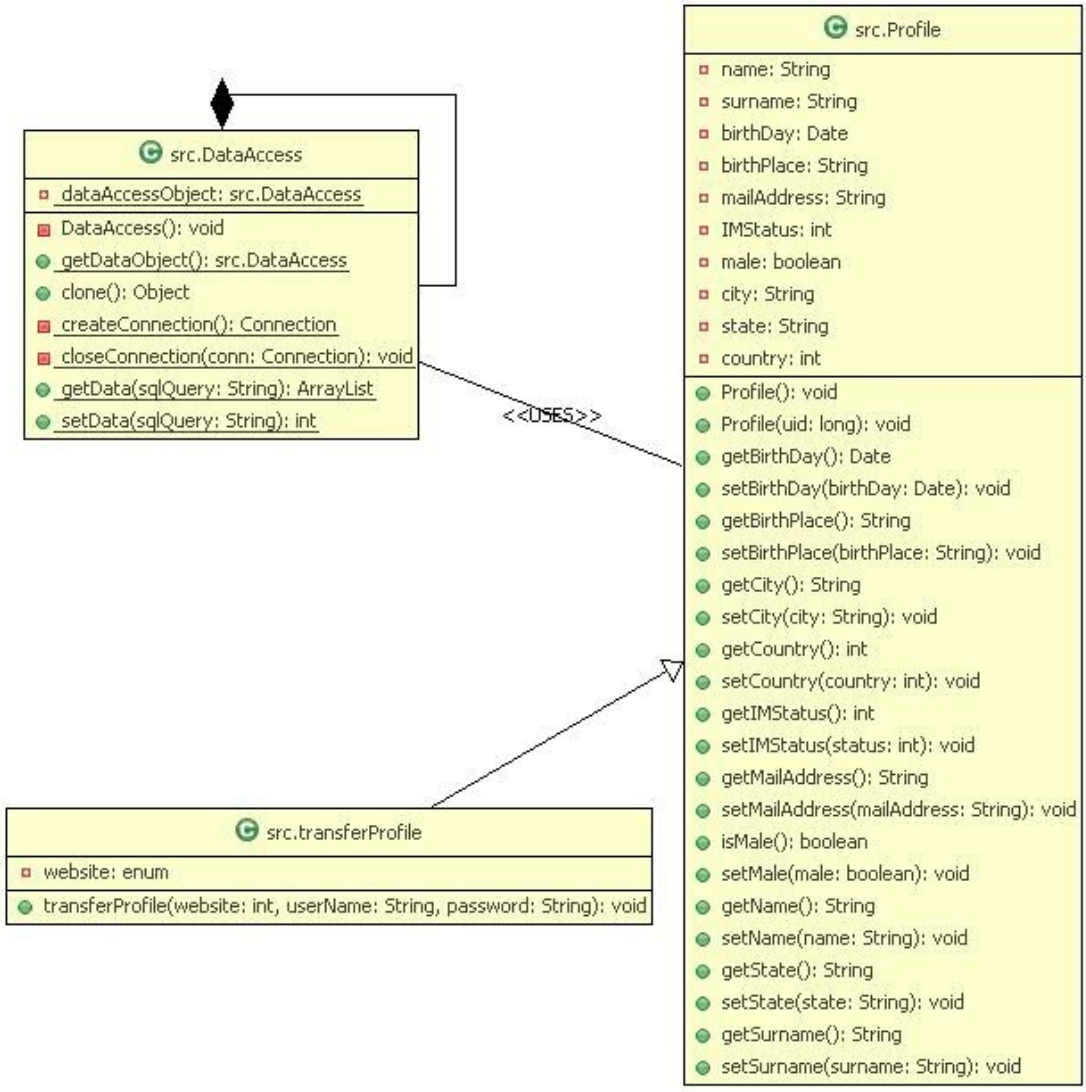
Online friends:

This instance and its method is used to get the list of the online friends of the user in order to show them in the left bottom corner of the page.

Online experts:

This instance and its method is used to get the list of the available experts which is displayed in a page of a group.

4.1.2 Profile Module



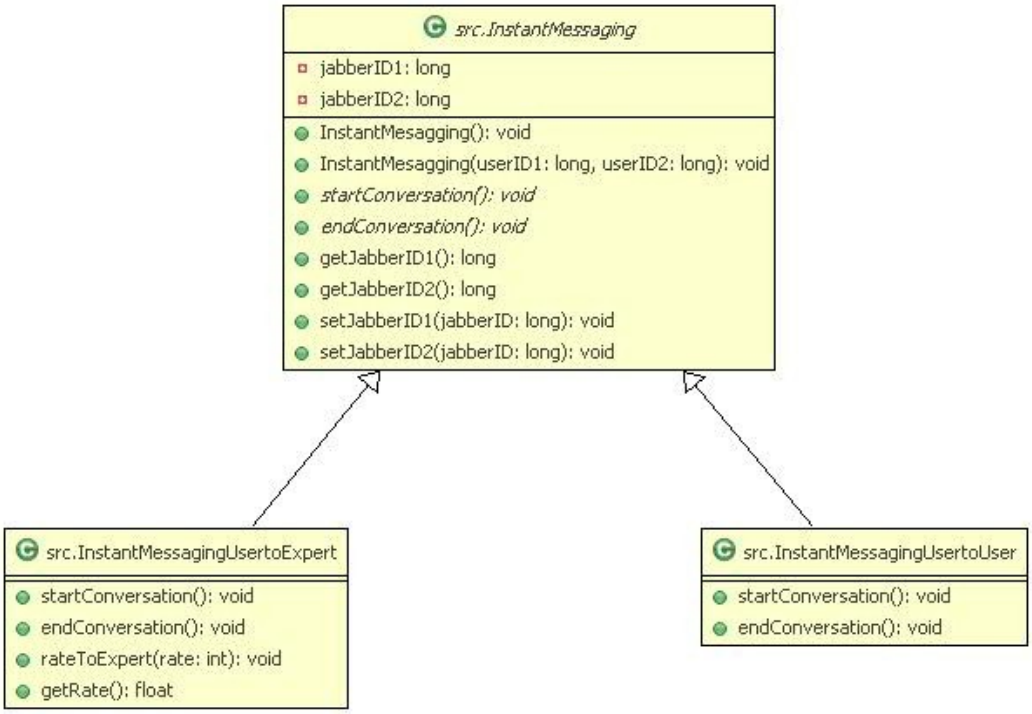
Profile:

It is the class which is apparently used to load and display the user profile data.

TransferProfile:

In our system there is an option to transfer data of the user from other certain websites to our system. To do that an instance of transferProfile is created.

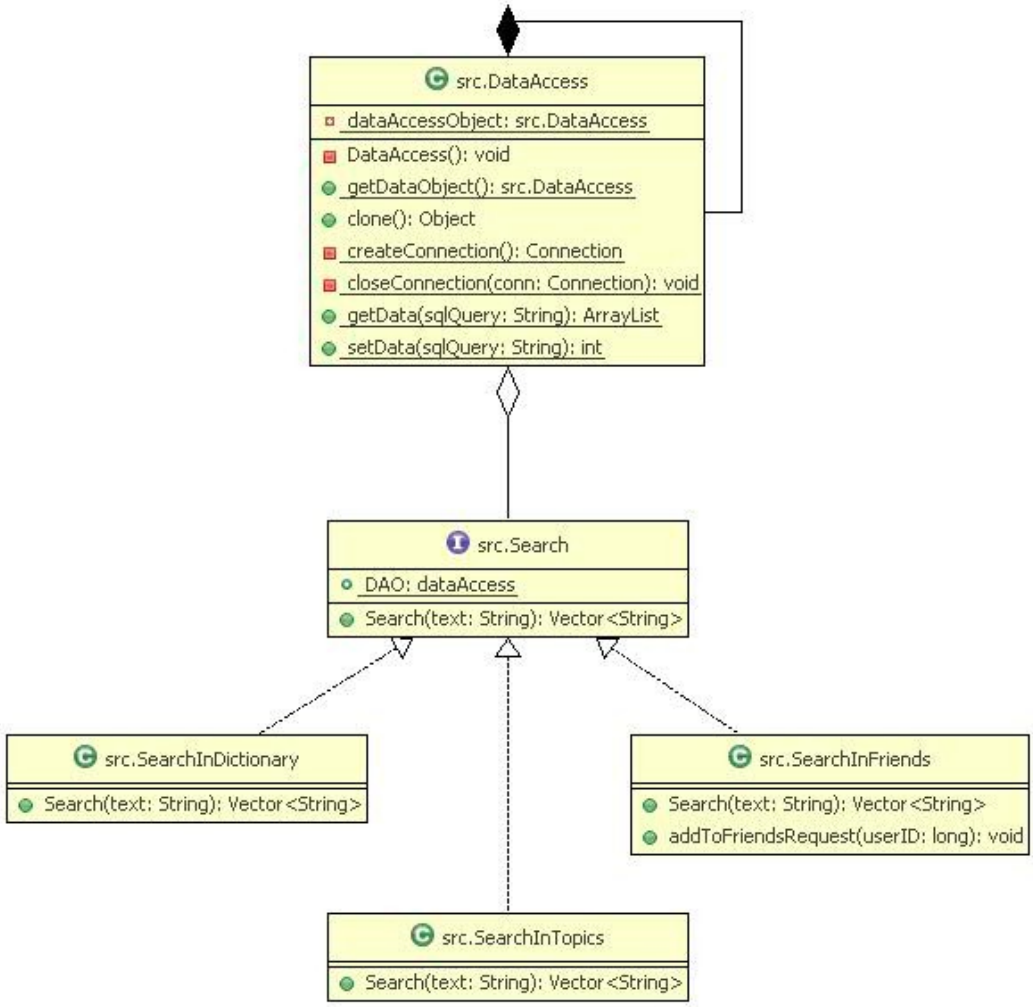
4.1.3 Instant Messaging Module



Instant Messagging:

Instant messaging is used for two aims , namely; conversation with friends and discussion with an expert. To construct an intant messaging the contructor of the instance gets the user id of two user and passes these ids to Jabber that will make a connection to start a conversation.

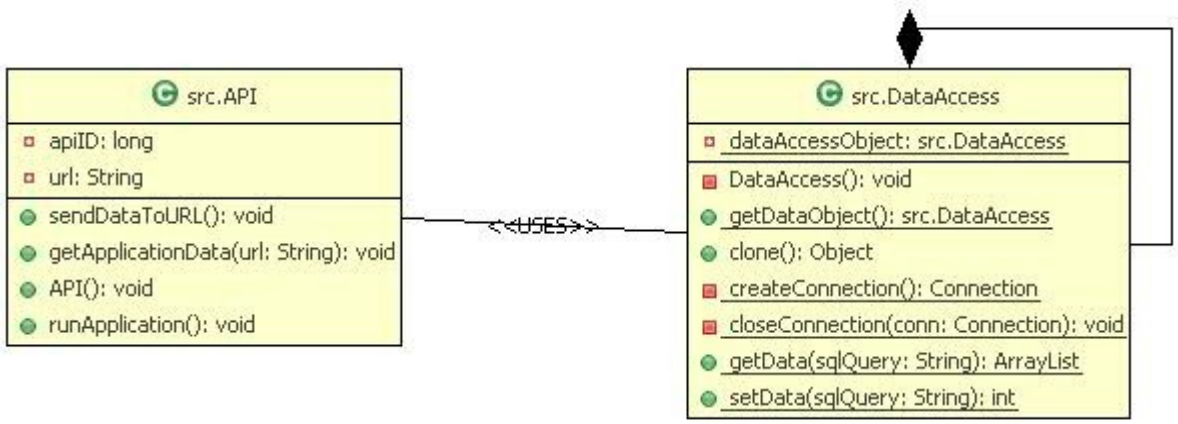
4.1.4 Search Module



Search:

There are three instance of searching in our system. These are used to make a search in dictionary, topics and friends(among all users).

4.1.5 API Module



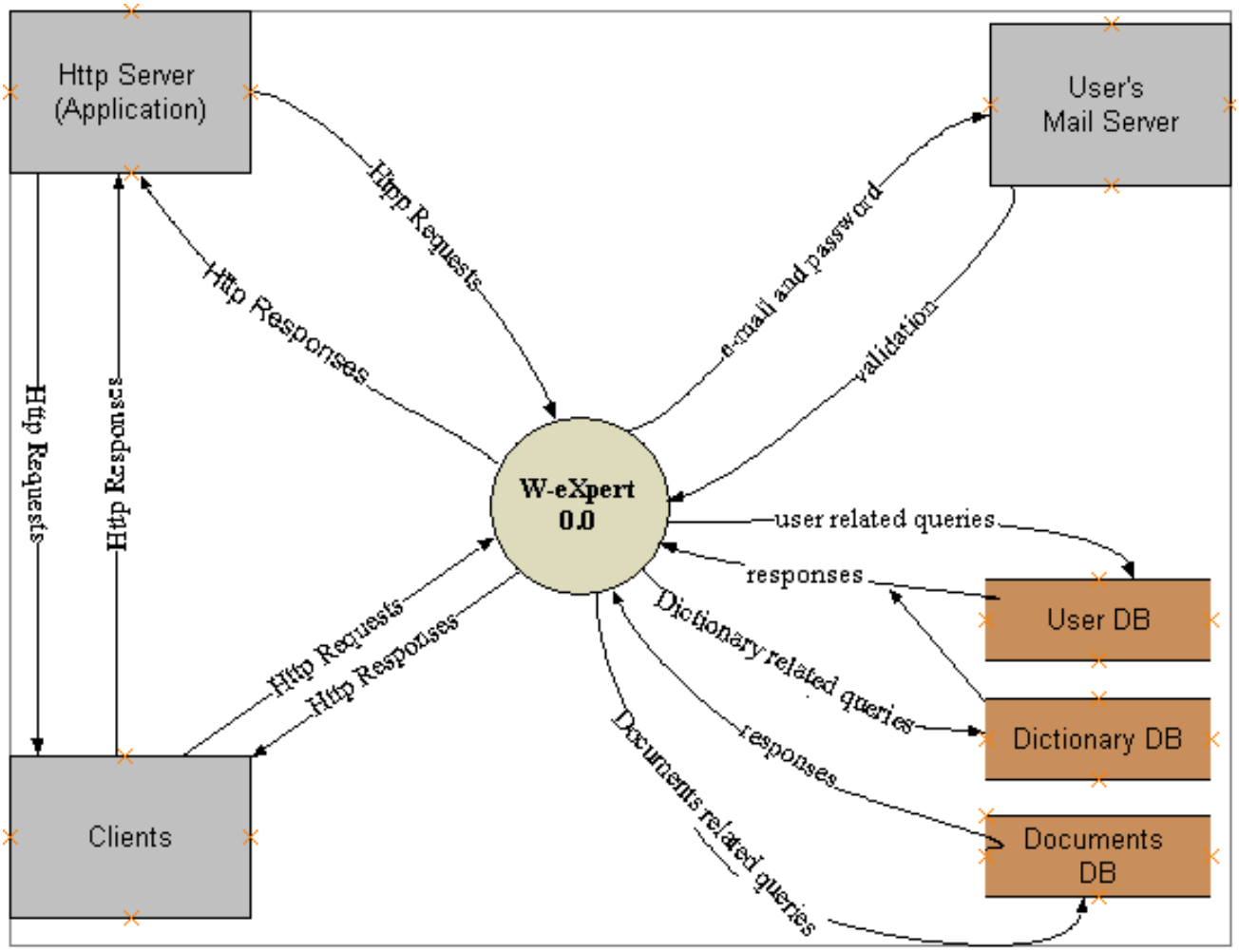
API (Application Interface):

Developers outside of our system is able add applications to expend our system. These application will be loaded from an external server. This external server can get data from our databases and use these data for its our application. This API instance gets the application with its method from an URL, and runs it in our system.

