MIDDLE EAST TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

# INITIAL DESIGN REPORT

FALL 2007

WEBMES

ASSISTANT: ÇAĞATAY ÇALLI

# AQUT

## Anatolian QUalified Technology


| | |
|---|---|
| Mehmet Ali Özkeskin | e1395375 |
| Mustafa Çöçelli | e1394840 |
| Uğur Irmak | e1347558 |
| Şevket Dokgöz | e1347384 |

# Table of Contents

# 1. INTRODUCTION

During the requirement analysis phase of our project, we have examined enough number of products in detail that have some similar base parts. Examining these AJAX based, Web 2.0 applications, helped us to determine advantageous and disadvantageous parts of them. By the experiences that we have gained while we were making extensive web search, we are now aware of required basic functionalities and extra features that we can add to our product, XILENT. Even though, developing a messaging environment via plug-in is not a widespread study nowadays, by making deep research, we got the knowledge to prepare our solutions to overcome all kind of problems that we may face.

This initial design report has the purpose of giving initial solutions especially on design concepts. These design concepts were analyzed in 4 main parts namely; data design, architectural design, interface design and procedural design. We have shaped database structure, used data flow diagrams, sequence diagrams and activity diagrams to make every part of the project clear. At this point, our analysis report helped us to shape all of these.

## 1.1. PROJECT TITLE

Our project title is “XILENT”. Because our project is developing a plug-in with many specialties, it will do its job silently, without warning the browser. So we think that, a word that has a similar pronunciation with silent will be suitable to be the title of our project.

## 1.2. PROBLEM DEFINITION

As the web technology evolves, people start to spend most of their times on the internet. Internet is being used as communication and information gathering environment at any time. In other words, people now socialize on the internet. Like in the real world, it becomes an important issue to bring people together on the web. On the other side, giving people information about what they need as quick as possible becomes another important issue. According to these needs, one of the most important necessities of the internet is now an

instant messaging platform that is independent from the website with question answering agent. So our aim is to develop such an application.

During this year we will be working on developing a collaborative web messaging environment based on Ajax and XMPP open platforms in Web 2.0 form. At user registration, we will use APACHE_TOMCAT server which can execute JSP files. So user information will be sent to the database by this way. On the server side of web messaging, OPENFIRE, a JABBER server will be used and on the client side, AJAX will be used. While OPENFIRE deals with instant messaging protocols, AJAX based user interfaces will make things work faster on the client side. By the help of these technologies we are going to develop Firefox/Internet Explorer plug-in that provides users, on-page messaging environment. With these plug-in you can chat with anybody that has added the plug-in to their browsers. But we will give communication chance to people that are visiting the same web pages. Moreover this project will have question answering (QA) side. But instant messaging is the first issue that we need to handle. At this point OPENFIRE really makes our job easier as we don't need to care too much about instant messaging mechanism because OPENFIRE will do it for us most of the time. We only have to account for creating some plug-in to help OPENFIRE to overcome these messaging protocols.

On the other hand, our extension will be AJAX based. A toolbar will appear at the left side of our browser and this application provides users on-page messaging environment. But this part will have extra properties. One of them is, this extension will have the ability to take many information from the webpage such as which webpage we are visiting at that moment, whether the webpage includes any tagged information or not and so on. This part will be achieved by the appropriate Java Scripts that we are going to develop. The other one is that this part will also have a simple question answering agent in it that has the ability of providing users much information about visited webpage. That is, if a user is interested in something on that side, then the agent should answer his/her question about the issue interested on that page. QA agent may also use user tagged information on the webpage to achieve its goal. The thing that we should keep in our minds is that QA is requiring more complex natural language processing (NLP) techniques than other information retrieval techniques.

In addition to these core requirements some more developments can be integrated into the project. For instance a user will have the ability to leave notes on a webpage for himself/herself for his/her friends or for everybody. This feature can help a user to remember to

look something on that site later. If note is left for a friend, this friend can see the note when he/she visit that page or the note can reach that friend immediately. Also the user can tag any information at the webpage to use it later, to share it with friends. In here we don't mean tagging the entire page, any part of the webpage can be tagged. Measuring the rating of web pages by looking which pages are visited by users will also be added to the project. This shows users, the web pages mostly visited in a day. This feature may help users to indicate the web pages that are popular. Moreover, a user can vote for a web page. This feature gives a chance to users for evaluating web pages. All users can see the average note of that web page. Also users can see the web pages that are being visited by his/her friends at that time. Therefore he/she can visit that page and establish a contact with his/her friend on that page. These are some of the extensions that we are planning to develop.

By developing such plug-in, we are planning to satisfy many web users' needs, because we think that only a webpage like facebook will not meet the socialization requirements of users, at the future this process should be done independent from a webpage.

## 1.3. PROJECT SCOPE & GOALS

XILENT will offer web users, a user friendly instant messaging toolbar together with question answering agent with rich user interface. We will develop a Web 2.0 application which is independent from the webpage that is visited at that moment. Our toolbar will be placed at the left side of the browser and provide basically instant messaging and many other functionalities. XILENT will be developed regarding the following basic properties:

- User friendliness: Xilent will be an easy application to use and its features will be both clear enough and understandable with user interface
- Security: If a user is looking his/her mails in a password protected area other users can not able to follow this user and also our application will not have any right to reach the information in this area
- AJAX: Full integration of AJAX technology will be achieved on user side
- Database: Our database will have a recovery feature in case of a database failure

- Fastness: XILENT will be capable of sending and receiving messages very fast by the power of OPENFIRE and won't affect browsers speed
- Extendibility: By using modular design and keeping the degree of coherence of modules low, any change can be integrated to XILENT with less amount of effort

## 1.4. CURRENT STATUS OF THE PROJECT

In this part, what has been done so far about the project and what can be done in near future will be told, because without any implementation, design issue will be abstract. This fact forces us to make an early start to implementation part. For this purpose we have designed a simple prototype. The applications in this prototype will surely be improved day by day by but this sample is very meaningful in the aspect of making a first step towards success. First we have developed a registration page to send the user information such as username and password to our MYSQL database server.

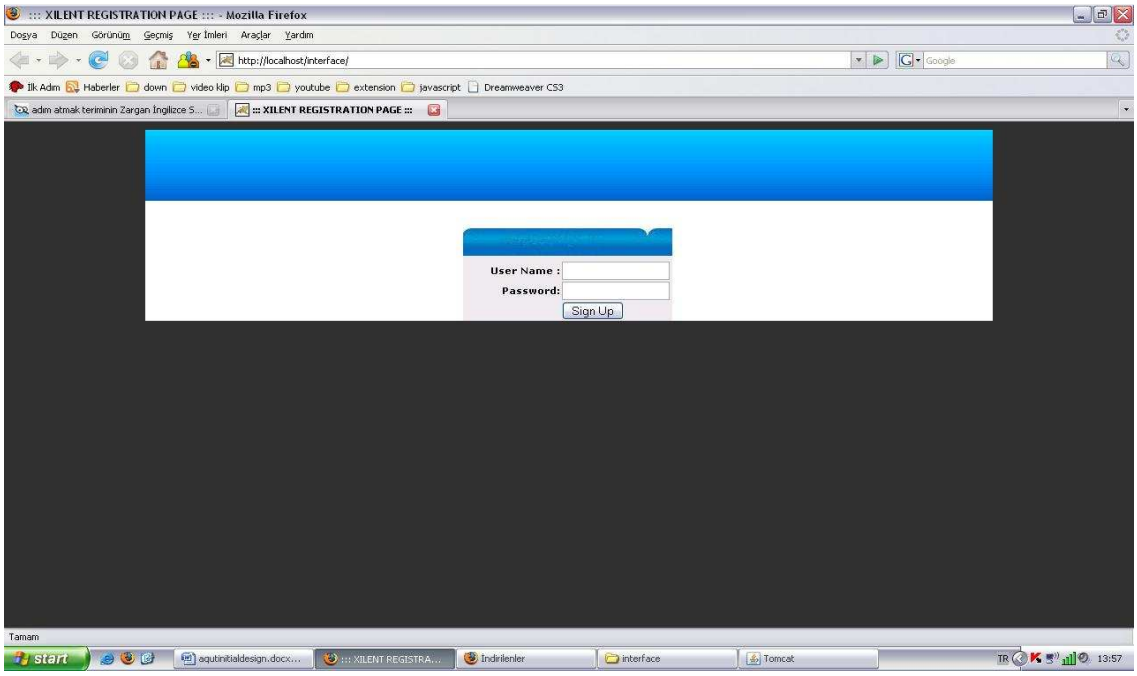



FIGURE 1.4.1: XILENT REGISTRATION PAGE

To signup a XILENT account, you have to type a unique user name otherwise you are redirected to this registration page. After a successful registration, you get the chance to download the plug-in. This plug-in will be located to the left side of your browser and have capabilities such as getting webpage title where you are currently visiting and directing the browser to some web pages.


FIGURE 1.4.2: XILENT EXTENSION AT THE LEFT SIDE

As we have mentioned above, we will always be developing the application. From now on, our first aim in near future is to establish XMPP connection from registration page to sent user information also to JABBER server's database. On the plug-in side we will integrate AJAX methodology and add much functionality which was told previously in the length of time.

# 2. DESIGN CONSTRAINTS & REQUIREMENTS

## 2.1. TIME CONSTRAINTS

Due to strictly set deadlines for this project it becomes undeniable to make an excellent scheduling analysis. Also our heavy course load is another factor that takes us under pressure. While preparing necessary reports for our project, we also have to work on the prototype. But in order to complete this project, we know we should meet the strict deadlines that were determined at the beginning of the term.

## 2.2. HARDWARE CONSTRAINTS & REQUIREMENTS

As we are going to develop an instant messaging environment, we need many computers within the context of our project especially in testing phase. This means we will need many computers both at the development side and server side. For the development of our project being problem free, these minimum requirements should be satisfied:

- Pentium IV or equivalent AMD processor
- At least 512 MB ram
- At least 40 GB hard disk
- Internet connection

## 2.3. SOFTWARE CONSTRAINTS & REQUIREMENTS

We will need many software and tools for developing our project which will help us at implementation phase, drawings and documentation phase. Since there are many tools that are widely being used, we think we can easily solve our problems related with software except plug-in development phase. After doing some research on the internet and talking with the experts

of the subject, we have seen that there is no plug-in development tool yet that can properly work, but there are some studies about it. So our plug-in development phases will be very time consuming. Because when we make any change on JavaScript and XUL files, we will generate XPI document by hand all the time. Moreover we need to work with CVS for faster development. In other words, coding separately and then combining the codes will increase the performance. On the other hand, our software requirements can be listed as:

- Apache Tomcat server
- JABBER/XMPP server, most probably Openfire
- Mozilla Firefox
- Java Development Software Kit and Java Runtime Environment
- Microsoft Project
- Smart Draw
- Microsoft Office and Adobe Professional
- Eclipse
- Dreamweaver

# 3. ARCHITECTURAL & DATA DESIGN

## 3.1. XILENT MODULES

Our design consists of seven modules. In this part these modules are going to be explained briefly but in class diagrams part of the report, the roles of these modules will be explained in more detailed way.