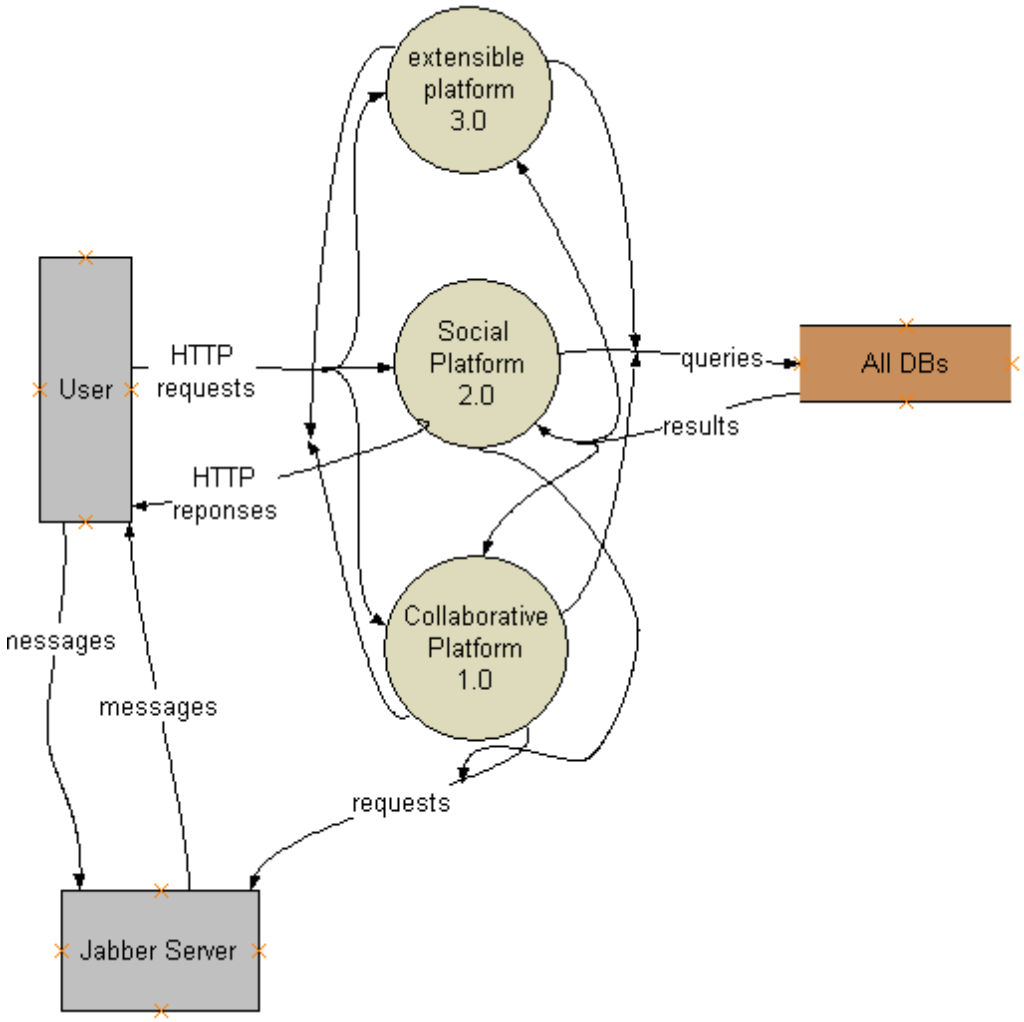
4.2 DFD

In this part we will look at Data Flow Diagrams of our project. We have already given these diagrams in Requirements Analysis Report and we haven't change it. At the end of this part a data dictionary is added for description.

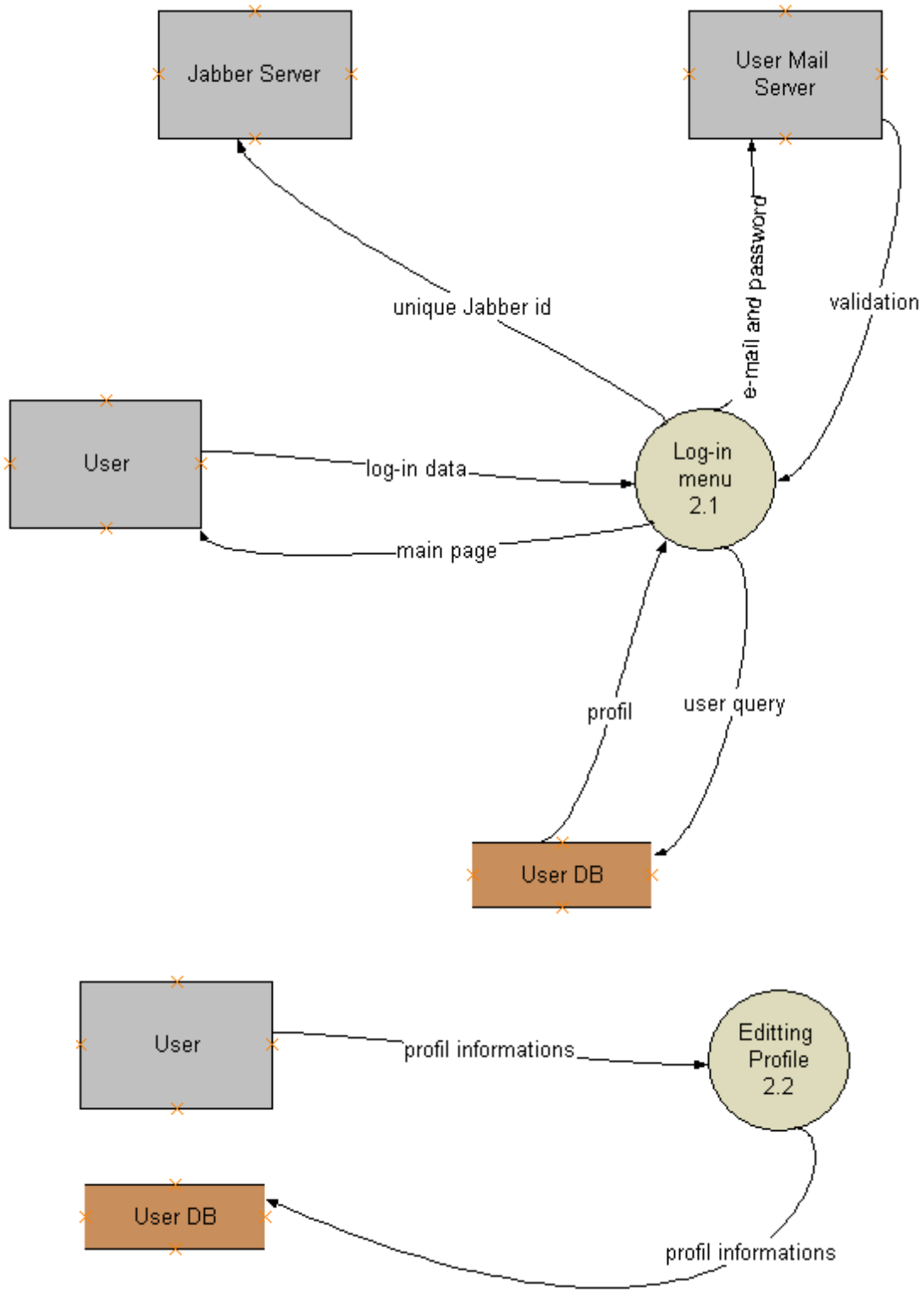
4.2.1 Level 0

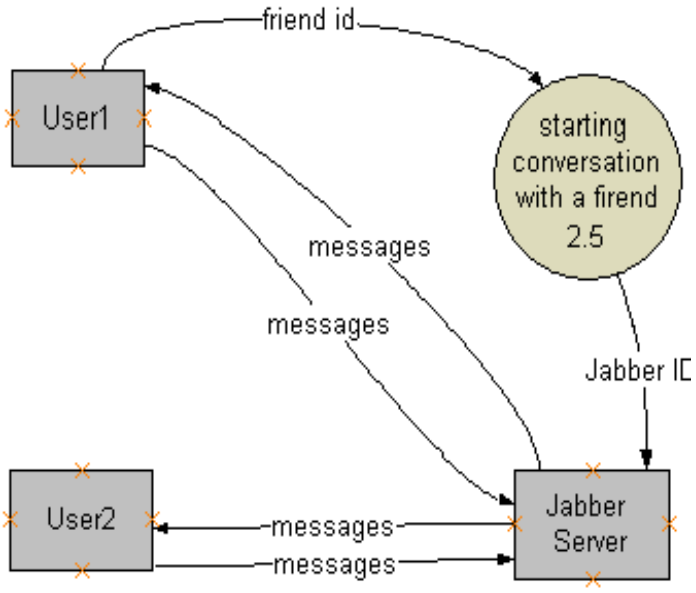

Http Server (Application)
User's Mail Server
Clients
W-eXpert 0.0
User DB
Dictionary DB
Documents DB
Http Requests
Http Responses
e-mail and password
validation
Http Requests
Http Responses
Http Requests
Http Responses
user related queries
responses
Dictionary related queries
responses
Documents related queries

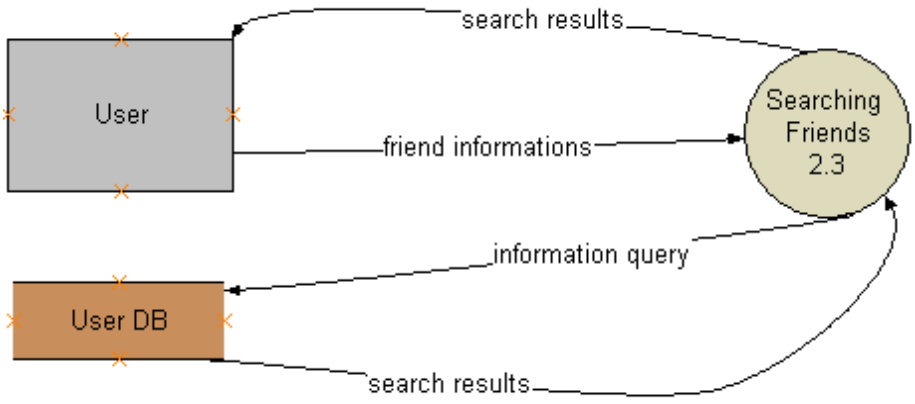
4.2.2 Level 1

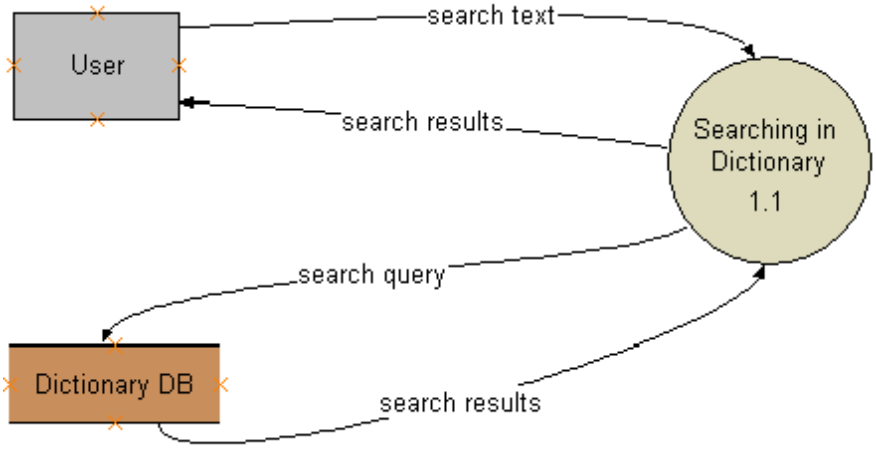

extensible platform 3.0
Social Platform 2.0
Collaborative Platform 1.0
User
HTTP requests
HTTP reponses
queries
results
All DBs
nessages
messages
requests
Jabber Server

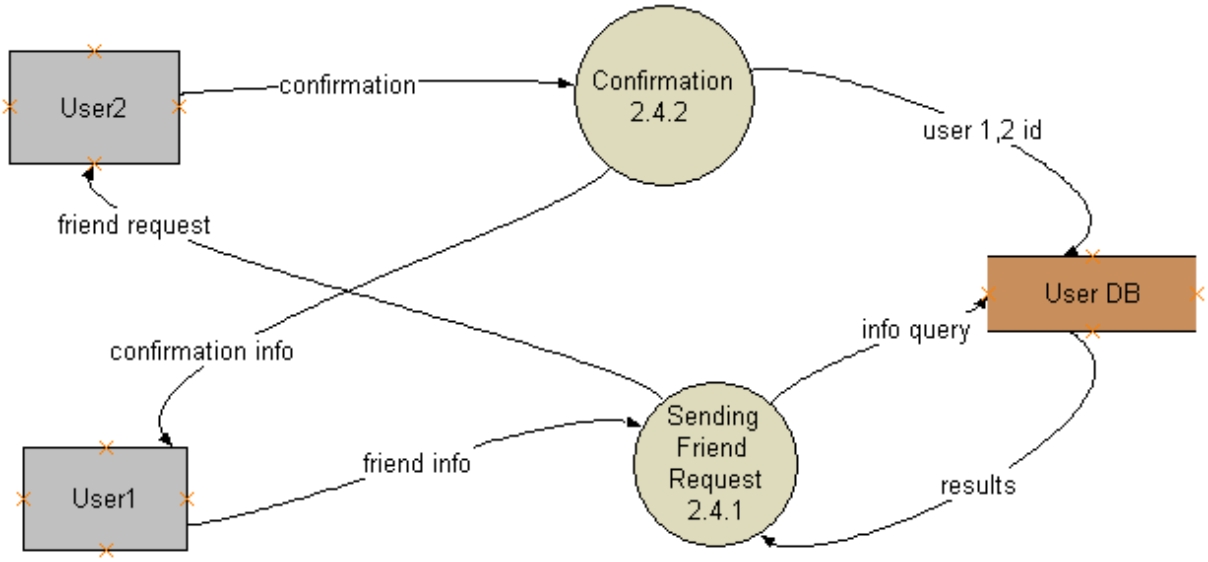
4.2.3 Level 2

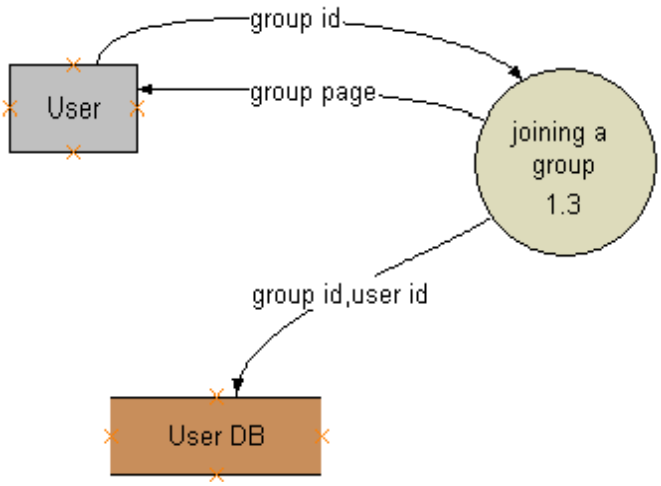

Jabber Server
User Mail Server
unique Jabber id
e-mail and password
validation
User
log-in data
Log-in menu 2.1
main page
profil
user query
User DB
User
profil informations
Editting Profile 2.2
User DB
profil informations