- ✓ USER MANAGEMENT MODULE

In this module, we are planning to handle user related issues. These processes are user signup and login, user profile update, note and tag deletion requests of the user. This module is about the capabilities of a user when he/she is at his/her account. Signup of a visitor and login

of a system user will be handled inside this module. Also if a user wants to make any change on his/her account information, make friends or add someone to blacklist, these requests of the user will be resolved within user management module. Finally, user management module is responsible for not leaving but deleting notes or tags.

- ✓ NOTE OPERATIONS MODULE

This module is responsible for performing user's note leaving or editing requests. Note operations module will send the note information to database. This data includes the node information of the leaved note, webpage and owner information and the note itself. This data will be inserted to the database.

- ✓ TAG OPERATIONS MODULE

This module is responsible for performing user's tag leaving or editing requests. Tag operations module will send the tag information to database. This data includes webpage and owner information of the leaved tag and tag itself. This data will be inserted to the database.

- ✓ RATE MODULE

By this module, we are planning to provide user to rate the website that he/she is currently visiting. The webpage information and user rate pair will be inserted to the database through this module.

- ✓ INSTANT MESSAGING MODULE

In this module, we based our design on providing users an instant messaging environment by generating XML requests from users' messages and parsing the XML response that comes from JABBER server. For a general JABBER server, sender and receiver information

should be kept to make the process easy, but by a powerful JABBER server like OPENFIRE, even keeping sender and receiver info may not be necessary.

- ✓ QUESTION ANSWERING MODULE

Design of this module is based on simple question answering methodology. User will find the answers of his/her questions via this module. Firstly a question customizing will be done to be sure that our question answer agent understands the question well. After this process, possible answers will be searched from the database. At this point tag information will be helpful. When the answer is created, it is presented to user. In case of user satisfaction this question and its answer will be inserted to the database for future use.

- ✓ GUI MODULE

GUI module is responsible for users' easy use of the system and displaying process for a system user. Display process has actually two parts. One is plug-in side and the other is webpage side. On the plug-in side, interface for messaging environment will be presented to the user. This includes popular web pages and messaging range display, communication with question answering agent and other users. On the other hand, webpage side presents the interface for notes, tags and user account related issues.