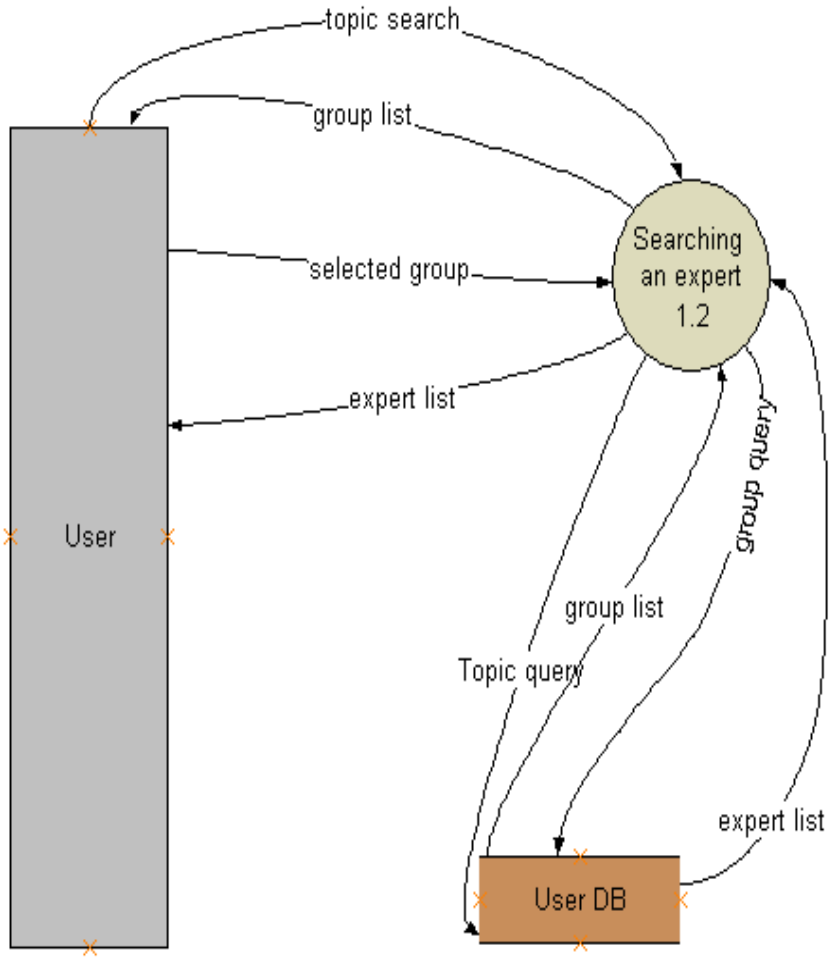

friend id
User1
starting conversation with a firend 2.5
messages
messages
Jabber ID
User2
messages
messages
Jabber Server

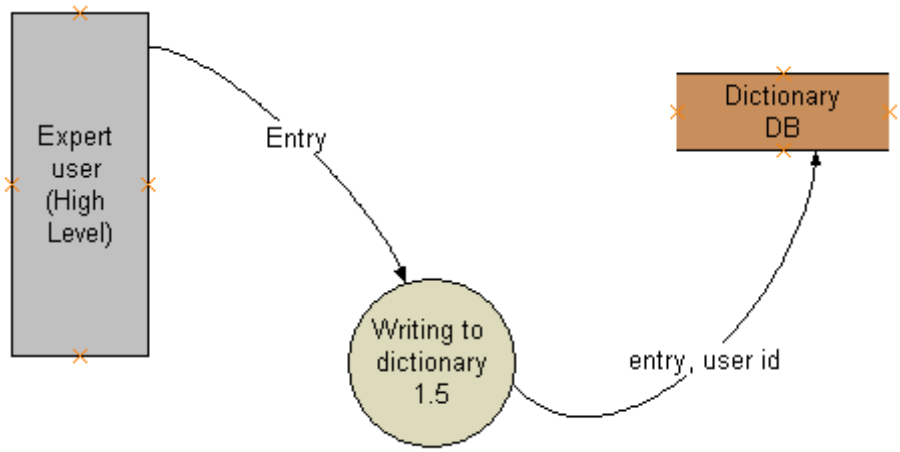

search results
User
Searching Friends 2.3
friend informations
information query
User DB
search results

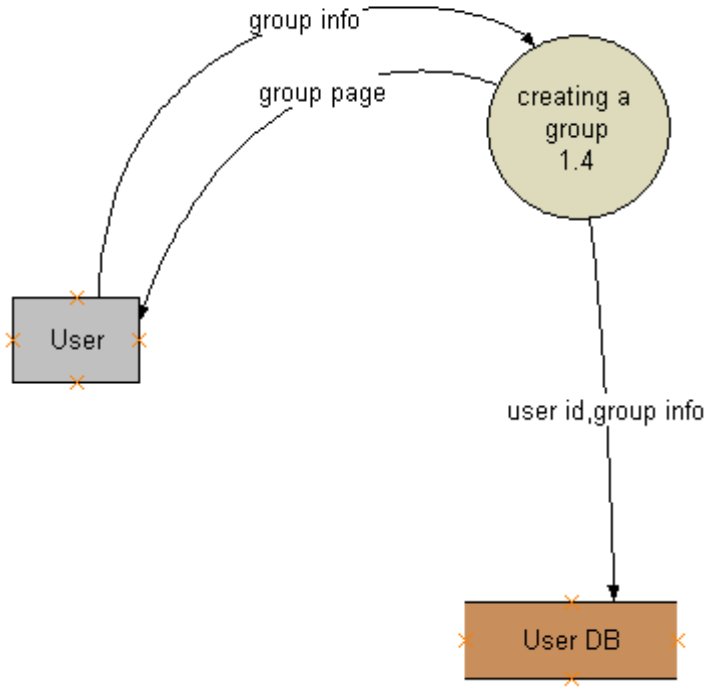

User
search text
search results
Searching in Dictionary 1.1
search query
Dictionary DB
search results

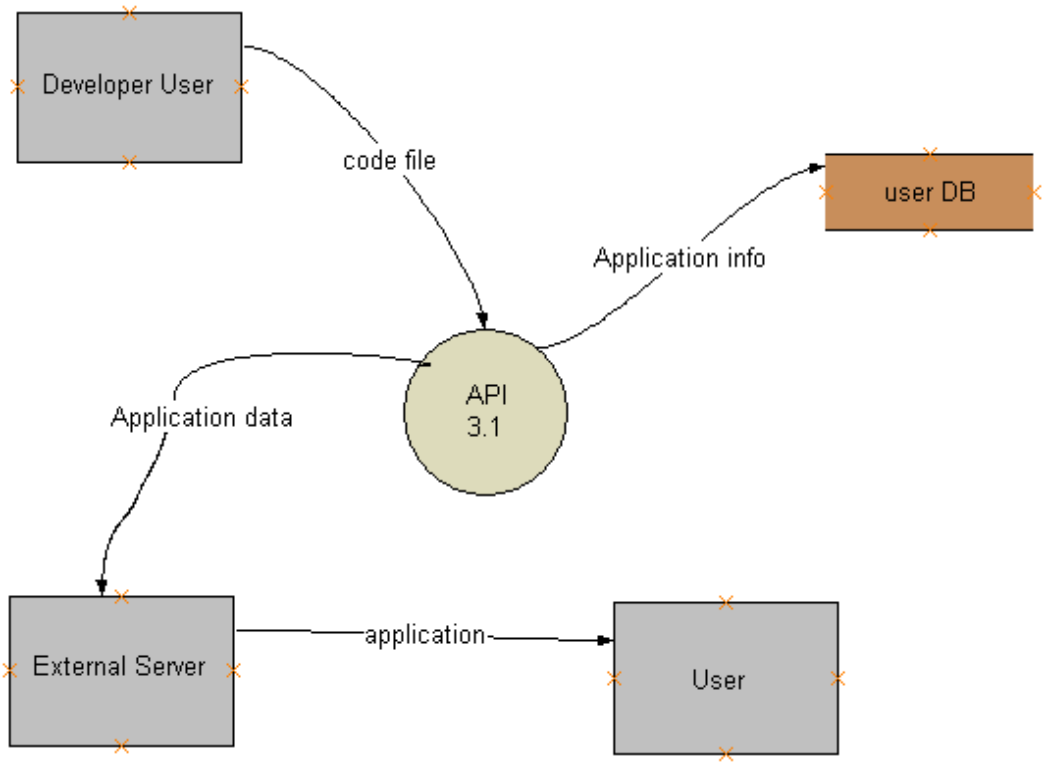

User2
confirmation
Confirmation 2.4.2
user 1,2 id
friend request
User DB
info query
confirmation info
Sending Friend Request 2.4.1
User1
friend info
results

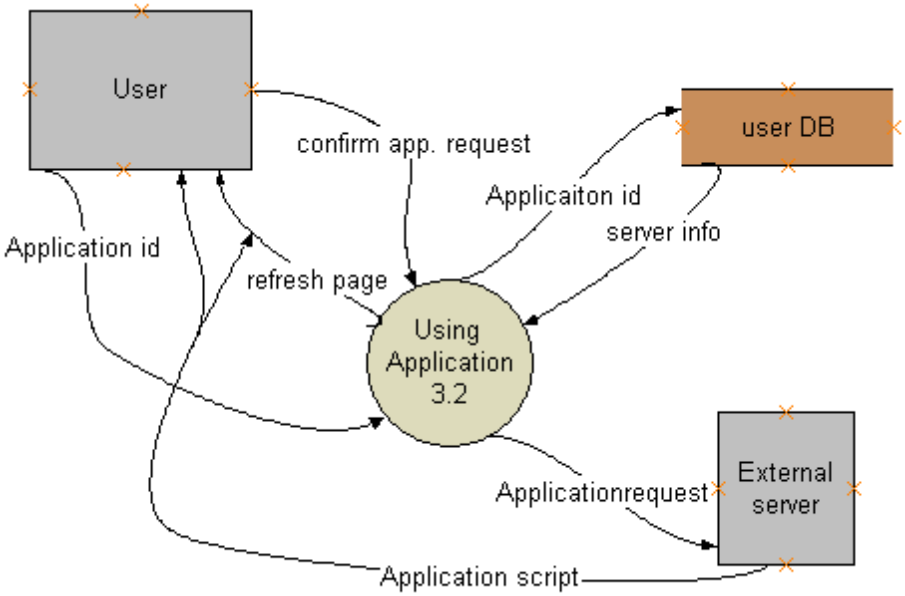

group id
User
group page
joining a group 1.3
group id,user id
User DB


topic search
group list
selected group
Searching an expert 1.2
expert list
User
group query
group list
Topic query
expert list
User DB


Expert user (High Level)
Entry
Dictionary DB
Writing to dictionary 1.5
entry, user id


group info
group page
creating a group 1.4
User
user id,group info
User DB


Developer User
code file
user DB
Application info
API
3.1
Application data
External Server
application
User


User
confirm app. request
user DB
Applicaiton id
server info
Application id
refresh page
Using
Application
3.2
External
server
Applicationrequest
Application script

4.2.4 Data dictionary

Name	Login data
Where used	USER output 2.1 Input
Description	User e-mail address and password

Name	Unique jabber id
Where used	2.1 output jabber server input
Description	Unique user id

Name	E mail and password
Where used	2.1 output user mail server input
Description	User e mail and password

Name	Validation
Where used	User mail server output 2.1 input
Description	Boolean value about the existance of the login data

Name	Profile
Where used	User DB output 2.1 input
Description	User profile data

Name	User query
Where used	2.1 output user DB input
Description	User related SQL queries

Name	Main page
Where used	2.1 output user input
Description	Main page after login

Name	Profile information
Where used	User output 2.2 input
Description	HTML form about user profile infos

Name	Profile information
Where used	2.2 output user DB input
Description	SQL update queries about profile infos

Name	Friend id
Where used	User 1 output 2.5 input
Description	User 2 id

Name	Messages
Where used	User1 input output user 2 input output Jabber server input output
Description	XML based texts for IM

Name	Jabber ID
Where used	2.5 output jabber server input
Description	Jabber id used for IM

Name	Search results
Where used	2.3 output input User DB output user input
Description	Results of the friend search

Name	Friend information
Where used	User output 2.3 input
Description	Information about a user for searching

Name	Information query
Where used	2.3 output user DB input
Description	Select queries for search

Name	Search text
Where used	User output 1.1 input
Description	The text for searching in dictionary

Name	Search result
Where used	1.1 input output user input dictionary DB output
Description	Result of the search

Name	Confirmation
Where used	User 2 output 2.4.2 input
Description	User 2 confirms the user1's friend request

Name	Friend request
Where used	2.4.1 output user 2 input
Description	Request to be friend

Name	Confirmation info
Where used	2.4.2.output user 1 input
Description	Message to user 1 after user2's confirmation

Name	Friend info
Where used	User 1 output 2.4.1 input
Description	User 2 profile information

Name	User 1,2 id
Where used	2.4.2 output user DB input
Description	Insert query to user DB

Name	Info query
Where used	2.4.1 output User DB input
Description	Select query for user2 search

Name	Results
Where used	User DB output 2.4.1 input
Description	User 2 information results

Name	Topic search
Where used	User output 1.2 input
Description	Text to search in groups

Name	Group list
Where used	1.2 output input user DB output user input
Description	The list returned after topic search

Name	Selected group
Where used	User output 1.2 input
Description	Id of selected group by user

Name	Expert list
Where used	1.2 output input user DB output user input
Description	Available expert list of the selected group

Name	Topic query
Where used	1.2 output user DB input
Description	Select query for topic search

Name	Group query
Where used	1.2 output user DB input
Description	Select query to get group id

Name	Group id
Where used	1.3 input user output
Description	Id of the group to join

Name	Group page
Where used	1.3 output user input
Description	Group main page

Name	Group id , user id
Where used	1.3 output user DB input
Description	Insert query to add user to group

Name	Group info
Where used	User output 1.4 input
Description	Informations about the group will be created

Name	Group page
Where used	User input 1.4 output
Description	Group main page which just created

Name	User id,group info
Where used	1.4 output User DB input
Description	Insert query for creating a new group and add the user to this group

Name	Entry
Where used	Expert User(High Level) output 1.5 input
Description	Dictionary entry

Name	Entry,user id
Where used	1.5 output dictionary DB input
Description	Insert query for adding entry to dictionary DB

Name	Code file
Where used	Developer user output 3.1 input
Description	PHP code file

Name	Application information
Where used	3.1 output user DB input
Description	Insert query (Application name,user id v.s.)