## 3.2. DATA FLOW DIAGRAMS

### 3.2.1. DFD (LEVEL 0)

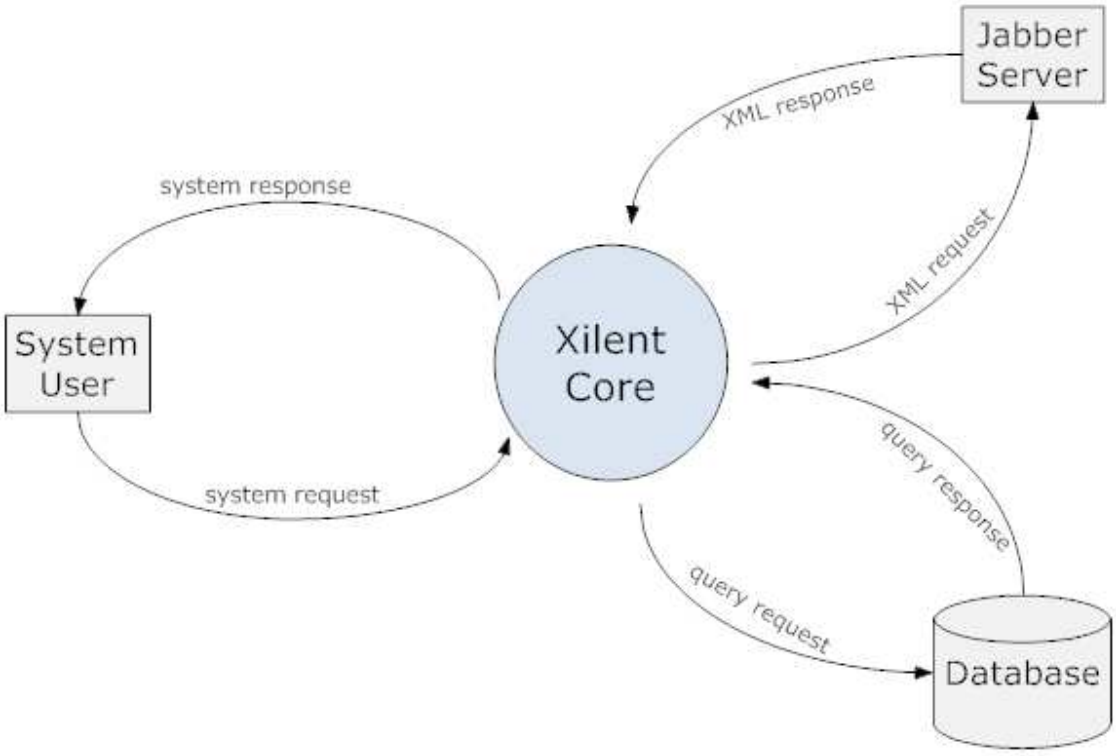



FIGURE 3.2.1.1: LEVEL 0 DFD

## 3.2.2. DFD (LEVEL 1)

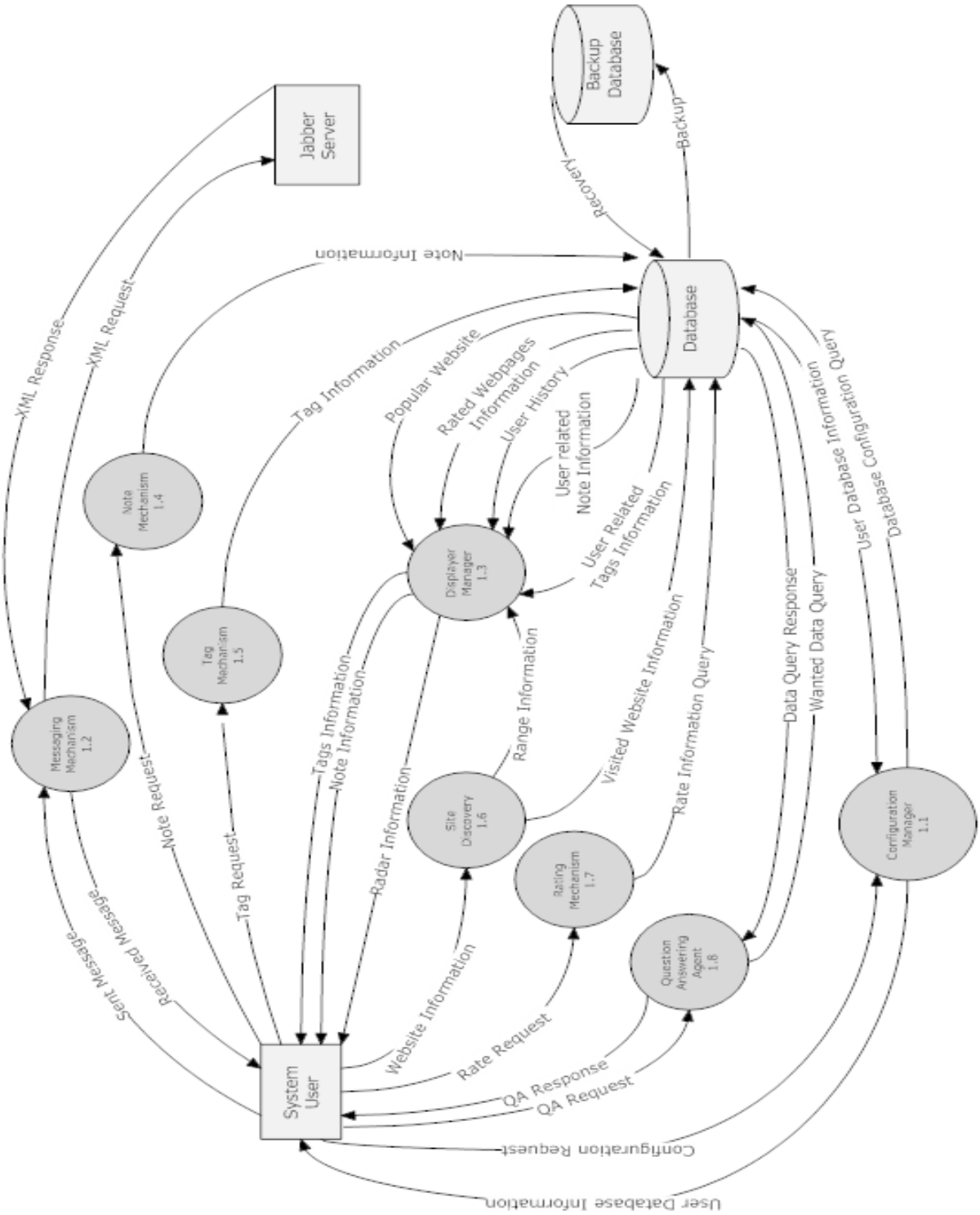



FIGURE 3.2.2.1: LEVEL 1 DFD

## 3.2.3. DFDs (LEVEL 2)

- 1.1 CONFIGURATION MANAGER

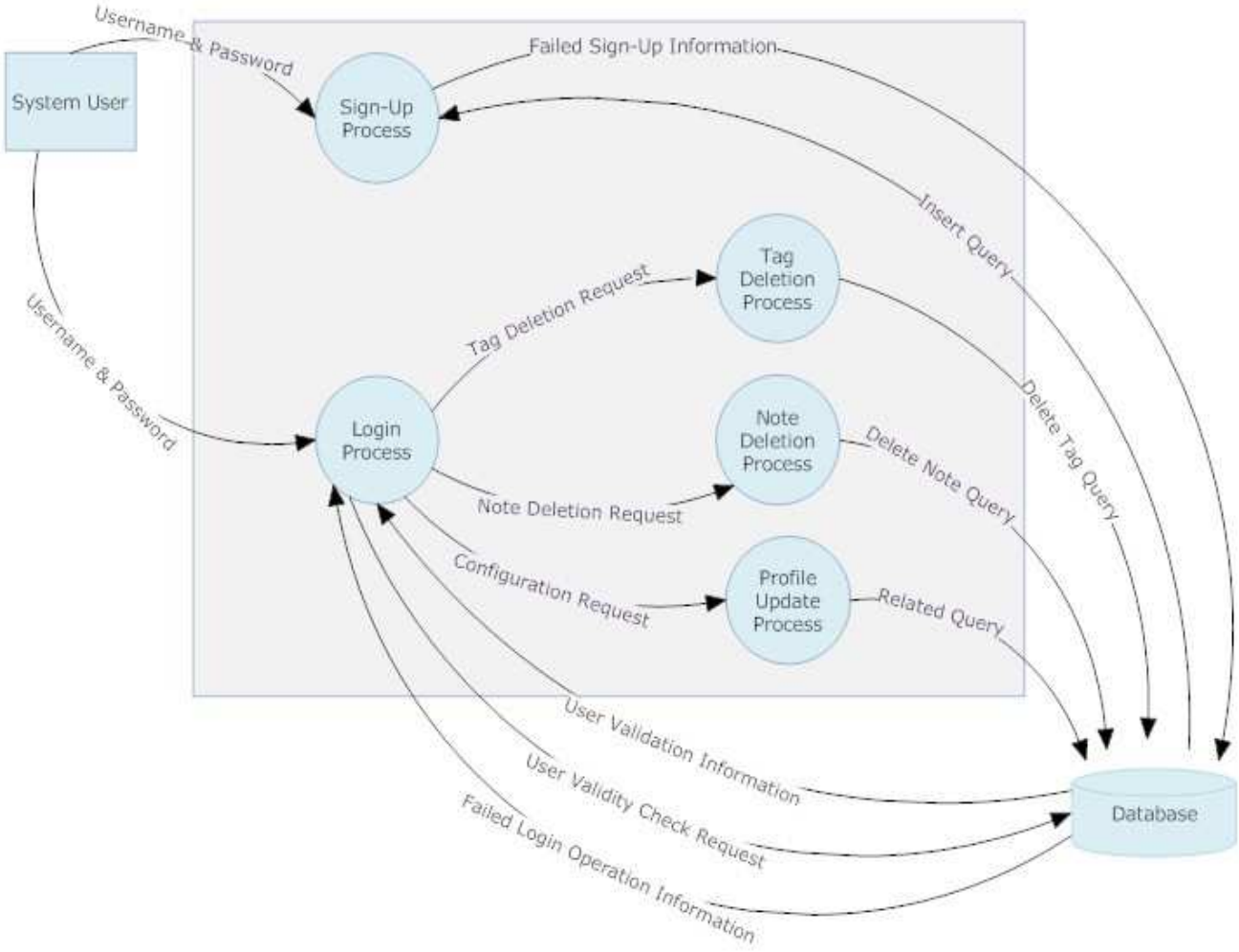



FIGURE 3.2.3.1: LEVEL 2 DFD FOR CONFIGURATION MANAGER

❖ 1.2 MESSAGING MECHANISM

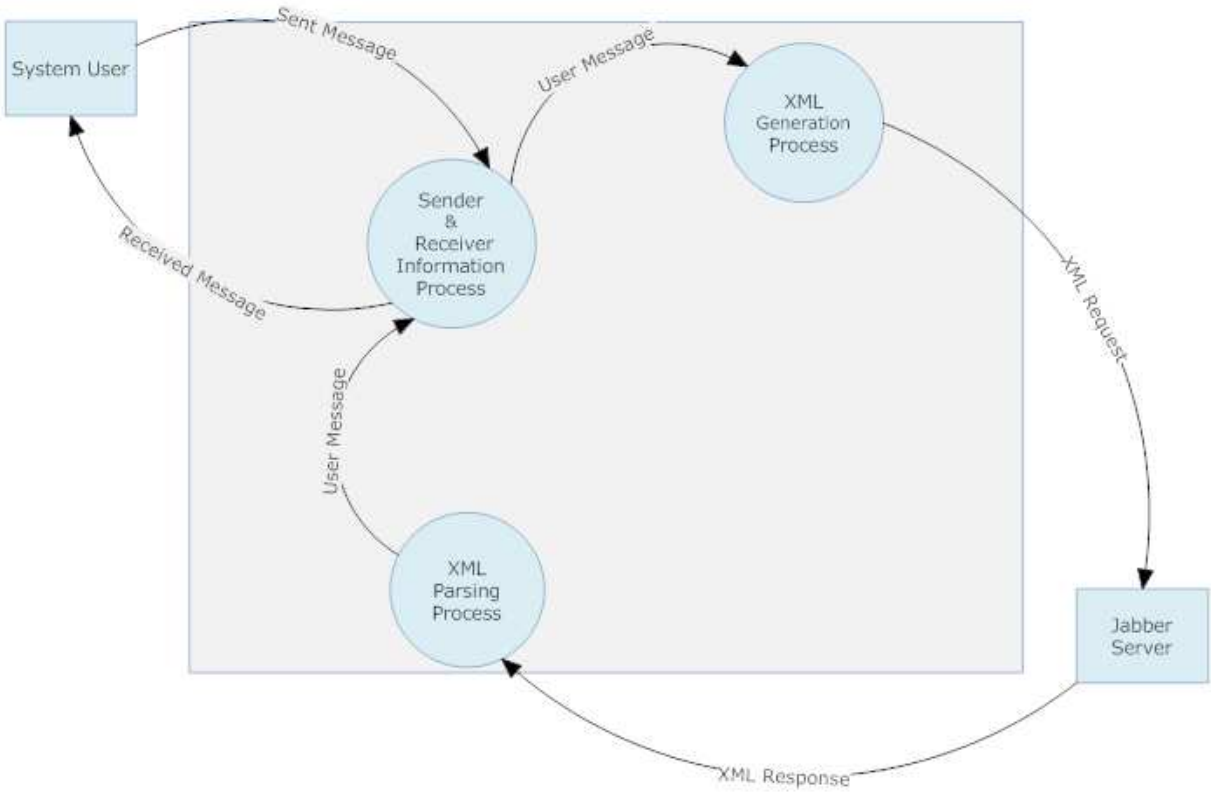



FIGURE 3.2.3.2: LEVEL 2 DFD FOR MESSAGING MECHANISM

❖ 1.3 DISPLAY MANAGER

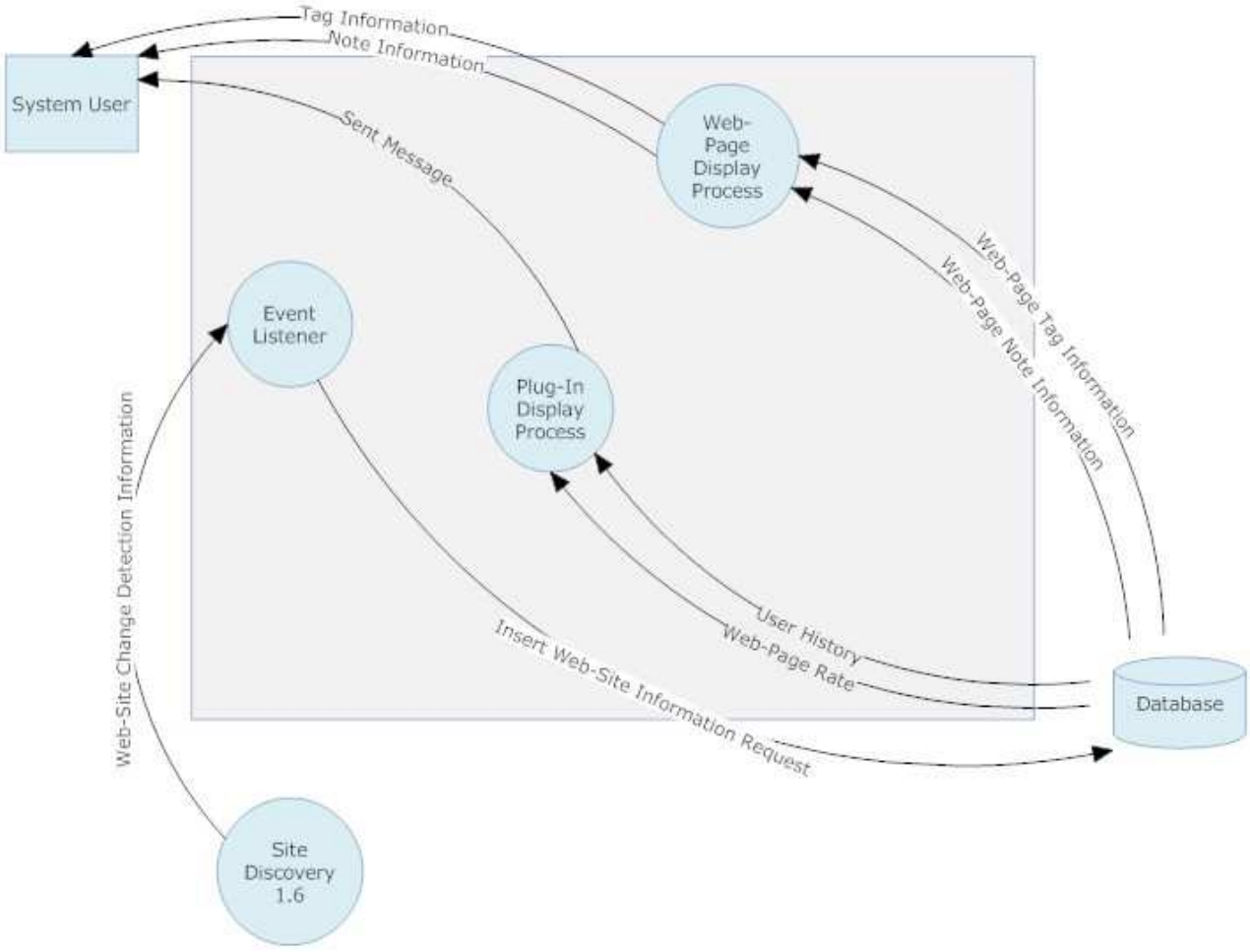



FIGURE 3.2.3.3: LEVEL 2 DFD FOR DISPLAY MANAGER

❖ 1.4 NOTE MECHANISM

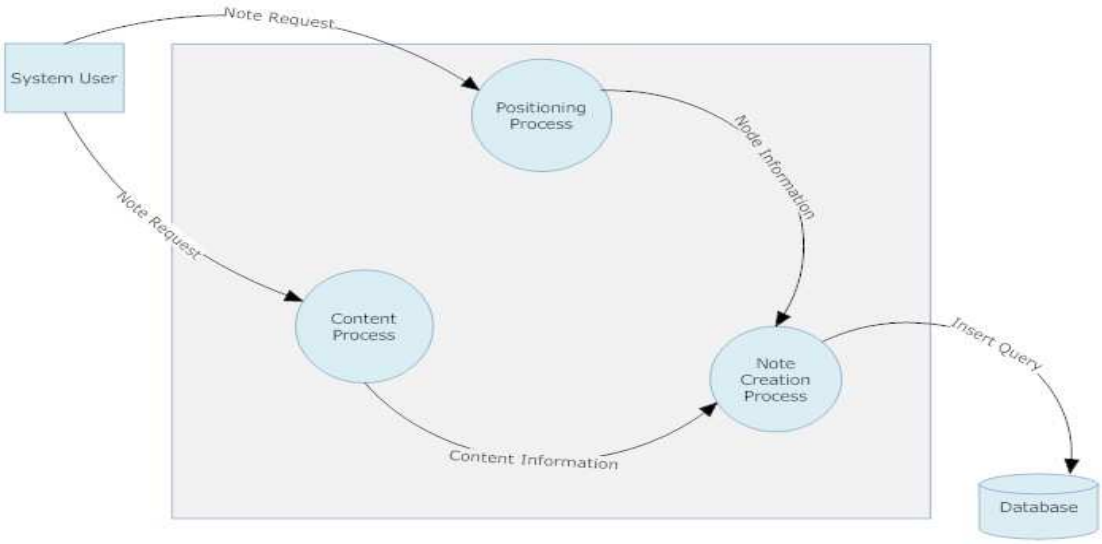



FIGURE 3.2.3.4: LEVEL 2 DFD FOR NOTE MECHANISM

❖ 1.5 TAG MECHANISM

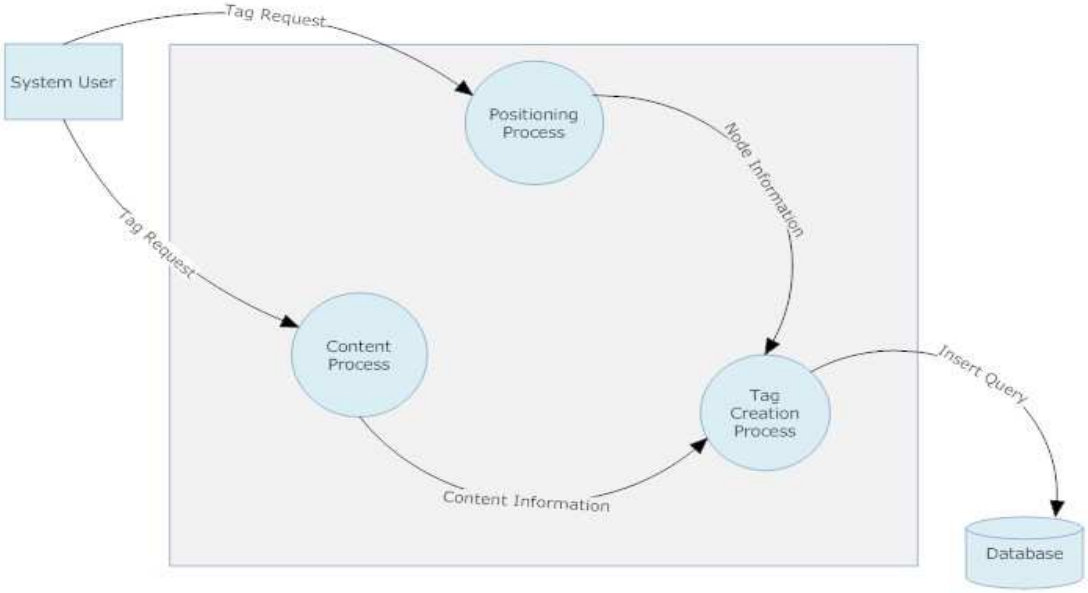



FIGURE 3.2.3.5: LEVEL 2 DFD FOR TAG MECHANISM

❖ 1.6 SITE DICOVERY

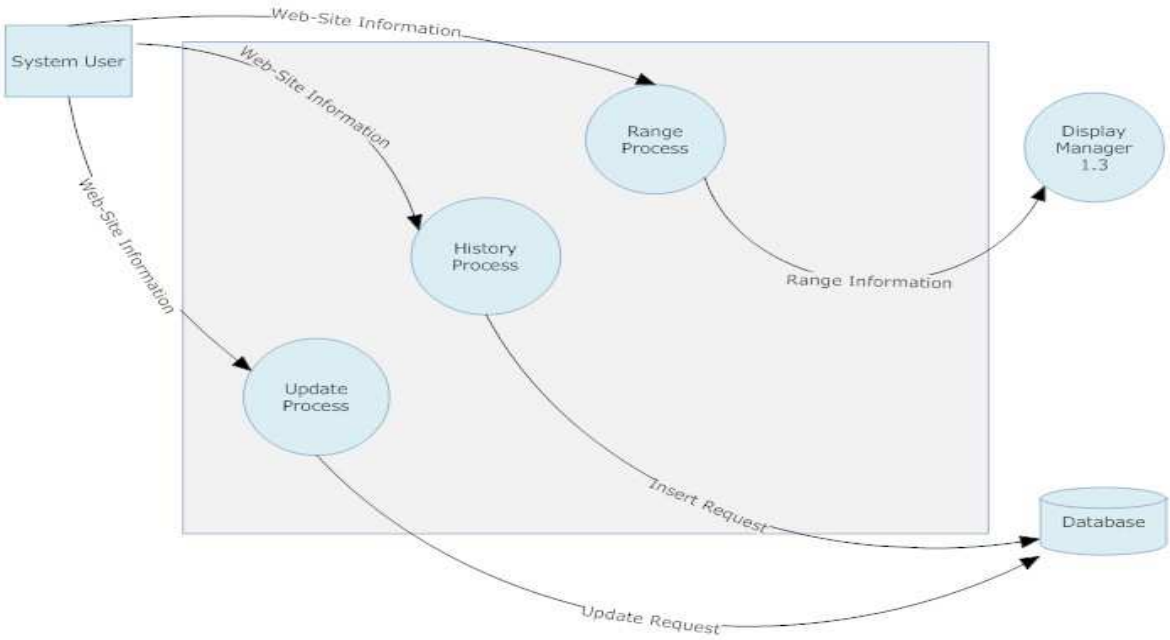



FIGURE 3.2.3.6: LEVEL 2 DFD FOR SITE DISCOVERY

❖ 1.7 RATING MECHANISM

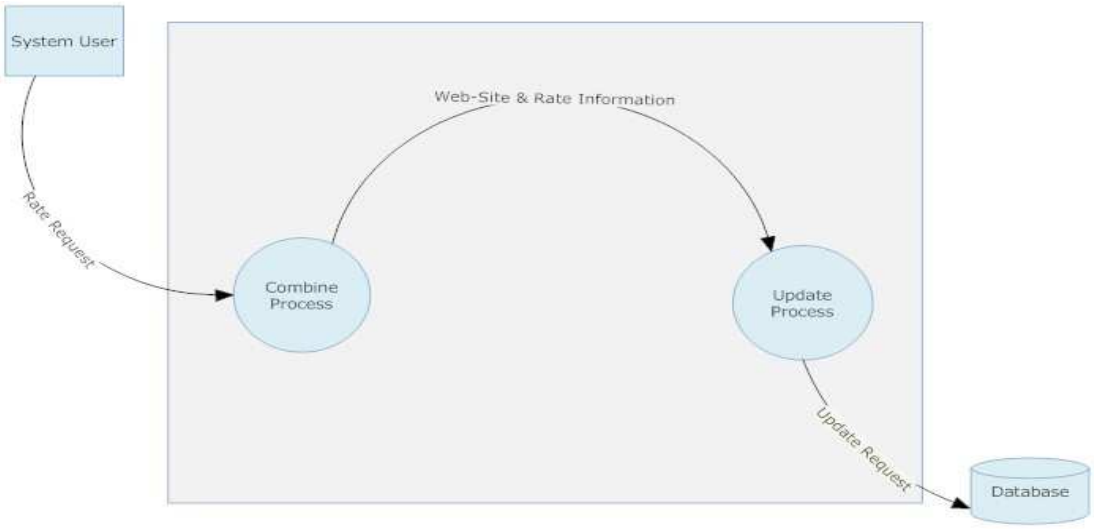



FIGURE 3.2.3.7: LEVEL 2 DFD FOR RATING MECHANISM

❖ 1.8 QUESTION ANSWERING AGENT

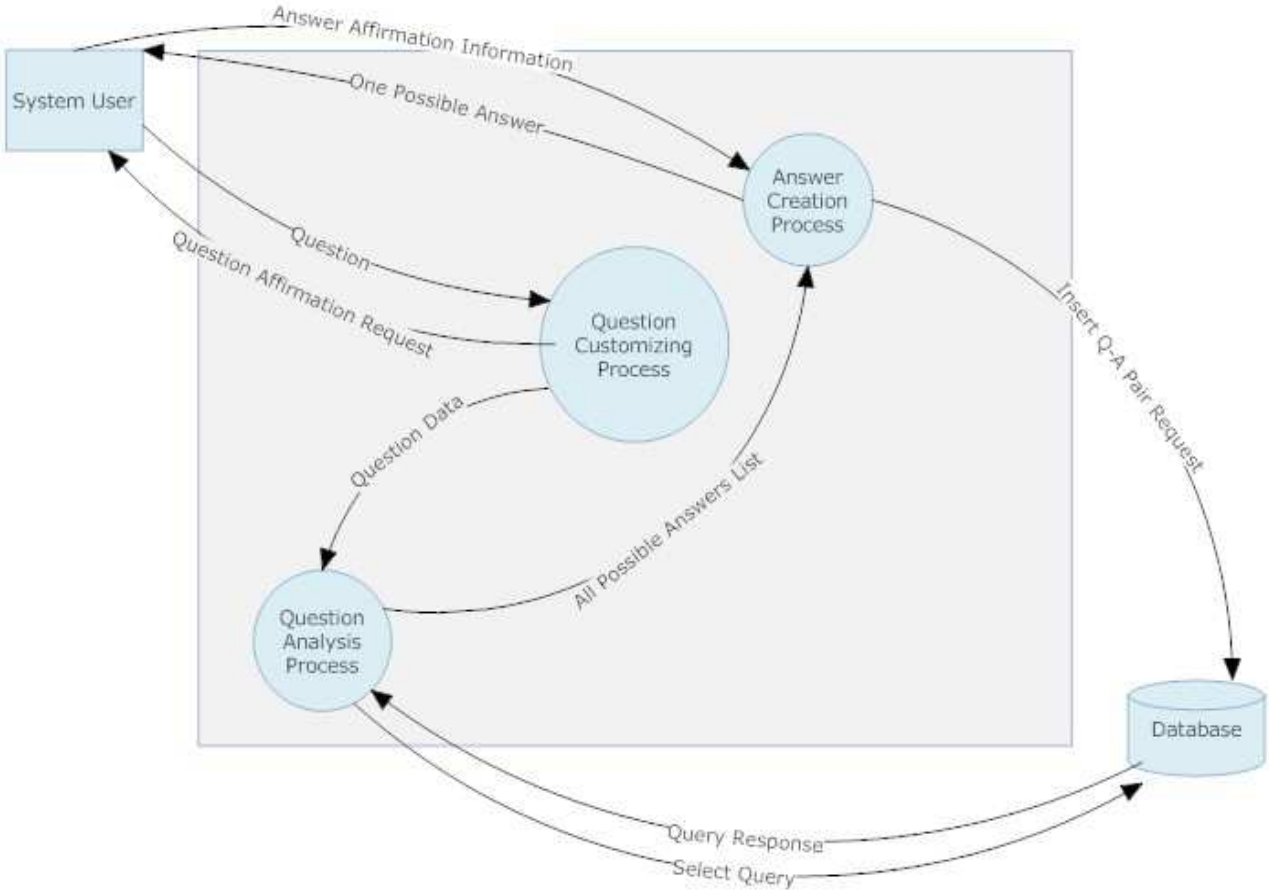



FIGURE 3.2.3.8: LEVEL 2 DFD FOR QA AGENT

## 3.3. DATA DICTIONARY

| Name | Sent Message |
|---|---|
| Where | From System User to  Message Mechanism 1.2 |
| Description | "collection of strings that are sent by System User" |