Name	Application Data
Where used	3.1 output external server input
Description	Scripts which will run on the developers' server

Name	application
Where used	External server output user input
Description	HTML formatted files

Name	Confirm app. request
Where used	User output 3.2 input
Description	Conformation to the application request

Name	Application ID
Where used	User output 3.2 input
Description	The ID of the application which is wanted by the user

Name	Server info
Where used	User DB output 3.2 input
Description	Information about the server on which the application runs

Name	Refresh page
Where used	3.2 output user input
Description	The page which is refreshed and sent to the user

Name	Application script
Where used	External Server output User input
Description	Application, returned to the user

Name	Application Request
Where used	3.2 output External server input
Description	Application wanted by the client

5.0 Interface Design

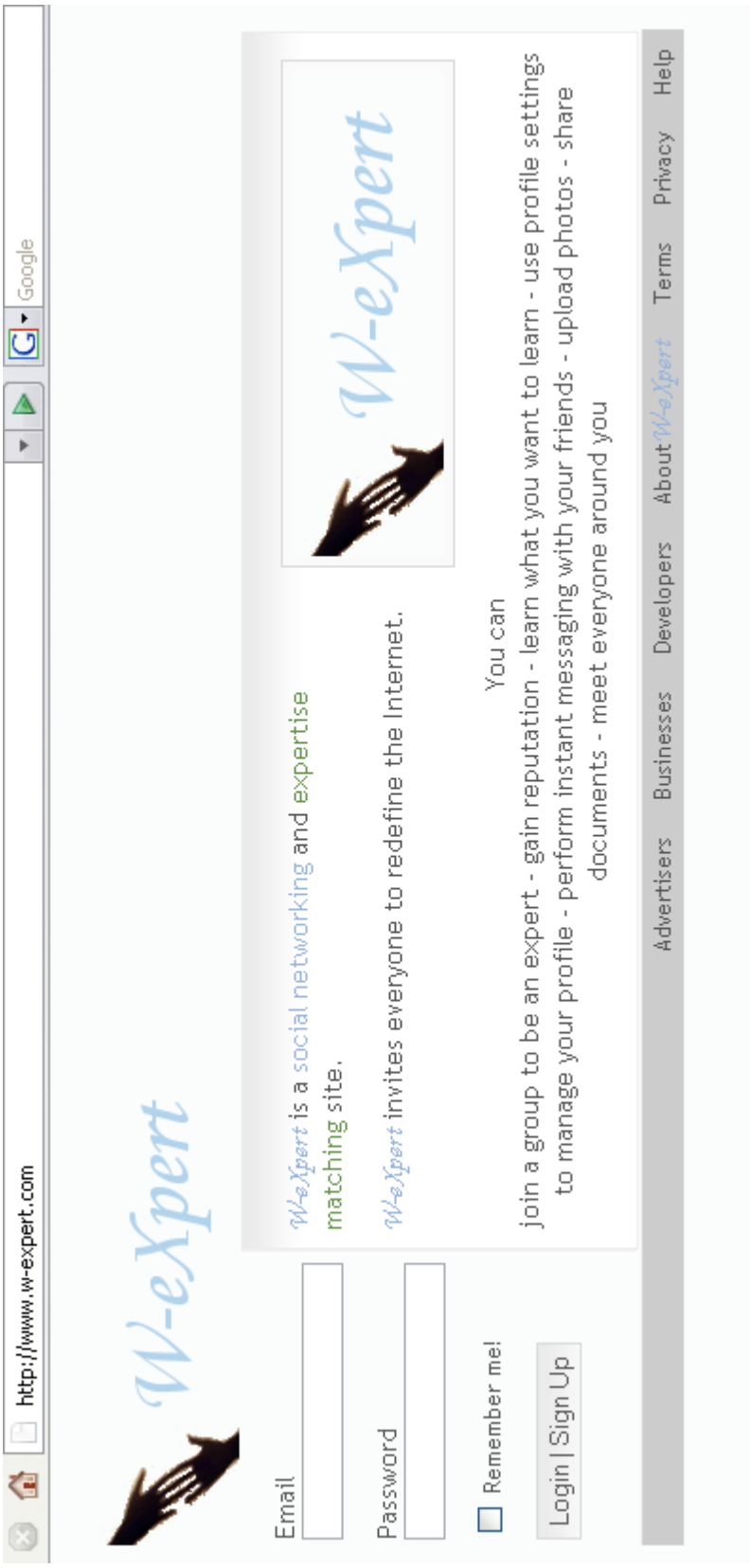
In this part we will show examples from our projects website. A general description of these pages are also given below.

5.1 Login page

This page is used not only for logging in but also for signing in. The right side of the page is reserved for these issues. A user or a newcomer enters a valid email address and its password to login or sign up. There is a 'Remember me!' checkbox for an option. If this is checked the user's password will be remembered.

The right side of the page includes our logo and one sentence of information about what the site/platform is. A slogan and capabilities, the platform has, has been specified below that.

There is a bottom menu at the bottom of the page. It includes other information related with the site. This menu is contained by most of the pages of the site.


http://www.w-expert.com
Google
W-eXpert
Email
Password
Remember me!
Login | Sign Up
W-eXpert is a social networking and expertise matching site.
W-eXpert invites everyone to redefine the Internet.
W-eXpert
You can
join a group to be an expert - gain reputation - learn what you want to learn - use profile settings to manage your profile - perform instant messaging with your friends - upload photos - share documents - meet everyone around you
Advertisers Businesses Developers About W-eXpert Terms Privacy Help

5.2 Profile page

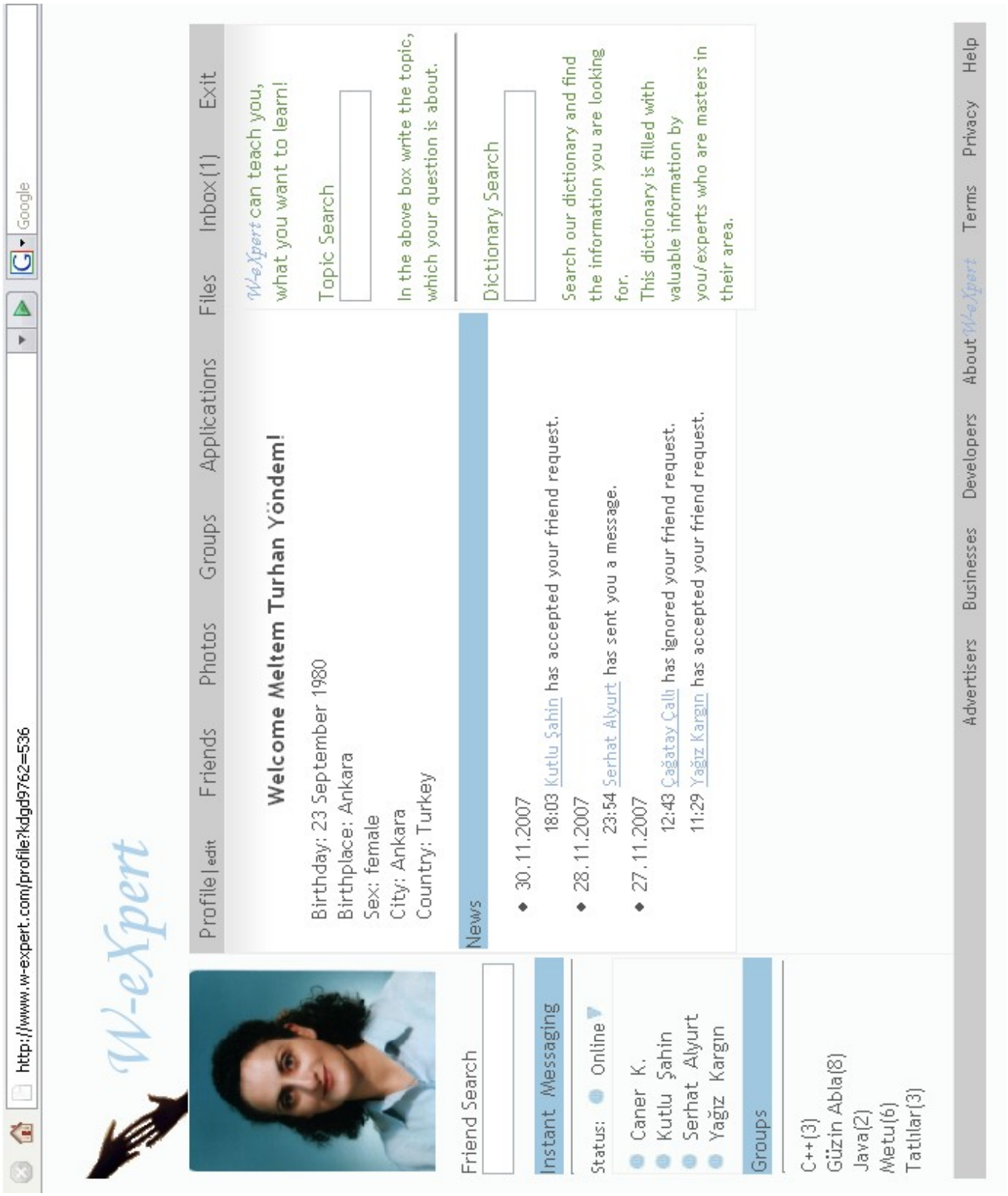
This page is the main page of the system for a user. A user can use every capability of the platform by using this page.

There is a horizontal menu at the top. This menu is included by most of the pages of the site, especially pages which are related with the user. When you click on the ‘Profile’ you will go to this page. You can edit your profile by clicking on the small ‘edit’. If you click on ‘Friends’, a friend list will welcome you. You can reach your all photos using ‘Photos’ menu item. If you click on groups, applications and files, you will see your groups, applications and files respectively. You can see a number next to the ‘Inbox’ menu item. That number shows the user’s unread messages. If you click on ‘Inbox’, you will read your messages.

The left side of the page includes the display photo of the user, friend search option, instant messaging feature and groups which the user is a member of. In the instant messaging part the user can change her/his instant messaging status and can only see his/her friends who are not offline. Below that part the groups of the user is listed. The numbers in parentheses are the maximum number of users who can ask a question to user about that group’s topic. The availability status in the groups will be adjustable from this page.

The middle part of the page contains the profile information that the user has specified. After that part there is the news panel which contains the actions of the user. If you click on the blue user names you will go to the profile page of that person. Below that there will be the applications the user uses.

The right part of the page is reserved for the expertise matching feature. The user can search for a topic to see the groups related with it, then he or she can ask a question to an expert. Or the user can search the dictionary for the immediate answer. This part of the page is included by most of the pages.


http://www.w-expert.com/profile?kdgd9762=536
Google
W-eXpert
Profile | edit
Friends
Photos
Groups
Applications
Files
Inbox(1)
Exit
Welcome Meltem Turhan Yöndem!
Birthday: 23 September 1980
Birthplace: Ankara
Sex: female
City: Ankara
Country: Turkey
W-eXpert can teach you, what you want to learn!
Topic Search
In the above box write the topic, which your question is about.
Dictionary Search
Search our dictionary and find the information you are looking for.
This dictionary is filled with valuable information by you/experts who are masters in their area.
Friend Search
News
30.11.2007
18:03 Kutlu Şahin has accepted your friend request.
28.11.2007
23:54 Serhat Alyurt has sent you a message.
27.11.2007
12:43 Çağatay Çallı has ignored your friend request.
11:29 Yağız Kargın has accepted your friend request.
Instant Messaging
Status: Online
Caner K.
Kutlu Şahin
Serhat Alyurt
Yağız Kargın
Groups
C++(3)
Güzin Abla(8)
Java(2)
Metu(6)
Tatlılar(3)
Advertisers
Businesses
Developers
About W-eXpert
Terms
Privacy
Help

5.3 Groups page

The middle part includes the information about the group and recent actions. If you click on the blue user nicknames you will send a communication request to that expert to start the question answering period.

In the right part of the page the user can see the available experts and their ratings. The user can send a communication request to that expert to start the question answering period by clicking on the ‘Request’ link. If the user clicks on the ‘Advised Documents’ he/she will see the documents that the expert has uploaded.

The other parts are same as the profile page.

http://www.w-expert.com/group?id=78762=java
Google
W-eXpert
Profile | edit
Friends
Photos
Groups
Applications
Files
Inbox(1)
Exit
Java
Description:
Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun's Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. This group is created to satisfy the needs of the java learners.
Keywords:
Java, Object Oriented, C++
Creation Time:
23.10.2007 04:15
Available Experts
Yöndem
Rating: 10 Request
Advised Documents
Alican145
Rating: 2.6 Request
Advised Documents
Group Dictionary Search
Search your/our dictionary and find the information you are looking for.
This dictionary is filled with valuable information by you/experts who are masters in this area.
Friend Search
Instant Messaging
Status: Online
Caner K.
Kutlu Şahin
Serhat Alyurt
Yağız Kargın
Groups
C++(3)
Güzin Abla(8)
Java(2)
Metu(6)
Tatilar(3)
Recent Actions
27.11.2007
17:03 alican145 has joined to the group.
25.11.2007
10:45 Yöndem has written a new entry to the dictionary.
24.11.2007
12:05 ÇÇallı has written a new entry to the dictionary.
05:05 yk has joined to the group.
Advertisers
Businesses
Developers
About W-eXpert
Terms
Privacy
Help

5.4 Instant Messaging Box

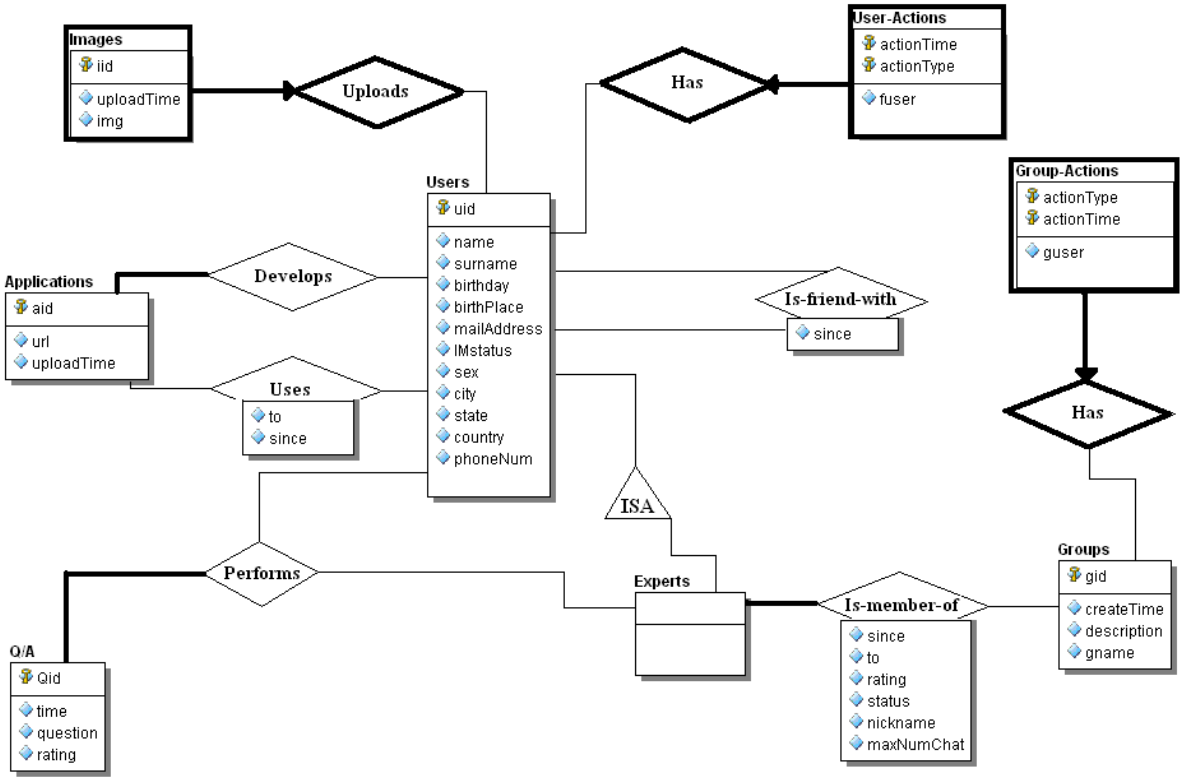
The user can see the whole dialog with the above text box and can send an instant message if he/she writes in the below box.