| Name | Received Message |
|---|---|
| Where | From Message Mechanism 1.2 to System User |
| Description | "collection of strings that are sent to System User" |

| Name | XML Request |
|---|---|
| Where | From Message Mechanism 1.2 to Jabber Server |
| Description | "XML data that contains System User's message" |

| Name | XML Response |
|---|---|
| Where | From Jabber Server to Message Mechanism 1.2 |
| Description | "XML data that contains message to System User" |

| Name | Note Request |
|---|---|
| Where | From System User to Note Mechanism 1.4 |
| Description | "data related with Sender System User, content of notes " |

| Name | Note Information |
|---|---|
| Where | From Tag Mechanism 1.5 to Database |
| Description | "data related with System User, content of note, attributes of note" |

| Name | Tag Request |
|---|---|
| Where | From System User to Tag Mechanism 1.5 |
| Description | "data related with Sender System User, content of tag" |

| Name | Tag Information |
|---|---|
| Where | From Tag Mechanism 1.5 to Database |
| Description | "data related with System User, content of tag, attributes of tag" |

| Name | Website Information |
|---|---|
| Where | From System User to Site Discovery 1.6 |
| Description | "website information visited by System User" |

| Name | Range Information |
|---|---|
| Where | From Site Discovery 1.6 to Display Manager 1.3 |
| Description | "information of websites that are in scope of System User" |

| Name | Visited Website Information |
|---|---|
| Where | From Site Discovery 1.6 to Database |
| Description | "visited web pages by System User " |

| Name | Rate Request |
|---|---|
| Where | From System User to Rate Mechanism 1.7 |
| Description | "rating that is given by System User" |

| | |
|---|---|
| Name | Rate Information Query |
| Where | From Rate Mechanism 1.7 to Database |
| Description | "database query that store rating information to database" |

| | |
|---|---|
| Name | QA Request |
| Where | From System User to QA Agent 1.8 |
| Description | "content of question asked by System User" |

| | |
|---|---|
| Name | QA Response |
| Where | From QA Agent 1.8 to System User |
| Description | "possible answer to question asked by System User" |

| | |
|---|---|
| Name | Wanted Data Query |
| Where | From QA Agent 1.8 to Database |
| Description | "database query that looks for answer to asked question" |

| | |
|---|---|
| Name | Data Query Response |
| Where | From Database to QA Agent 1.8 |
| Description | "possible answer to wanted data" |

| | |
|---|---|
| Name | Configuration Request |
| Where | From System User to Configuration Manager 1.1 |
| Description | "commands for updating profile, editing, deleting tags and notes " |

| | |
|---|---|
| Name | User Database Information |
| Where | From Configuration Manager 1.1 to System User |
| Description | "information about System User's profile" |

| | |
|---|---|
| Name | Database Configuration Query |
| Where | From Configuration Manager 1.1 to Database |
| Description | "database query that update profile, delete and edit tags and notes" |

| | |
|---|---|
| Name | User Database Information |
| Where | From Database to Configuration Manager 1.1 |
| Description | "information about System's User profile" |

| | |
|---|---|
| Name | Tags Information |
| Where | From Display Manager 1.3 to System User |
| Description | "content of tags" |

| | |
|---|---|
| Name | Note Information |
| Where | From Display Manager 1.3 to System User |
| Description | "content of note" |

| | |
|---|---|
| Name | Radar Information |
| Where | From Display Manager 1.3 to System User |
| Description | "data about website that are in scope of System User" |

| | |
|---|---|
| Name | Popular Website |
| Where | From Database to Display Manager 1.3 |
| Description | "data about most visited websites" |

| | |
|---|---|
| Name | Rated Webpage Information |
| Where | From Database to Display Manager 1.3 |
| Description | "rating of website" |

| | |
|---|---|
| Name | User History |
| Where | From Database to Display Manager 1.3 |
| Description | "data about websites that are recently visited by System's User" |

| | |
|---|---|
| Name | User Related Note Information |
| Where | From Database to Display Manager 1.3 |
| Description | "data about notes that are belong to System's User" |

| | |
|---|---|
| Name | User Related Tag Information |
| Where | From Database to Display Manager 1.3 |
| Description | "data about tags that are belong to System's User |

| | |
|---|---|
| Name | Backup |
| Where | From Database to Backup Database |
| Description | "tables of database and their content" |

| Name | Recovery |
|---|---|
| Where | From Backup  Database to Database |
| Description | "tables of backup database and their content" |

## 3.4. STATE TRANSITION DIAGRAM

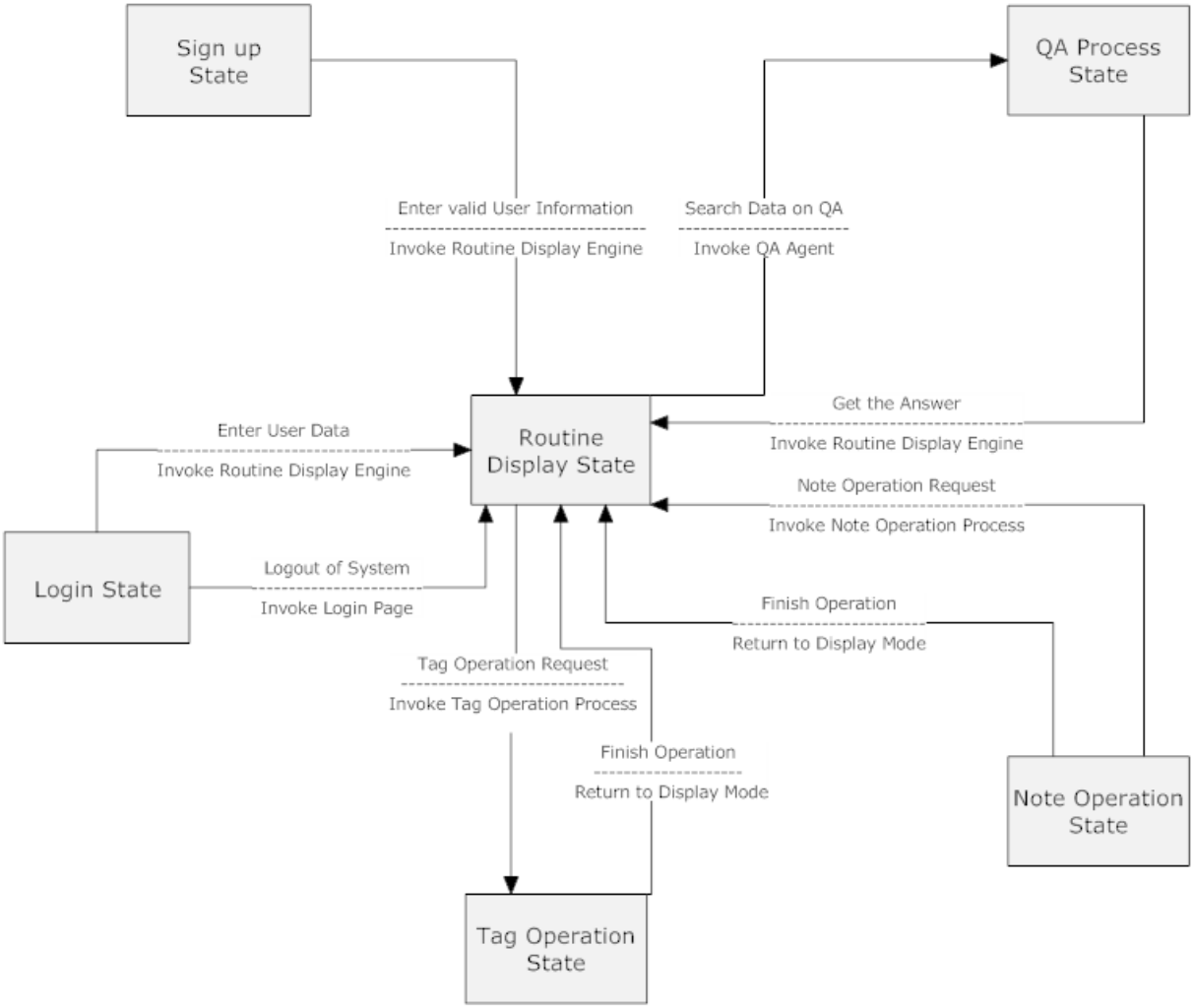



FIGURE 3.4.1: STATE TRANSITION DIAGRAM

In Xilent, web messaging and webpage discovery processes are always in progress for all system users, so these states are valid at any moment. In our state diagram, routine display state is considered to include these states. Others are signup, login, QA process, note operation and tag operation states.