6.0 Data Design

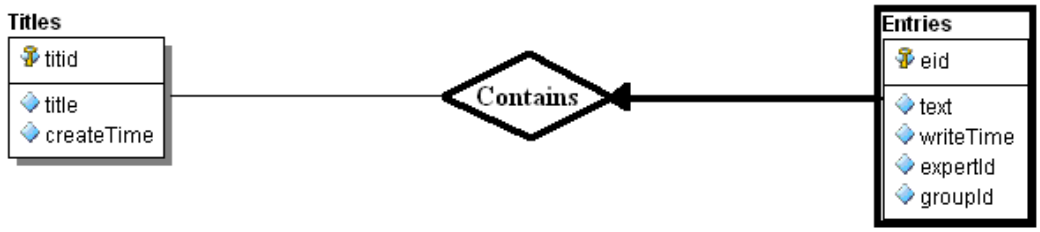
In the sixth part of design we will describe our data design in Entity Relationship Diagrams and given SQL queries (Appendix 9.3). A general description of the tables are also given.

6.1 Entity Relationship Diagrams

USER DB:



DICTIONARY DB:



DOCUMENTS DB:

6.2 Relational Schema

Translating E/R diagram to relations we have some relations that are not in all of the normal forms. Firstly we have specified the functional dependencies and organized the relations to make them be in 3NF and BCNF. Then we have searched for multivalued dependencies and we can not find any. So the relations are in 4NF. Finally we come up with these relations given below.

USER DB:

- users (uid, name, surname, birthday, birthplace, mail_address, im_status, sex, city, state, country, phone_number)

'users' table stores the profile information of the users.

- applications (aid, name, url, upload_time)

'applications' table stores the information of applications. 'name' attribute must be unique, because there can be no two distinct applications with the same name.

- question_answer(qid, time, question, rating)

This table stores the information of communication of users with experts. 'rating' attribute

stores a given specific rating from user to expert for this specific question-answer (communication) period.

- groups (gid, name, creation_time, description)

Information of groups is stored in this table. 'name' attribute must be unique, because there can be no two distinct groups with the same name.

- is_member_of (expert_id, group_id, since, to, rating, status, nickname, max_num_chat)

This table stores the expert information related with groups. 'nickname' attribute must be unique, because there can be no two distinct experts with the same nickname.

- performs (user_id, expert_id, question_id)

This table relates 'users', 'is_member_of'(experts), 'question_answer' tables. A row in this table means a user and an expert have been communicated in that question-answer period.

- group_actions (group_id, action_time, action_type, expert_id)

'group_actions' stores the actions in the group (e.g. new member, new dictionary entry).

- is_friend_with (user_id1, user_id2, since)

This table includes who is a friend with who.

- requests (from_user_id, to_user_id, request_time)

Suspended friend requests are stored in this table.

- user_actions (user_id, action_time, action_type, friend_user_id)

'user_actions' stores the actions for that user (e.g. new friend request, friend request acceptance, new message).

- images(user_id, image, upload_time)

This table stores users' images.

- develops (user_id, application_id)

This table stores which user develops which application.

- uses (user_id, application_id)

'uses' table stores which user uses which application.

-messages(message_id, from_user, to_user, title, text, time,reply_id)

messages table stores which user send/reply a message to which user.

A design decision: Since all experts are users and they can act like a user who is not an expert, we have 'users' table which includes all users and experts. We haven't created a table that stores the data related with experts, because experts do not have any special attribute other than a simple user. They only have some other attributes which are however related with the group that expert is a member of. Therefore no need to keep an expert table, because experts are presents only for their groups.

DICTIONARY DB:

- titles (tid, title, creation_time)

This stores the titles in the dictionary. 'title' attribute must be unique, because there can be no two distinct entries with the same title.

- entries (title_id, text, write_time, expert_id, group_id)

Entries under titles are stored in this table.

DOCUMENTS DB:

- documents(did, upload_time, doc, user_id, num_downloads, is_shared)

Documents are kept in this database and in this table. You can find creation queries of the tables given above at Appendix 9.3.

6.3 Description of ER Diagrams

USER DB:

Any operation on user profile will be directly get into interaction with "Users" table. In start we will take user's mail address and give him an auto incremented id. S/he can later edit his/her profile information with given functions or we will get some ready information from other sites(like facebook,cember.net etc) and add them to our database. User may also want to display their (or some other) picture as their profile picture. Or s/he may one to show a high school photograph long forgotten. So we need to keep images and their users. Our Images table is just doing this: keeps the image_id which is auto incremented with every new image, image in blob (which is a binary large object that can hold a variable amount of data) format, user_id of who is uploaded and the time image is uploaded.

Application information users' created are stored in "Applications" table. Whenever an api is created we will add its auto incremented id and the time api is uploaded. Its name and url will be defined by user, however there will be no two distinct applications having the same name.

Developers of these applications will be kept with Develops relation. We will take user's id from Users table and keep it with a api_id which is an auto incremented value. Removal of the user or the application will cause removal of these entries in our database.

For the list of applications a user has added to his profile we will keep a Uses table. When an application is added or removed from user's list we'll update this table. Also changes with user_id and/or api_id will cause changes in this table.

To handle group functions we have a general Groups table. When a user creates a group we will give it an auto incremented id and record its creation time. Group name and description will be filled by user. However there will be no two distinct groups with the same name. In addition we want to minimize very similar named groups exists. To do this we'll search database for exactly the same names and prevent user to create that group with that name. We will also give similar groups with similar names and ask for if s/he wants to join existing ones or still create a group. Creator will also give some keywords for his/her group to be founded in searches and question answering system.

In our system everybody who joins a group will be expected to be an expert with group interests (or lets say has some knowledge about the concept). So we keep an Is_member_of table for group members. When a member is added to group we take new member's id, id of the group s/he is joining and give him a default rating and keep all these with his membership start date. We can still keep his information with this group after he leaves. In this case we will fill till attribute of the table with his leaving date and hide his information with group. Members of the group may manage his answering quotas, for example I am an of C group and I am also have some knowledge about Java; I want to answer C group questions more than Java questions then I want max 8 questions to be answered from C and 2 questions from Java. As long as we make expert_id and group_id as primary keys we can change max_num_chat for each group of a user. Users may want to hide their names so we also keep nicknames for each group user joined. Rating system is the key to dictionary, people with high reputation have access to dictionary and this leads higher reputation.

Our real specialty will be question answering system. When people search for answers to their questions they want answers directly to their questions, not some related results produced by some boring search engines. To be better then these sites we provide question answering by others in a well designed system. Everything about answering system is planned really carefully but where do we keep these information. Rating is already kept within Is_member_of table but we need to keep and provide more resources to develop this system. So we created Question_Answer table which records questions with an auto incremented id. To get better evaluations we get rating per question and keep it stick to the question with its answered time variable. After having all the

calculations we will update experts rating in his/her group. So we need another table with expert's id, user's id and the question's id. Exactly Performs table is responsible for keeping this information. In the end whenever a question answering service started, a new row will be created for Question_Answer table and Performs table. In the end of the dialog with user's rating for expert new rating will be calculated and Is_member_of table will be updated with the new rating of the expert.

We will provide "chat with friends" service to our users, and we don't want anybody to get disturbed by others who are not friends of our user. So we need to know who is friend with who. People will also want get in contact with their friends. When someone adds a friend to his/her friend list we will create a row with user_ids and the time of their relation begins. We need a friend request protocol indeed before adding friend. Indeed people shouldn't add friends without other side's confirmation. So we keep the records of these records as two people's ids (receiver & sender) and the time request has been sent.

So what about messaging? We have to provide users with messaging service for lots of reasons (don't want to chat, user offline, reminder etc.). So we need to keep these messages somewhere. Messages table satisfies our needs with its attributes; an auto incremented id for each message to keep them easy to find, form_id and to_id to keep information about who send this message to who, a title to the message, and of course the message. To keep the track of messages that are reply to a message, we keep another attribute, reply_id, which is 0 by default. If a new message is created it will be created by its default reply_id so it will lead us to nowhere, because there are no messages with 0th id. Messages that are replies to messages that had been sent before will have this attribute filled indeed.

After describing friends and messaging services, now we have to inform our users about these changes (actions). A user will want to be informed when something important happened indeed. So we keep a User_Actions table for keeping records of users interactions with others like New Message, New Friend Request etc. Whenever these actions occur, we will add a row to our table with two user ids, action type and keep them with the action occurrence time.

What about groups? What are they doing now? Yes we also keep records about their actions. Our Group_Actions table is holding actions like New Member, New Entry etc. with expert_id that caused this action and group_id of this action occurred of course. Action time will also provide important information for the action.

DICTIONARY DB:

After having user database described in advance, we have produced the dictionary database to keep entries of the experts. We want this to be as simple, enjoyable and understandable as it should be. People in most dictionaries get lost in what they search and get bored with just one person's thoughts (if it is provided of course). So we got our inspiration from "ekşisözlük" and added some functionality to it. Like ekşisözlük, there will be titles opened by permitted people (experts with high reputation) and entries under these titles again written by permitted people. So we have two tables. First one of them is Titles table which keeps title and its auto incremented id for each title and their creation time. Whenever a creation of a title for some topic is occurred and insert query will be called. And when one of our experts wants to add some entry under the title, his/her entry will be kept with an auto incremented entry_id, his/her expert_id and group_id of group s/he has become expert in. Any change on the title will cause this table to be changed according to these changes.

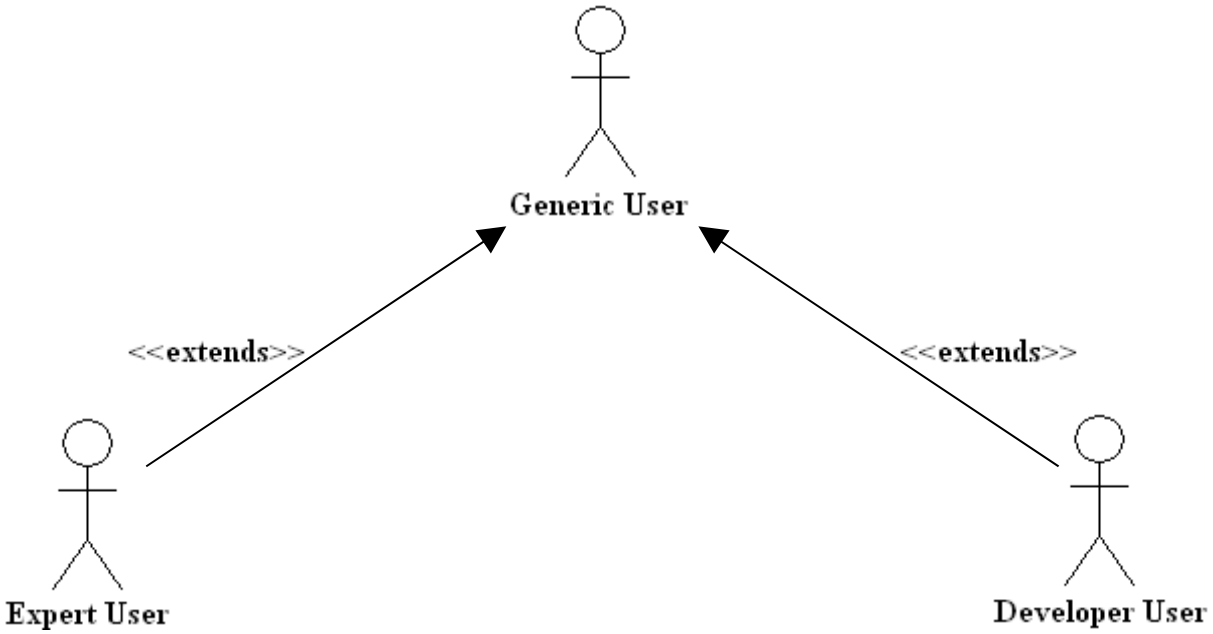
DOCUMENTS DB:

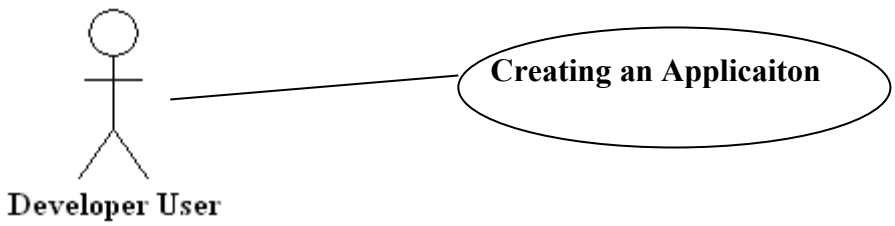
We also want to provide our users with document sharing opportunity. To achieve this we need a very large database with a document_id provided to each document. We will keep documents with blob like in Images table, but will give more space compared to image storage. We will give users the opportunity to share these documents or keep them secret, with an enum of (all, friends_only, no) options. We will also keep record of the document about how many times it has been downloaded. Of course owner of the document with user_id and upload time of the document will provide us valuable information to be kept.