## 3.5. ENTITY RELATIONSHIP DIAGRAMS

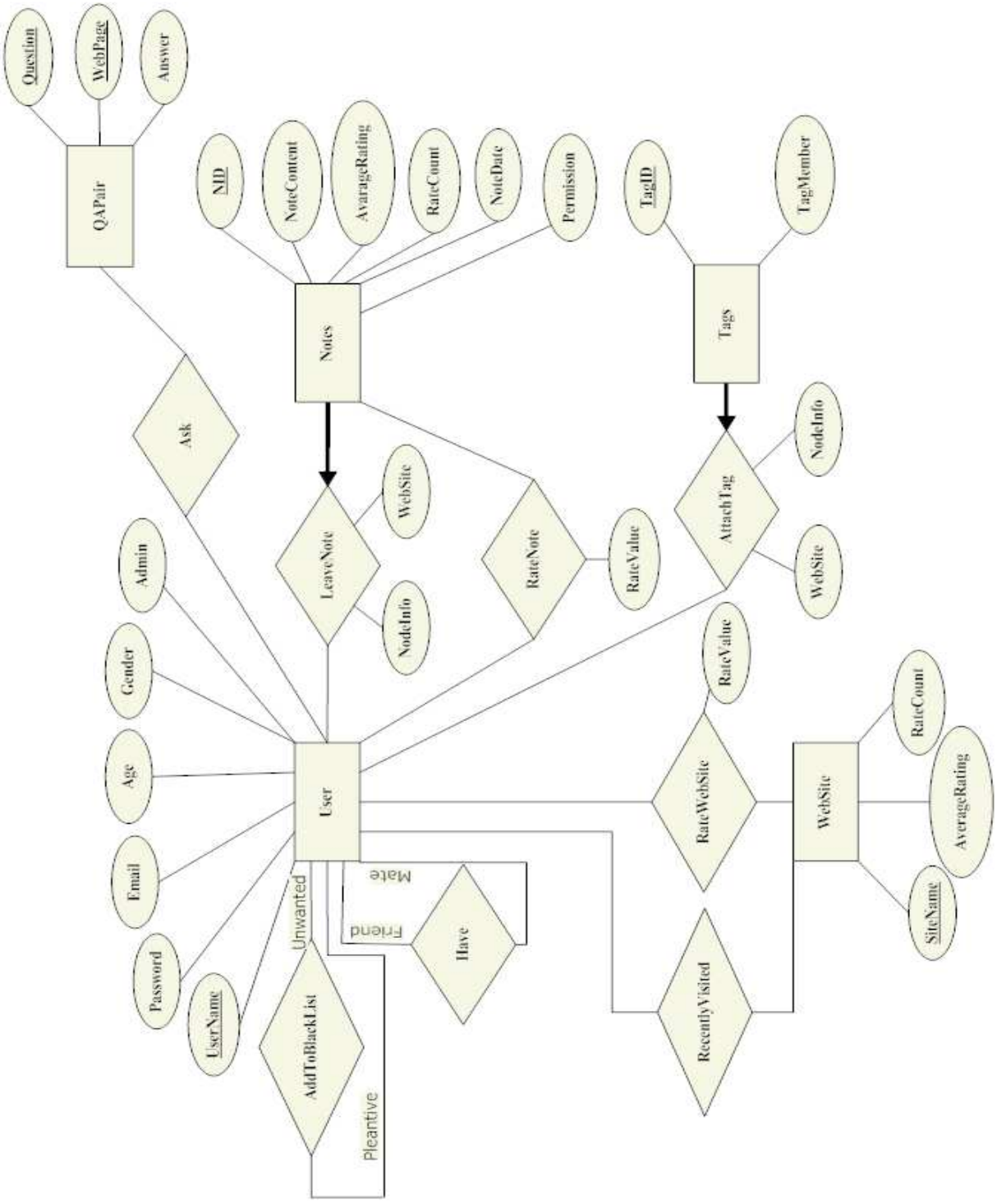



FIGURE 3.5.1: ER DIAGRAM

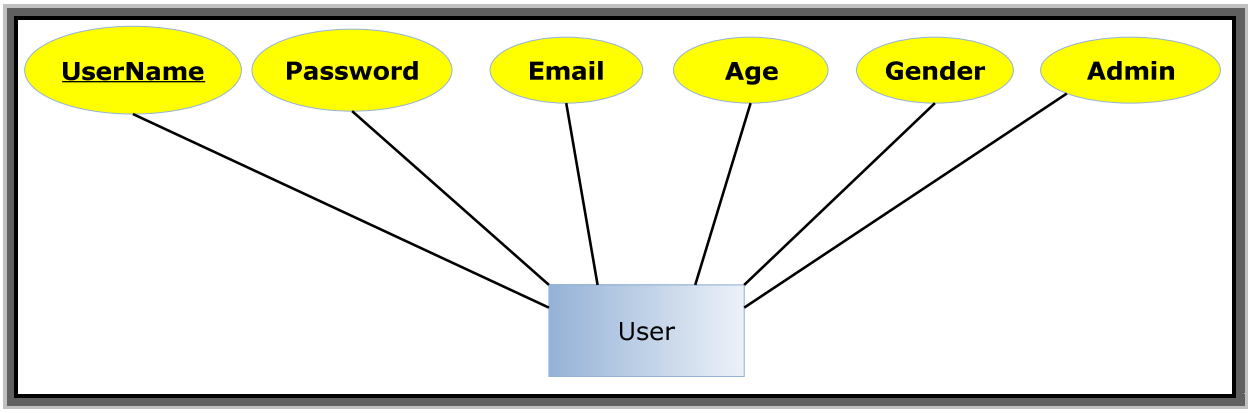



FIGURE 3.5.2: User ENTITY

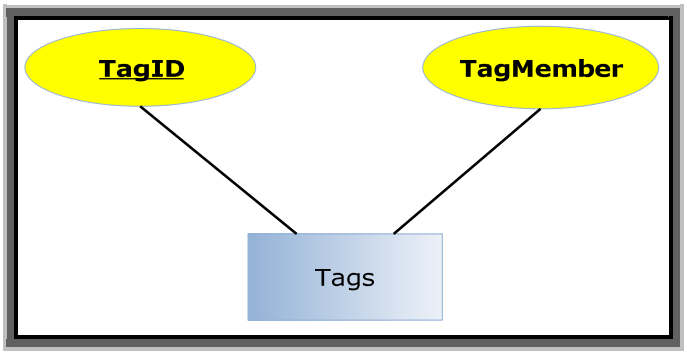



FIGURE 3.5.3: Tags ENTITY

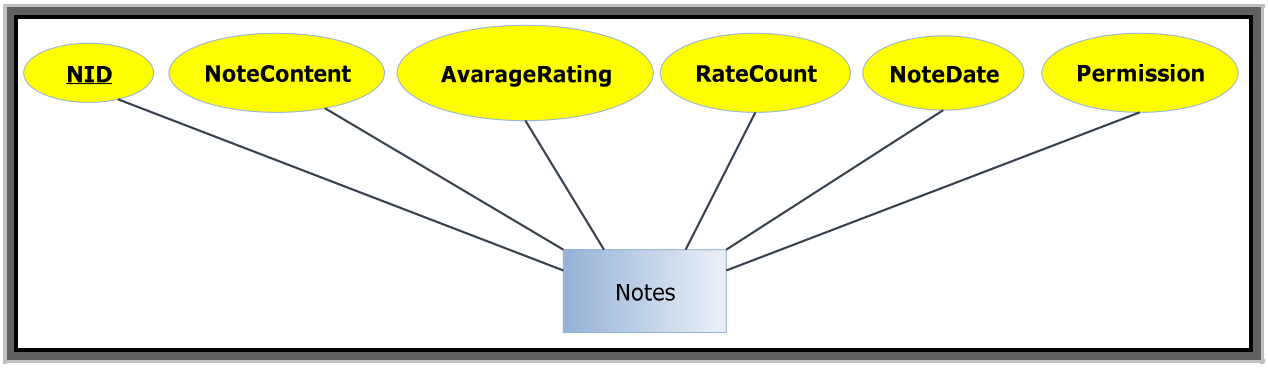



FIGURE 3.5.4: Notes ENTITY

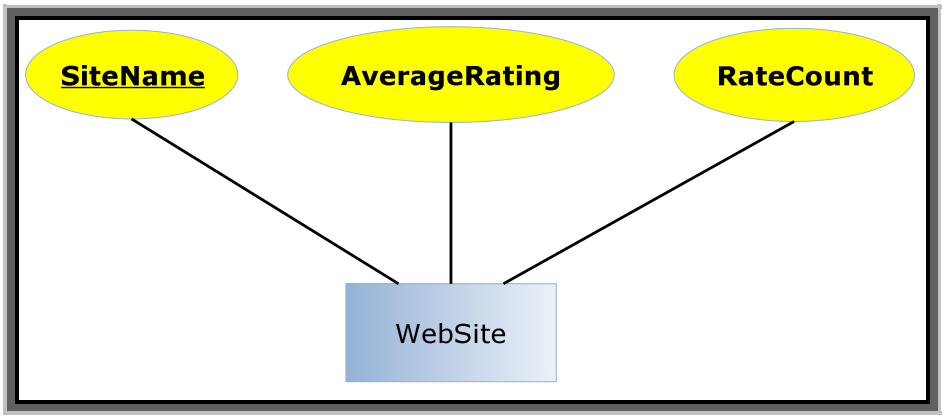



FIGURE 3.5.5: WebSite ENTITY

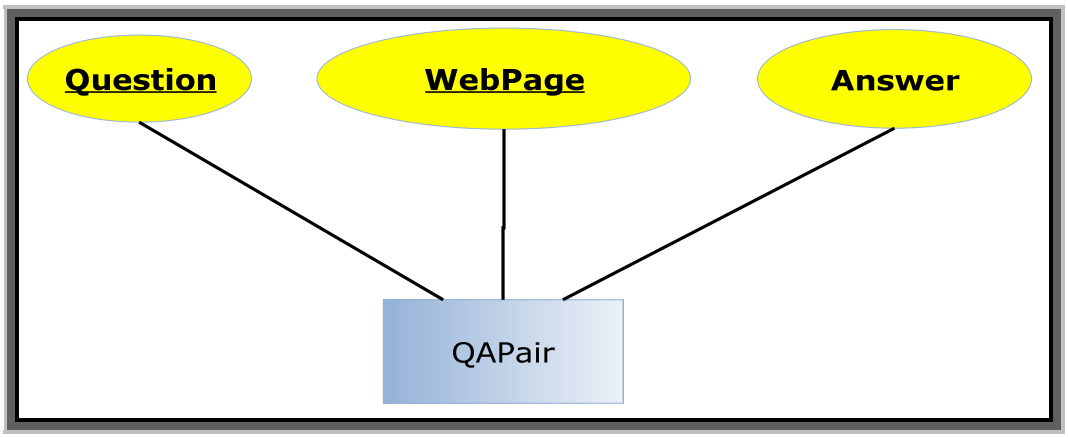



FIGURE 3.5.6: QAPair ENTITY

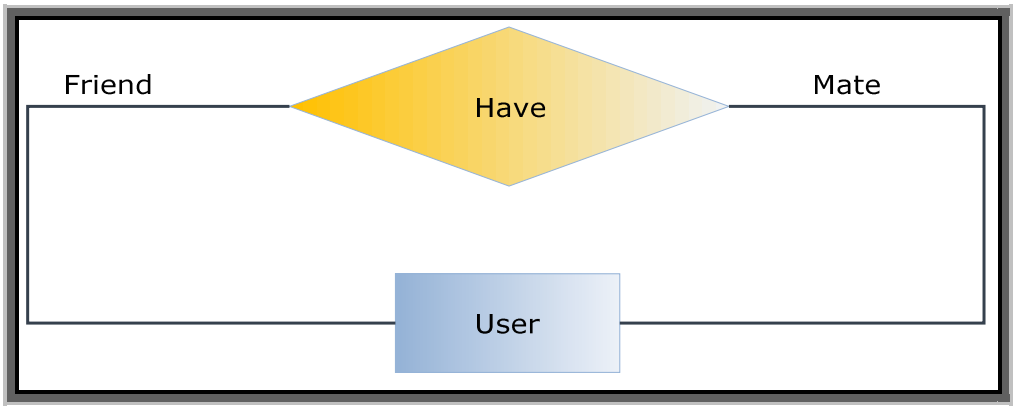



FIGURE 3.5.7: Have RELATION

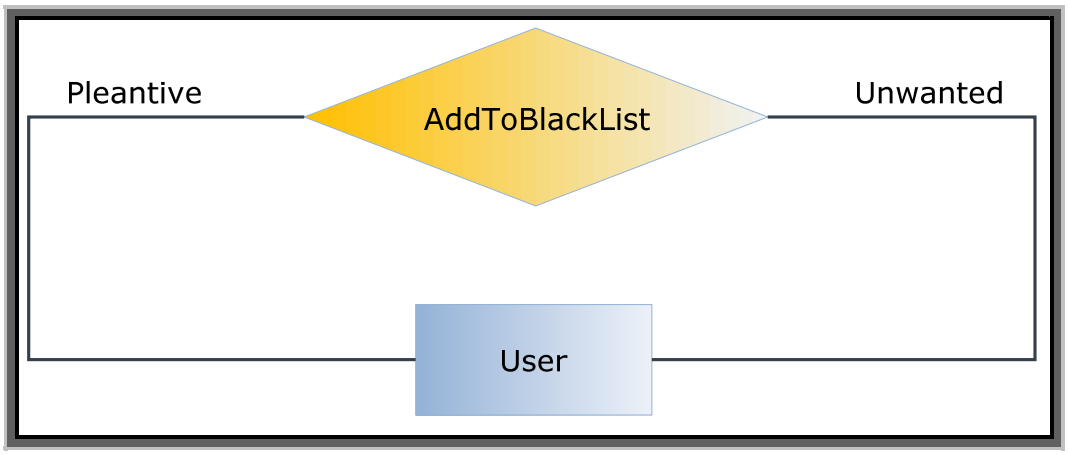



FIGURE 3.5.8: AddToBlackList RELATION

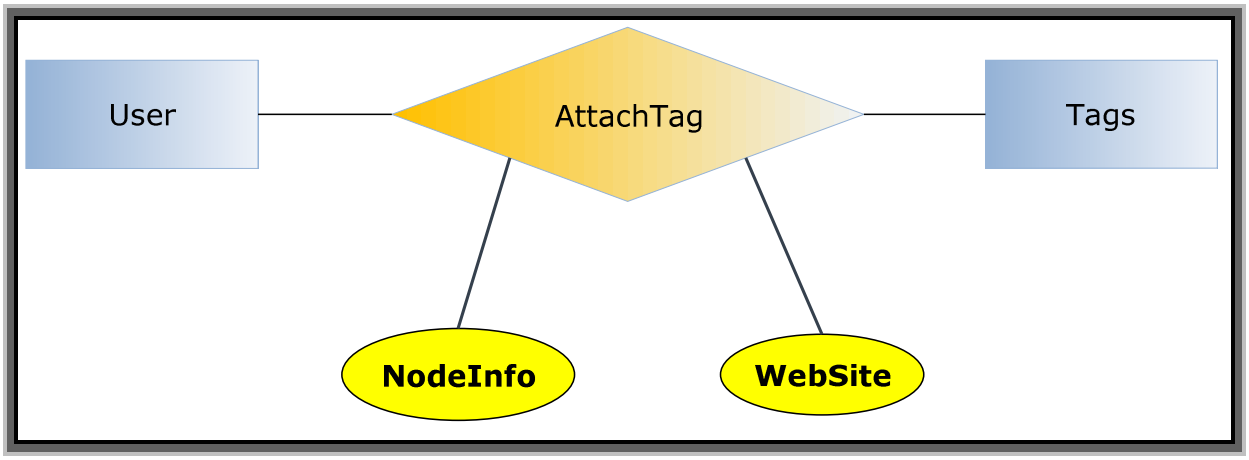



FIGURE 3.5.9: AttachTag RELATION

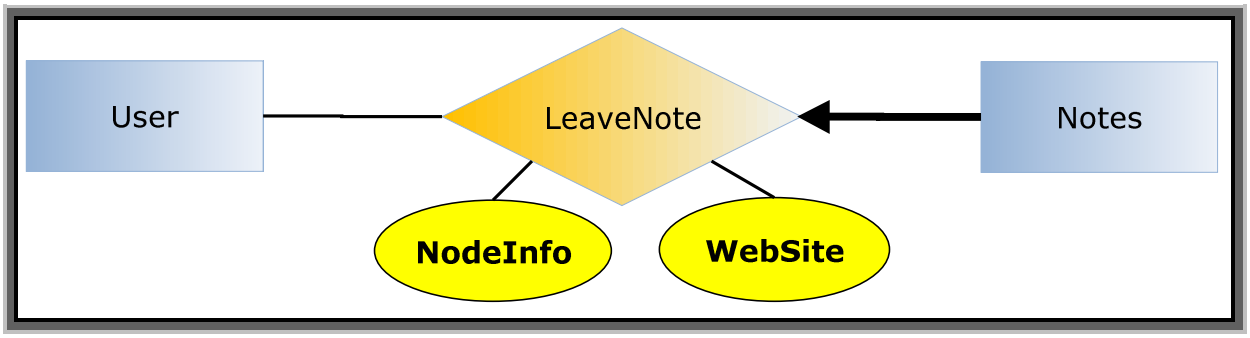



FIGURE 3.5.10: LeaveNote RELATION

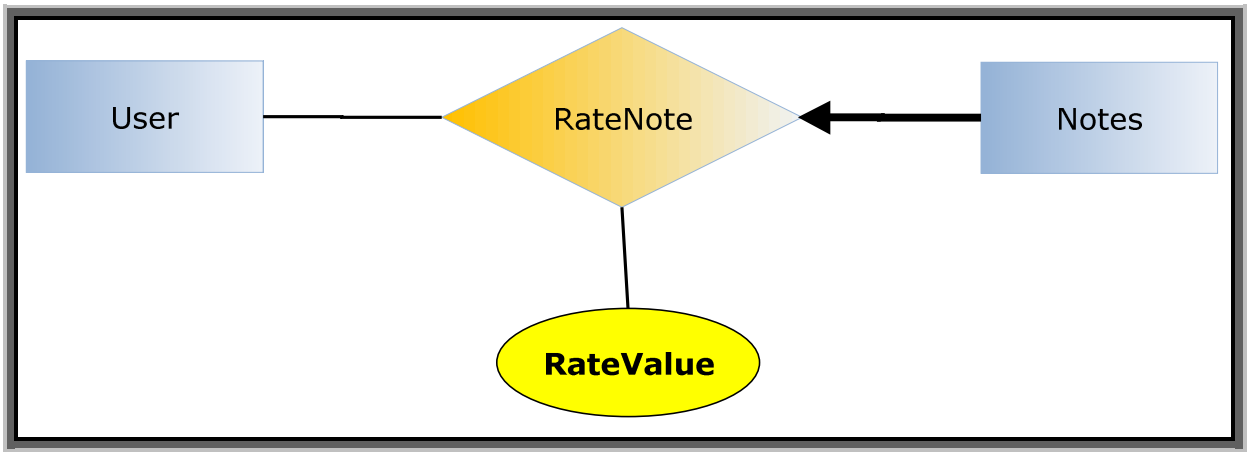



FIGURE 3.5.11: RateNote RELATION

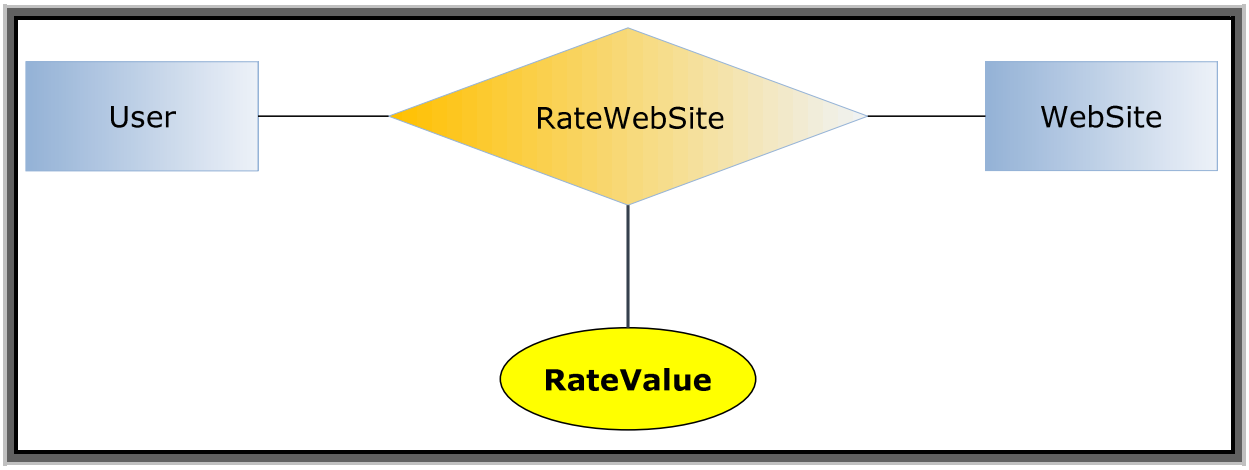



FIGURE 3.5.12: RateWebSite RELATION


FIGURE 3.5.13: RecentlyVisited RELATION

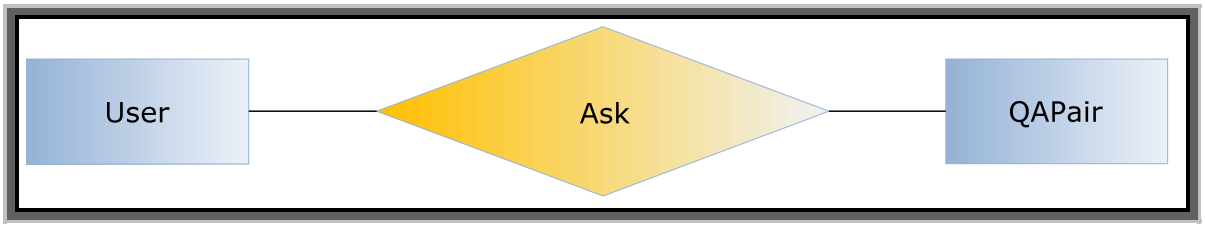



FIGURE 3.5.14: Ask RELATION

### 3.5.1. DATA DESCRIPTIONS

Attributes with "*" are can not take null value.