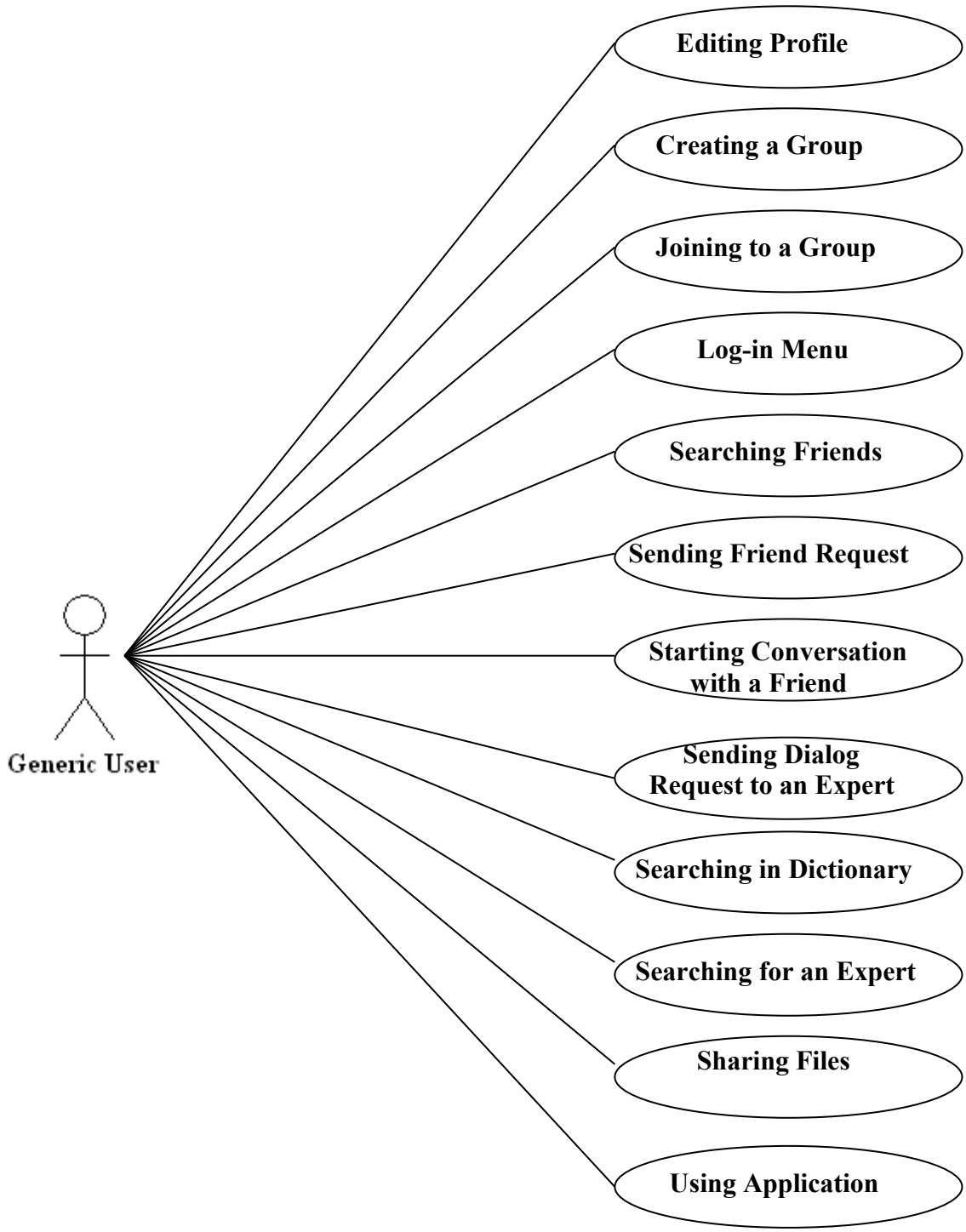
7.0 Procedural Design

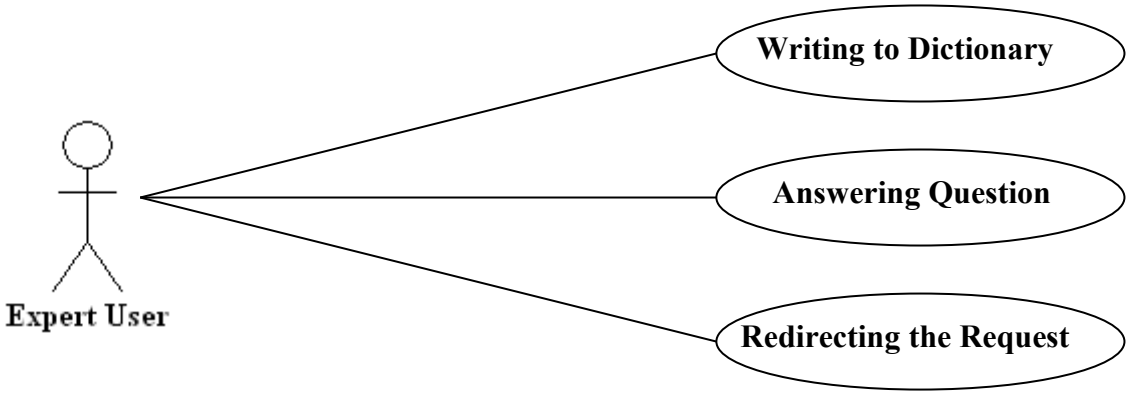
Our project's procedural design is described in this part. Visual support of the Use case diagrams, sequence diagrams and the State

7.1 Use Case Diagrams



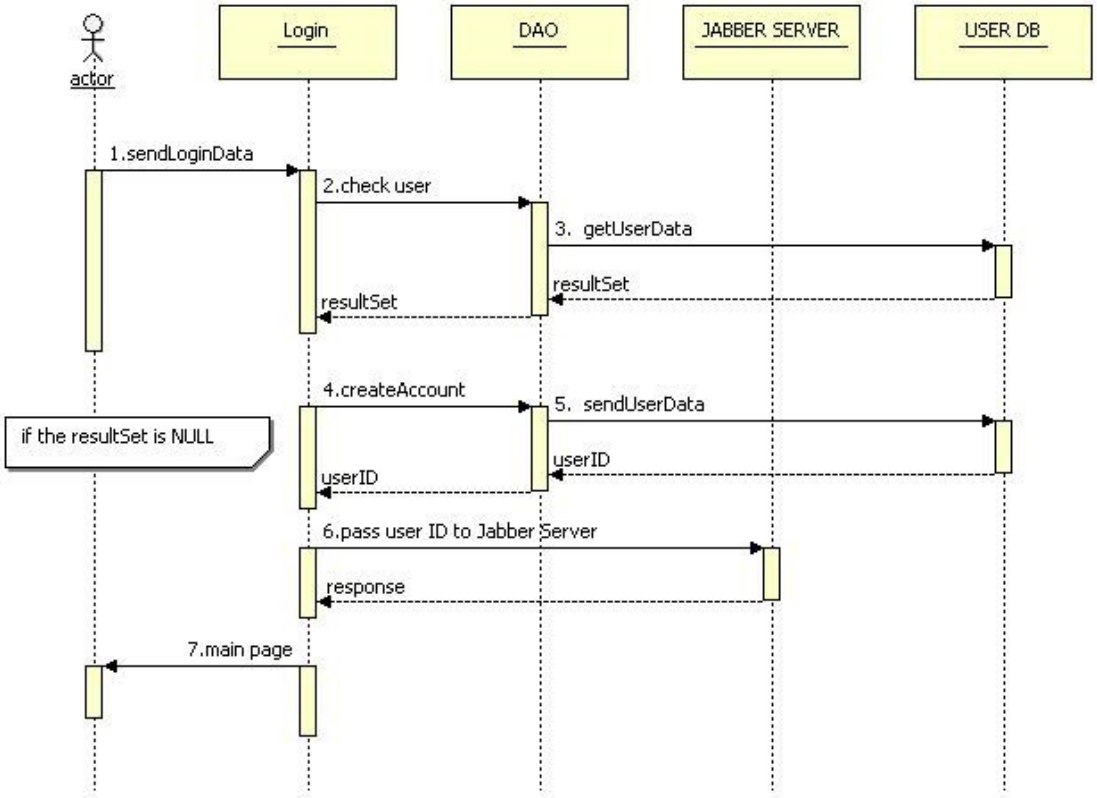



Editing Profile
Creating a Group
Joining to a Group
Log-in Menu
Searching Friends
Sending Friend Request
Starting Conversation with a Friend
Sending Dialog Request to an Expert
Searching in Dictionary
Searching for an Expert
Sharing Files
Using Application
Generic User

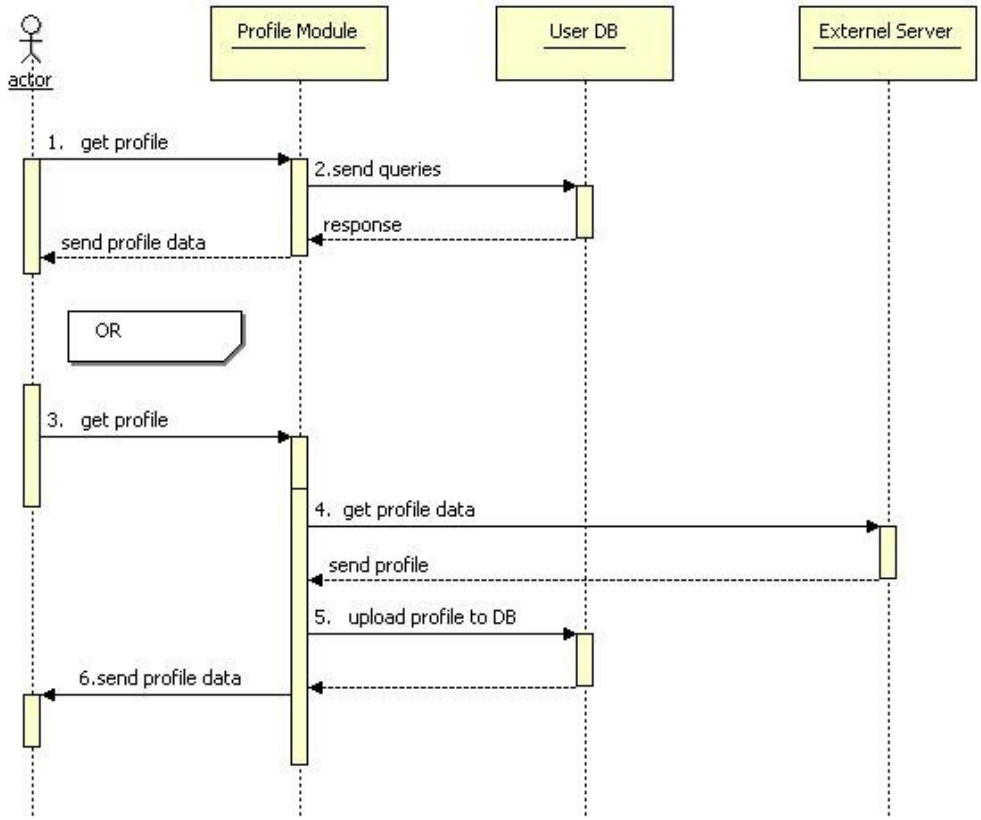


7.2 Sequence Diagrams

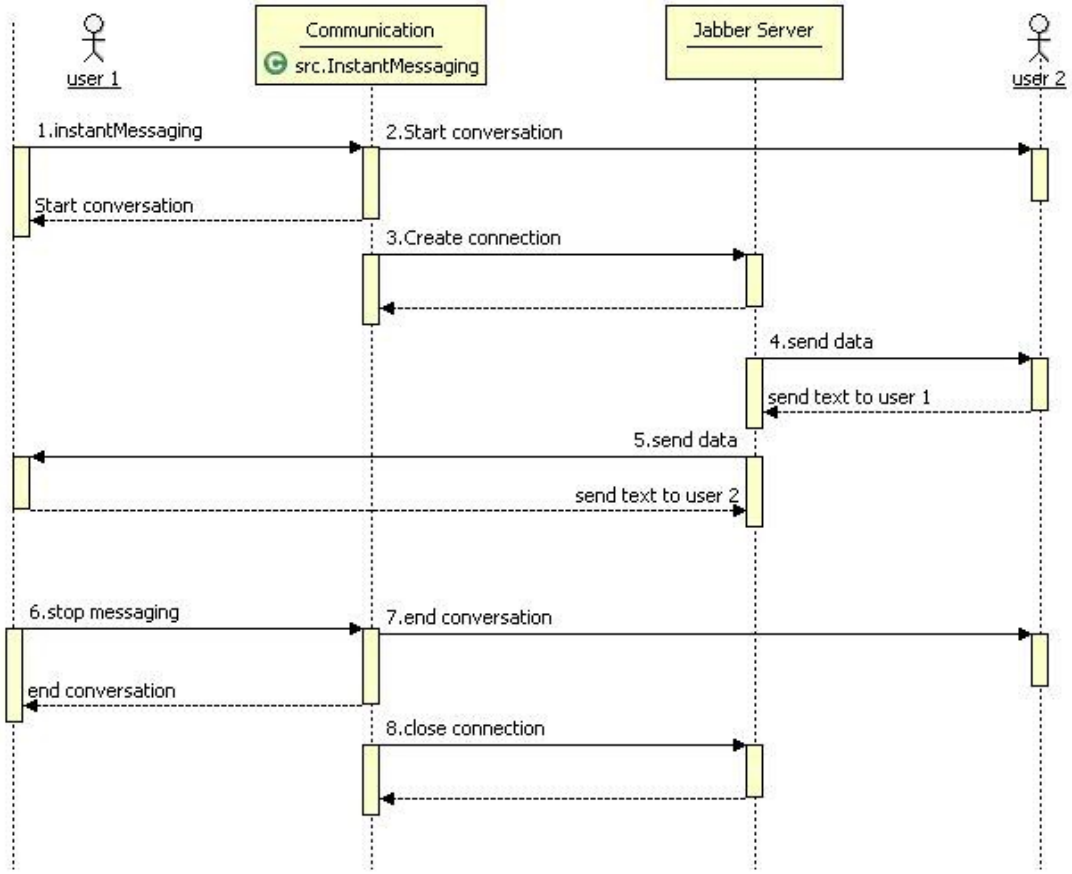
Login Module:



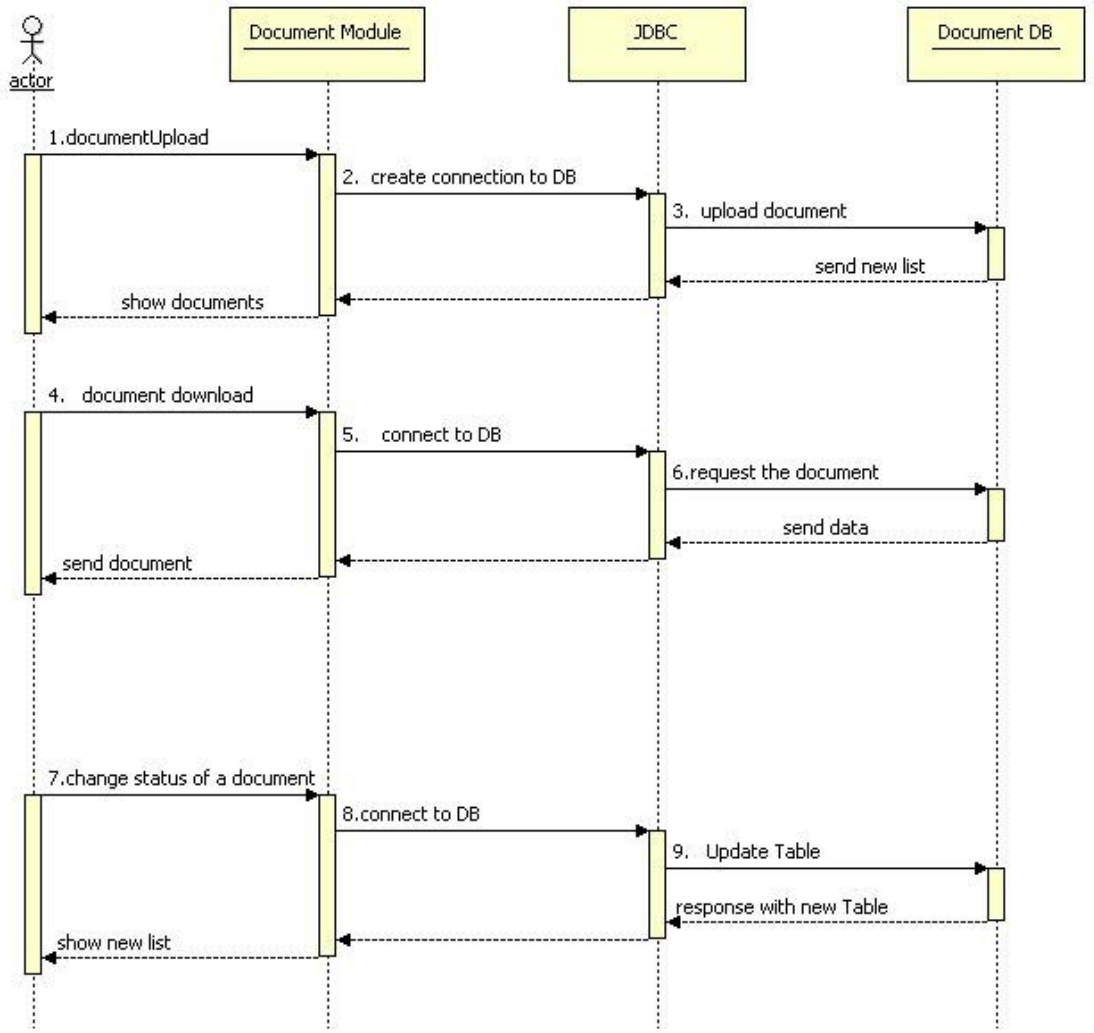
Profile Module:



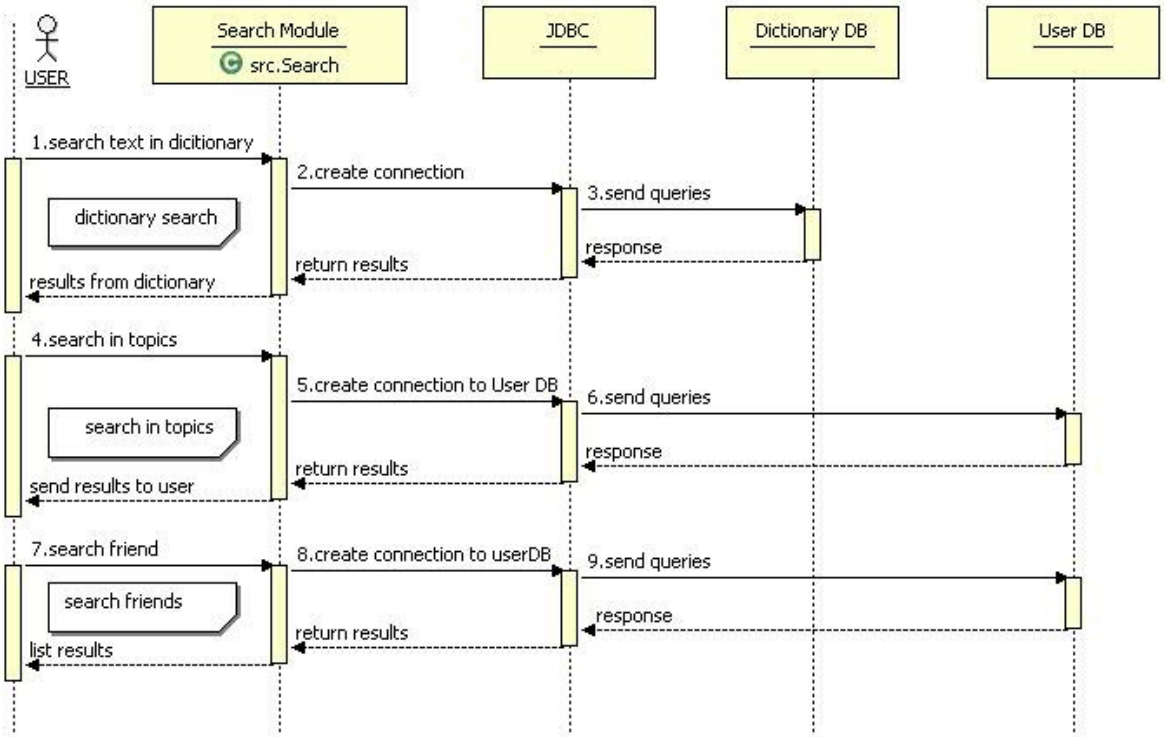
Instant Messaging Module:



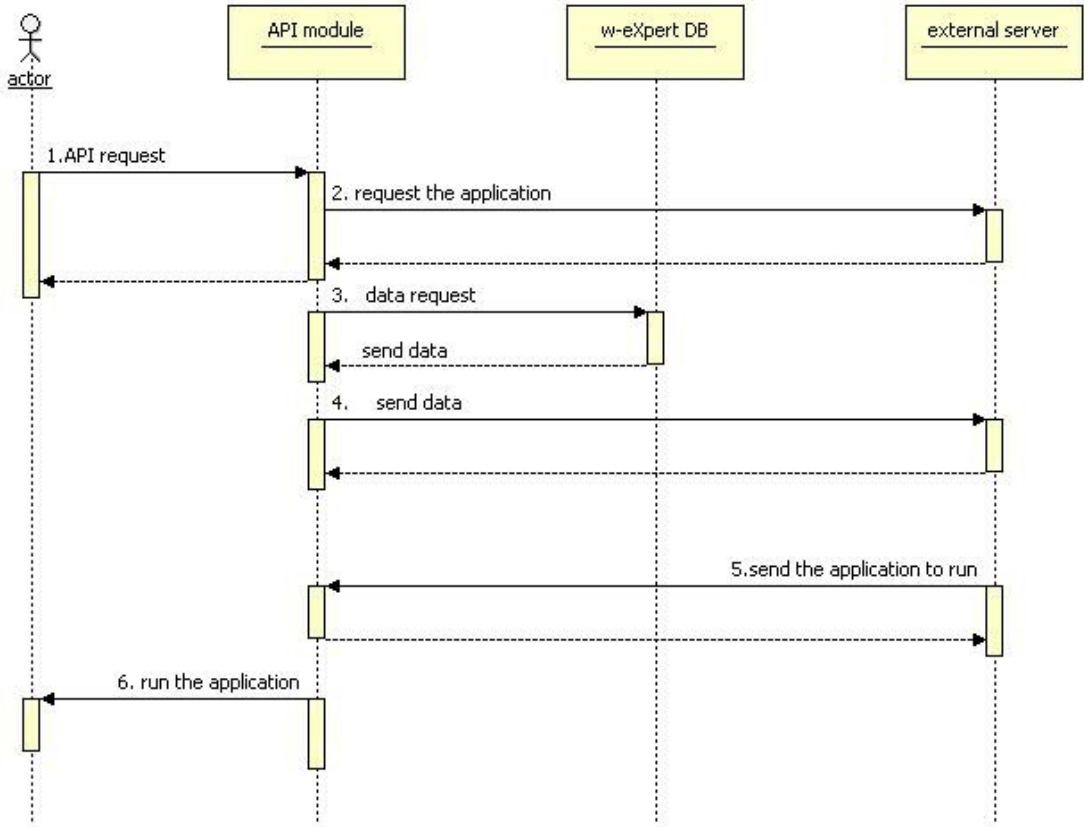
Document Module:



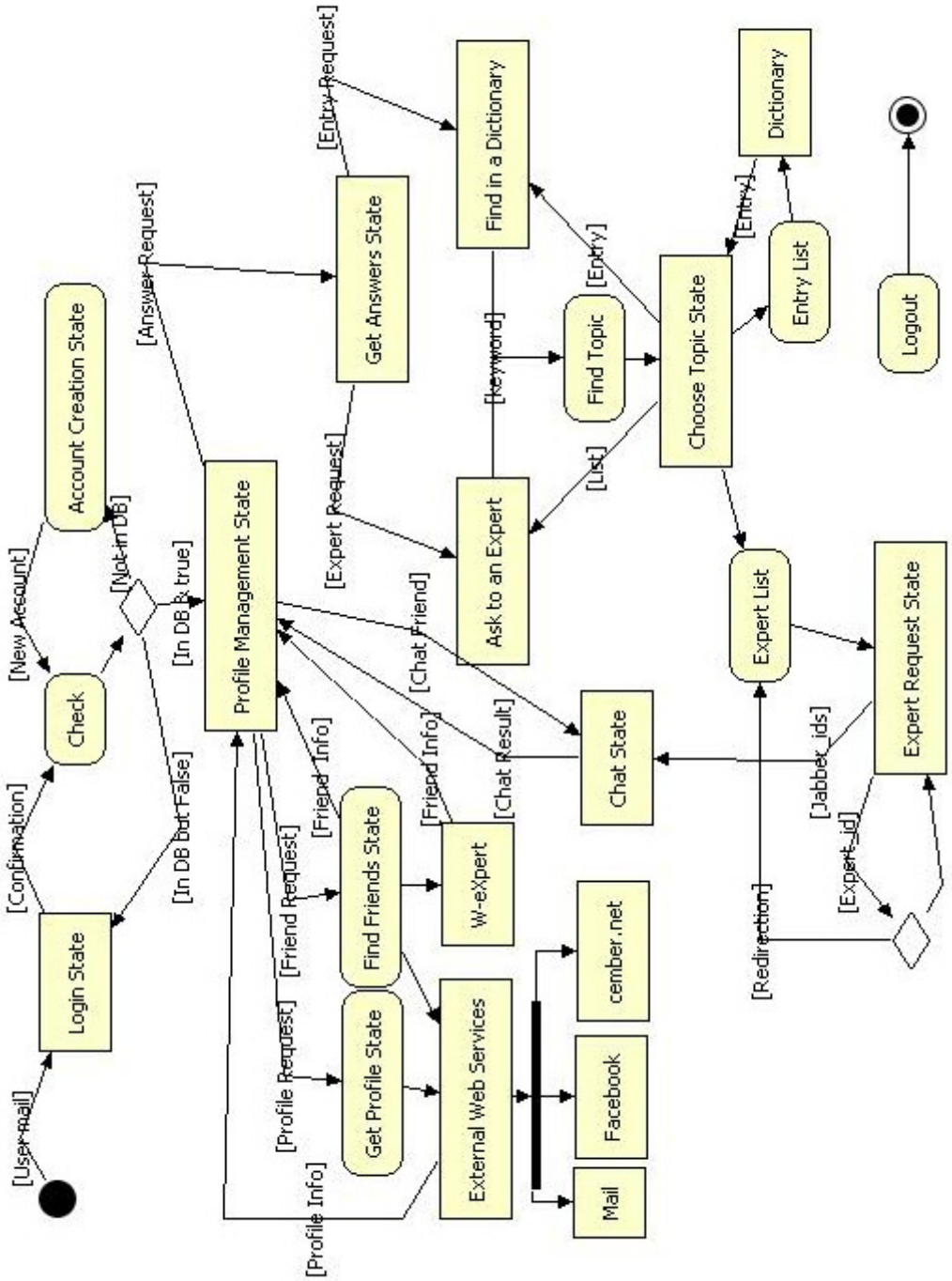
Search Module:



API Module:


actor
API module
w-eXpert DB
external server
1.API request
2. request the application
3. data request
send data
4. send data
5.send the application to run
6. run the application

7.3 State Transition Diagram



8.0 Inferences

This part of the report describes what we have learned till now and what we will do later. A general conclusion is included.

8.1 Works Done So Far

Until now from the start we have been dealing with general system about our project. Our environment will be a really large platform providing lots of services to its users. Nowadays there are lots of others with small changes in each of them. But we will come with a really big change gathering all of them around us.

When our project is finished question answering system by users will be accomplished. This service is our main project. So it has lots of configurations with its system. We have gathered information about how we can connect two people. Firstly our newly arrived problem with requirement analysis report was on the floor to be dealt with. We have to gather experts to our system. We will lure them; they will surely want to help people. So we created a rating system for question answering. After users have their answers from expert a small rating screen will greet them. People will rate experts and they will gain reputation. Not enough? We also thought about it and added some new feature to this rating system: there will be a dictionary for experts to write, but not all of them. A newly arrived user can easily introduce himself/herself as an expert about some topic. Unless they are at a certain level in their progress in this topic they cannot add entries to dictionary.

We have lots of discussions about this dictionary system and expert hierarchy. In the end we come up with lots of ideas. For example an expert can redirect a question to someone who can answer in his/her group. Dictionary search will be a useful service for frequently asked questions. Expert hierarchy and the feeling of becoming superior to others will encourage people to help with more determination. Advanced rating opportunity for our patient users will lead experts to be more polite. On the expert side question preview sending with request will help preventing irrelevant questions to be directed experts and wrong rating results. Redirection will help experts in two ways:

one of them is increasing reputation of newly joined members faster. Second one is, think about a master with his/her topic, s/he can have the opportunity of not answering really easy questions.

Other than ideas that will be added to our project we have done lots of researches about Jabber, JSP, AJAX. We started with some implementations of our prototype in basic. Some Graphical User Interfaces are generated to see things more clearly. Databases are created for most of the project. With the help of class diagram and sequence diagrams we have seen the project classes and their works to do in advance. With the state transition diagram we divided our system into states and see how will we proceed step-by-step.

API service of Facebook was one week's whole topic. We tried to create an application in Facebook and see how things work.

8.2 Future Work

In the following weeks, till the day which the demo will take place, we will be working on the first prototype. Implementing this first prototype, we will certainly face with some problems, those will help us to clarify the final construction elements of our system. The necessary changes will be done and the details of the classes, sequence etc. will be determined. Later after the first run of our system prototype, we will be focusing to the details of the interface of the system. This part is also important to reach and interact with the users. Debugging and re implementations will be done to maintain a consistence and stability for our system. All these works will help us to see problems may appear and to get experience on the project. After all, the final design report will be written that will provide us to start implementing the project in the second term.

8.3 Conclusion

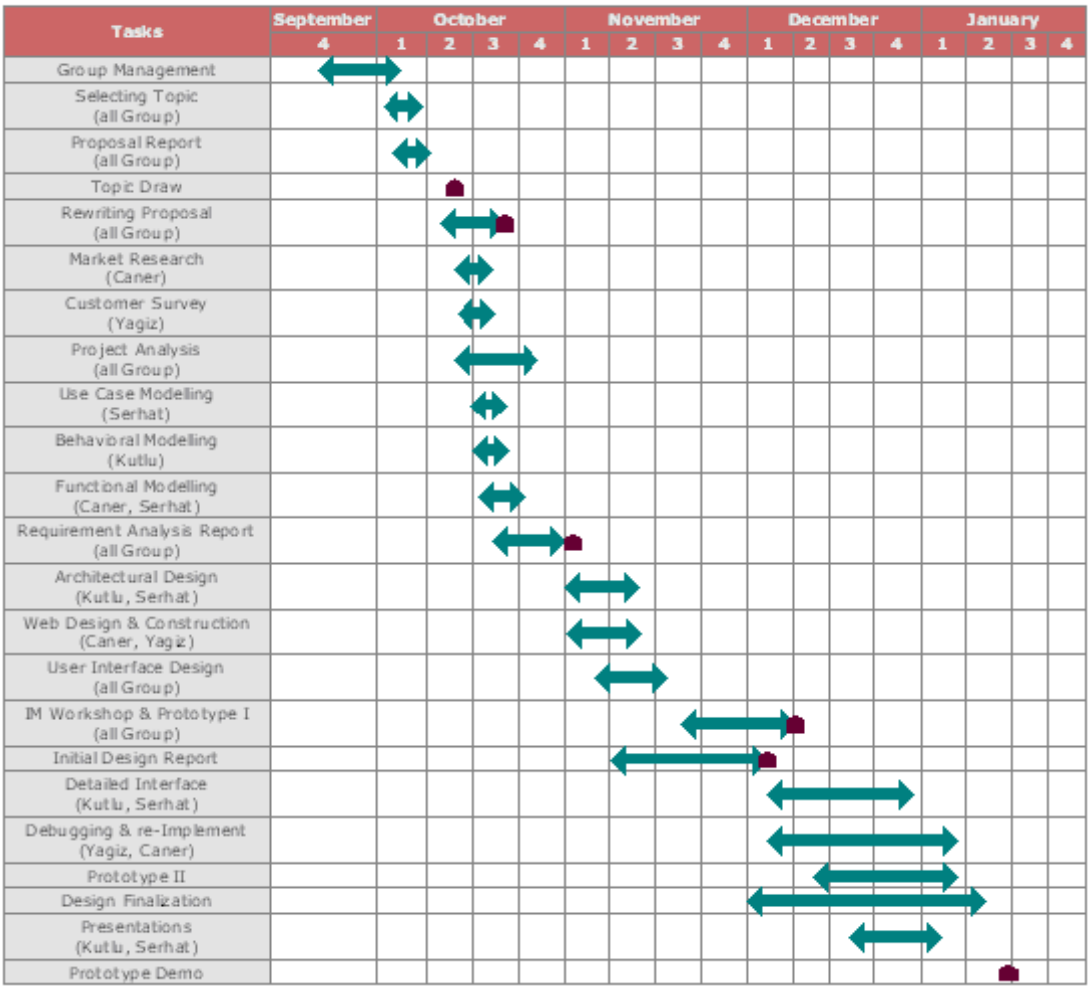
With this Initial Design Report we have stated what we have done so far and what we will be doing later. This Initial Design was really important for the project. Because we have started our project and see the problems we now have and we will have. Next week after this report submitted,

we want to prepare a prototype to see how things will work out and see if we have forgotten something. We have lots of work to do until this term finishes and we really want to come up with the best design in all projects.

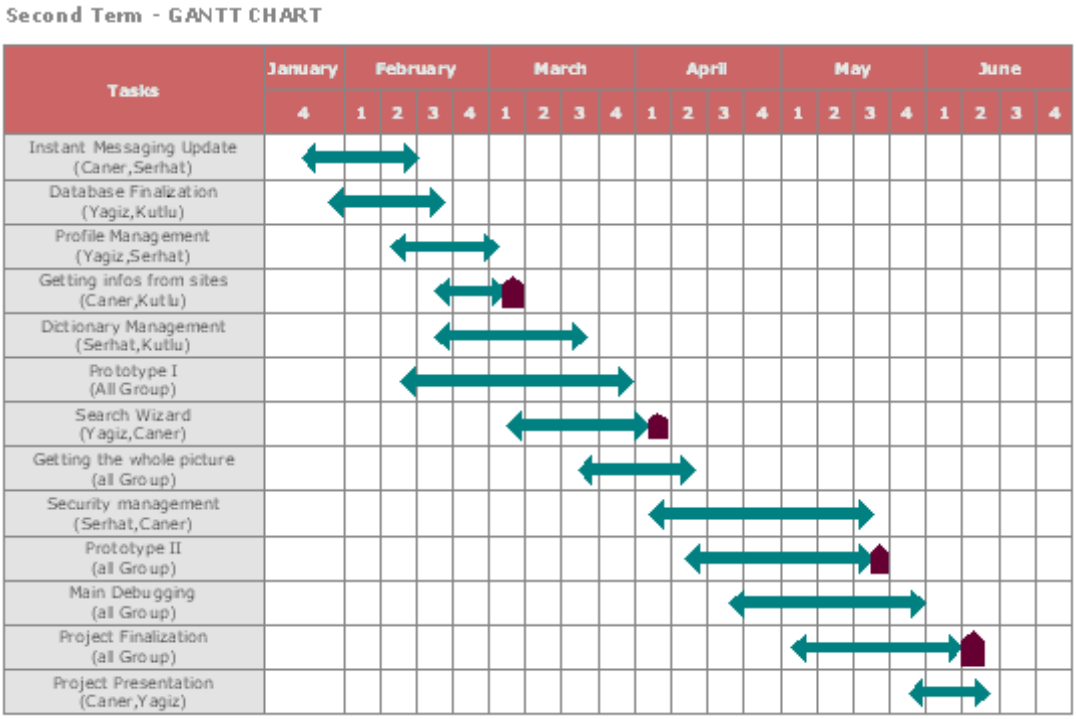
9.0 Appendices

9.1 Gantt Chart First Term

First Term - GANTT CHART



9.2 Gantt Chart Second Term



9.3 SQL Create Table Queries

Databases

```
CREATE DATABASE IF NOT EXISTS User_db

CHARACTER SET utf8;

CREATE DATABASE IF NOT EXISTS Dictionary_db

CHARACTER SET utf8;

CREATE DATABASE IF NOT EXISTS Documents_db

CHARACTER SET utf8;
```

Tables

```
/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/*--------------------------**W-eXpert**-----------------------*/

/*---------------------DATABASE by FREELANCERS-----------------*/

/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/

/*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*/


/*---------------USER DATABASE CREATION QUERRIES---------------*/

use User_db;

CREATE TABLE IF NOT EXISTS Users

(

     user_id INT UNSIGNED AUTO_INCREMENT,

     user_name VARCHAR(255),

     user_surname VARCHAR(255),

     birthday DATETIME DEFAULT '1900-01-01',

     birthplace VARCHAR(255),

     mail_address VARCHAR(255) NOT NULL,

     im_status
ENUM('Online','Offline','Busy','Away','Invisible','At_Lunch')   NOT
NULL DEFAULT 'Online',

     sex VARCHAR(50),

     city VARCHAR(255),
```

```
    state VARCHAR(255),
    country VARCHAR(255),
    phone_number VARCHAR(255),
    PRIMARY KEY(user_id)
);
CREATE TABLE IF NOT EXISTS Applications
(
    api_id INT UNSIGNED AUTO_INCREMENT,
    api_name VARCHAR(100) NOT NULL,
    url VARCHAR(255) NOT NULL,
    upload_time DATETIME NOT NULL,
    UNIQUE(api_name),
    PRIMARY KEY(api_id)
);

CREATE TABLE IF NOT EXISTS Question_answer
(
    q_id INT UNSIGNED AUTO_INCREMENT,
    ans_time DATETIME NOT NULL,
    question VARCHAR(255),
    rating INT UNSIGNED NOT NULL,
    PRIMARY KEY(q_id)
```

```
);

CREATE TABLE IF NOT EXISTS Groups

(

     group_id INT UNSIGNED AUTO_INCREMENT,

     group_name VARCHAR(255) NOT NULL,

     creation_time DATETIME NOT NULL,

     description VARCHAR(255),

     interests_keyword VARCHAR(50),

     UNIQUE(group_name),

     PRIMARY KEY(group_id)

);

CREATE TABLE IF NOT EXISTS Is_member_of

(

     expert_id INT UNSIGNED NOT NULL,

     group_id INT UNSIGNED NOT NULL,

     since DATETIME NOT NULL,

     till DATETIME NOT NULL,

     rating DOUBLE NOT NULL DEFAULT 0,

     status
ENUM('Online','Offline','Busy','Away','Invisible','At_Lunch','Not_
Answering') NOT NULL,

     nickname VARCHAR(255) NOT NULL,

     max_num_chat INT UNSIGNED,
```

```
    PRIMARY KEY(expert_id,group_id),
    UNIQUE(nickname),
    FOREIGN KEY(expert_id) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(group_id) REFERENCES Groups(group_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);


CREATE TABLE IF NOT EXISTS Performs
(
    user_id INT UNSIGNED NOT NULL,
    expert_id INT UNSIGNED NOT NULL,
    question_id INT UNSIGNED NOT NULL,
    PRIMARY KEY(user_id,expert_id,question_id),
    FOREIGN KEY(user_id) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(expert_id) REFERENCES Is_member_of(expert_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
```

```
    FOREIGN KEY(question_id) REFERENCES Question_answer(q_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Group_Actions
(
    group_id INT UNSIGNED NOT NULL,
    action_time DATETIME NOT NULL,
    action_type
ENUM('New_Member','New_Entry','New_Dictionary_Expert') NOT NULL,
    expert_id INT UNSIGNED NOT NULL,
    PRIMARY KEY(group_id,action_time,action_type,expert_id),
    FOREIGN KEY(group_id) REFERENCES Groups(group_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(expert_id) REFERENCES Is_member_of(expert_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Is_Friend_With
(
```

```
    user_id1 INT UNSIGNED NOT NULL,
    user_id2 INT UNSIGNED NOT NULL,
    since DATETIME NOT NULL,
    PRIMARY KEY(user_id1,user_id2),
    FOREIGN KEY(user_id1) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(user_id2) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Requests
(
    from_user INT UNSIGNED NOT NULL,
    to_user INT UNSIGNED NOT NULL,
    request_time DATETIME NOT NULL,
    PRIMARY KEY(from_user,to_user),
    FOREIGN KEY(from_user) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(to_user) REFERENCES Users(user_id)
```

```
        ON DELETE CASCADE

        ON UPDATE CASCADE

);


CREATE TABLE IF NOT EXISTS User_Actions

(

     user_id INT UNSIGNED NOT NULL,

     action_time DATETIME NOT NULL,

     action_type
ENUM('New_Friend','New_Friend_Request','Friend_Request_Accept','Fr
iend_Request_Decline','New_Message'),

     /*extendible with a new attribute action_refer which returns
an 'id' and evaluated with action_type directions:

     action_type   ENUM('Joined   group','Created   Group','Gained
dictionary   access','Friend   with','Application   Added','Created
Application'),

     action_refer INT UNSIGNED NOT NULL,

     */

     other_user_id INT UNSIGNED NOT NULL,

     PRIMARY KEY(user_id,action_time, action_type,other_user_id),

     FOREIGN KEY(user_id) REFERENCES Users(user_id)

        ON DELETE CASCADE

        ON UPDATE CASCADE,

     FOREIGN KEY(other_user_id) REFERENCES Users(user_id)

        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Images
(
     image_id INT UNSIGNED AUTO_INCREMENT,
     user_id INT UNSIGNED NOT NULL,
     user_image BLOB NOT NULL,
     upload_time DATETIME NOT NULL,
     PRIMARY KEY(image_id),
     FOREIGN KEY(user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Develops
(
     user_id INT UNSIGNED NOT NULL,
     api_id INT UNSIGNED NOT NULL,
     PRIMARY KEY(user_id,api_id),
     FOREIGN KEY(api_id) REFERENCES Applications(api_id)
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE,
    FOREIGN KEY(user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Uses
(
    user_id INT UNSIGNED NOT NULL,
    api_id INT UNSIGNED NOT NULL,
    PRIMARY KEY(user_id,api_id),
    FOREIGN KEY(api_id) REFERENCES Applications(api_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE IF NOT EXISTS Messages
(
    message_id INT UNSIGNED AUTO_INCREMENT,
```

```
    from_user INT UNSIGNED NOT NULL,
    to_user INT UNSIGNED NOT NULL,
    title VARCHAR(255) NOT NULL,
    text VARCHAR(255) NOT NULL,
    time DATETIME NOT NULL,
    reply_id INT UNSIGNED DEFAULT NULL,
    PRIMARY KEY(message_id),
    FOREIGN KEY(from_user) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(to_user) REFERENCES Users(user_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE,
    FOREIGN KEY(reply_id) REFERENCES Messages(message_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

/*-----------DICTIONARY DATABASE CREATION QUERRIES------------*/
use Dictionary_db;
CREATE TABLE IF NOT EXISTS Titles
(
```

```
    title_id INT UNSIGNED AUTO_INCREMENT,
    title VARCHAR(255) NOT NULL,
    creation_time DATETIME NOT NULL,
    PRIMARY KEY(title_id),
    UNIQUE(title)
);

CREATE TABLE IF NOT EXISTS Entries
(
    title_id INT UNSIGNED NOT NULL,
    text VARCHAR(255) NOT NULL,
    write_time DATETIME NOT NULL,
    expert_id INT UNSIGNED NOT NULL,
    group_id INT UNSIGNED NOT NULL,
    PRIMARY KEY(title_id,text),
    FOREIGN KEY(title_id) REFERENCES Titles(title_id)
       ON DELETE CASCADE
       ON UPDATE CASCADE
);

/*------------DOCUMENTS DATABASE CREATION QUERRIES------------*/
use Documents_db;
```

```
CREATE TABLE IF NOT EXISTS Documents

(

     document_id INT UNSIGNED AUTO_INCREMENT,

     upload_time DATETIME NOT NULL,

     doc BLOB,

     user_id INT UNSIGNED NOT NULL,

     num_downloads INT UNSIGNED NOT NULL DEFAULT 0,

     is_shared ENUM('All','Friends_Only','No') DEFAULT 'No',

     PRIMARY KEY(document_id)

);
```

Sample:

```
/*-----------A sample by Caner-----------------*/

use user_db;

INSERT                                                          INTO
Users(user_name,user_surname,birthday,birthplace,mail_address,sex,
city,state,country,phone_number)

VALUES("Caner","Kavakoglu",'1986-01-17',"Izmir","lalaith_bal@yahoo
.com","M","Ankara","ODTU","TR","0536*******");



INSERT                                                          INTO
Users(user_name,user_surname,birthday,birthplace,mail_address,sex,
city,state,country,phone_number)

VALUES("Yagiz","Kargin",'1986-05-25',"Nigde","xerxes862@yahoo.com"
,"M","Ankara","ODTU","TR","0505*******");
```

```
INSERT INTO Messages(from_user,to_user,title,text,time)

VALUES(1,2,"GUI?","GUI nasil gidiyor yagiz anasayfayi bitiremedin mi daha?",NOW());


INSERT INTO Messages(from_user,to_user,title,text,time,reply_id)

VALUES(2,1,"Re:GUI?","Bitti bitti cok guzel oldu super oldu!",NOW(),1);
```