- User

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | Text |
| Password* | VARCHAR-20 | Text is hidden |
| Email* | VARCHAR-40 | Text |
| Age | INTEGER | Number |
| Gender | VARCHAR-10 | Text |
| Admin* | BOOLEAN | |

- Notes

| Data | Type & Size | Format |
|---|---|---|
| NID* | INTEGER | Number |
| NoteContent | VARCHAR-100 | Text |
| AverageRating | INTEGER | Number |
| RateCount | INTEGER | Number |
| NoteDate* | DATETIME | Date/time |
| Permission | VARCHAR-10 | Text |

- Tags

| Data | Type & Size | Format |
|---|---|---|
| TagID* | INTEGER | Number |
| TagMember* | VARCHAR-20 | Text |

- WebSite

| Data | Type & Size | Format |
|---|---|---|
| SiteName* | VARCHAR-50 | Text |
| AverageRating | INTEGER | Number |
| RateCount | INTEGER | Number |

- AddToBlackList

| Data | Type & Size | Format |
|---|---|---|
| Unwanted_UserName* | VARCHAR-20 | Text |
| Pleantive-UserName* | VARCHAR-20 | Text |

- LeaveNote

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | Text |
| NID* | INTEGER | Number |
| NodeInfo* | VARCHAR-40 | Text |
| WebSite* | VARCHAR-50 | Text |

- Have

| Data | Type & Size | Format |
|---|---|---|
| Friend-UserName* | VARCHAR-20 | Text |
| Mate-UserName* | VARCHAR-20 | Text |

- RecentlyVisited

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | Text |
| SiteName* | VARCHAR-50 | Text |

- RateWebSite

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | Text |
| SiteName* | VARCHAR-50 | Text |
| RateValue* | INTEGER | Number |

- AttachTag

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | text |
| TagID* | INTEGER | Number |
| WebSite* | VARCHAR- | Text |
| NodeInfo* | VARCHAR- | Text |

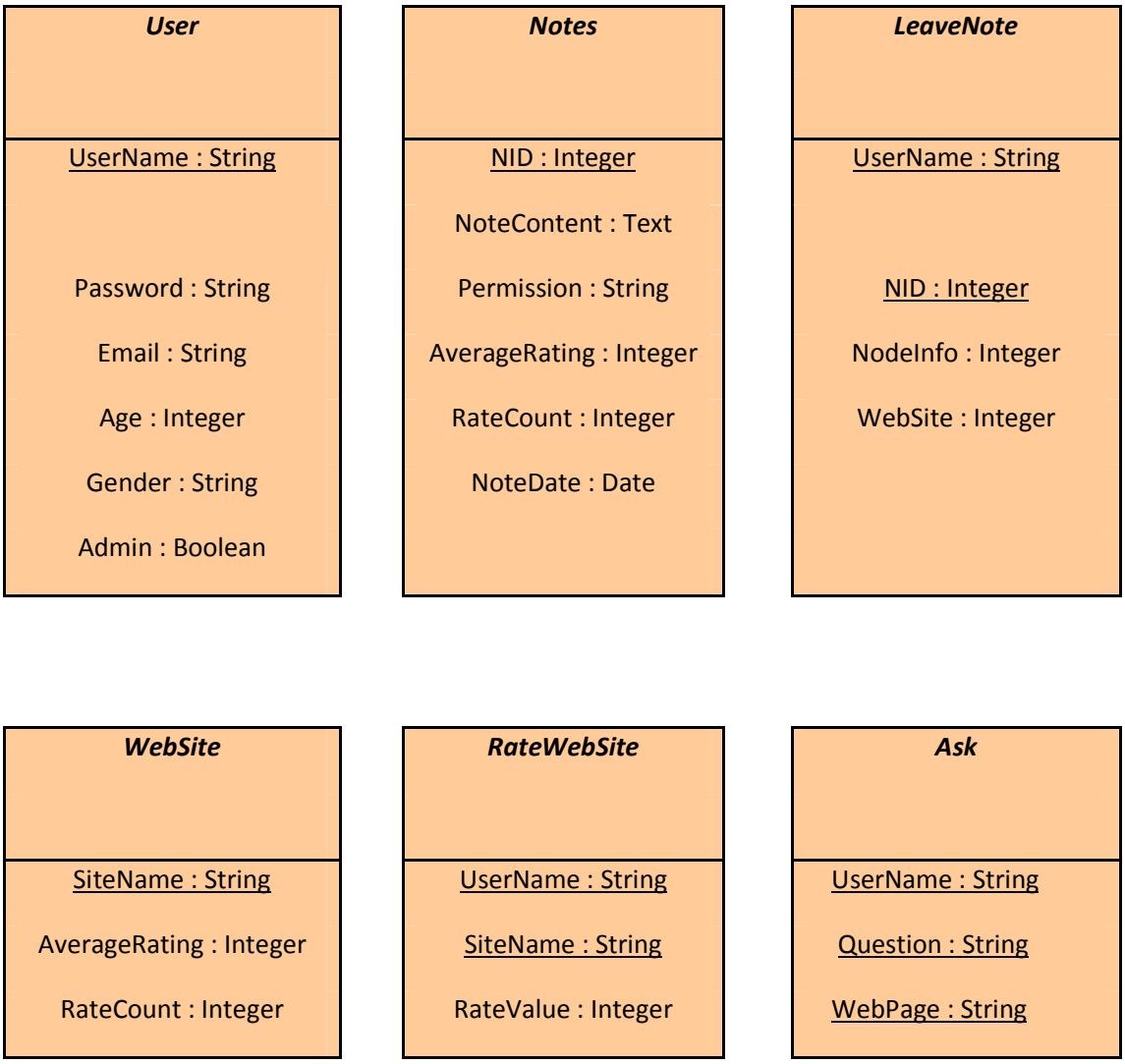
- QAPair

| Data | Type & Size | Format |
|---|---|---|
| Question* | VARCHAR-20 | Text |
| WebPage* | VARCHAR-50 | Text |
| Answer* | VARCHAR-50 | Text |

- Ask

| Data | Type & Size | Format |
|---|---|---|
| UserName* | VARCHAR-20 | Text |
| Question* | VARCHAR-50 | Text |
| WebPage* | VARCHAR-50 | Text |

### 3.5.2. ENTITY SETS & DESCRIPTIONS



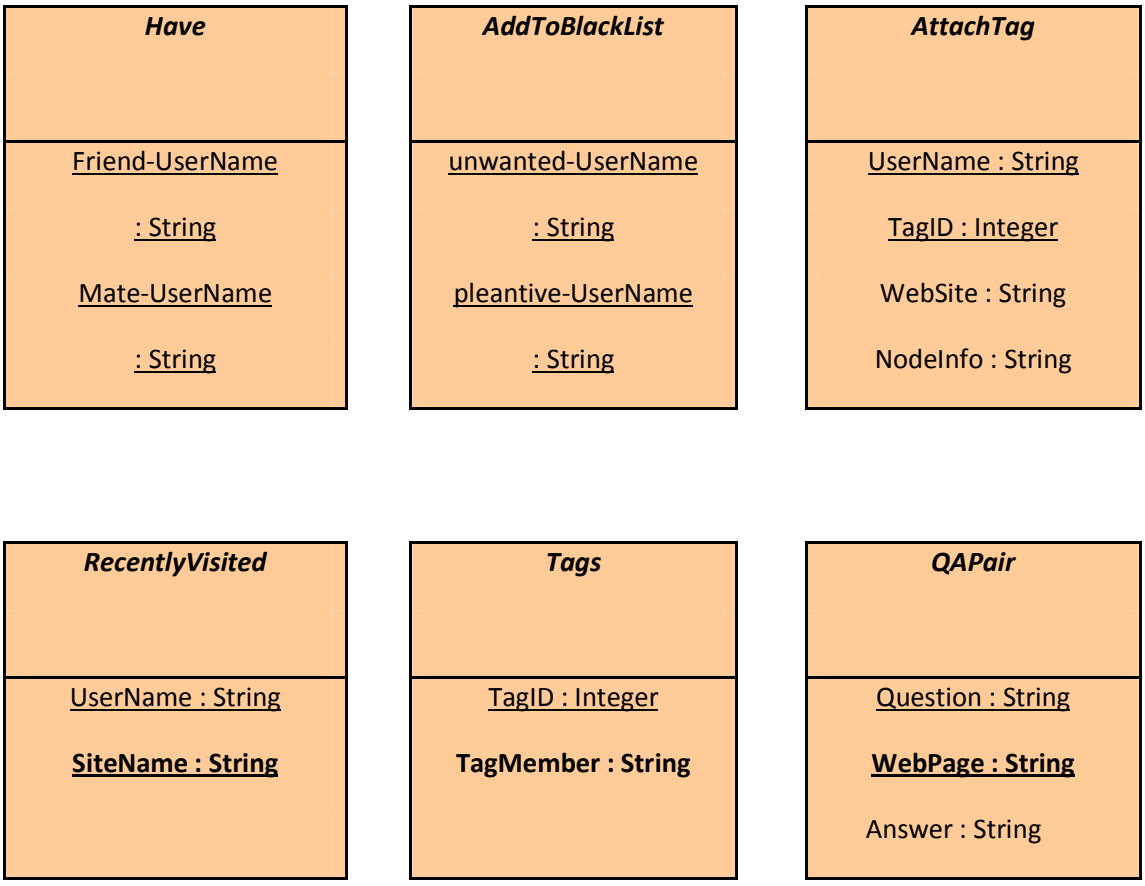



FIGURE 3.5.2.1: ENTITY SET

*ENTITY DESCRIPTIONS*

- User

In the database of Xilent; the account information and personal details of the administrators and other users are kept in the 'User' entity.

UserName: Each user and admin of the system has a unique user name in Xilent; thus the attribute 'UserName' which holds the user names is the primary key of the entity 'User'.

Password:  This string field is the matched password for the user name of the user.

Email: The electronic mails of the users are kept in this field.

Age: The age of the user

Gender: The gender of the user which can have the value male, female or null

Admin: This field can only be true or false. If it is true this means that the user is an admin; otherwise it is not.

- Notes

Xilent has a feature which allows the users to leave notes on any part of any pages they visit. The content of the note is bounded with the creativity of the user minds. Although they can be used individually by the leavers like writing on the pages of a reading book, they can be used for informing other visitors about anything.

NID:  In Xilent database each note has a unique id.

NoteContent: The content of the note, namely the text that the note involves is kept in the 'NoteContent' attribute.

AverageRating: The users which can see and read the note, can rate that note as well. The sum of these numeric rating values per the rating count of that note is kept in 'AverageRating' field and can be seen by the users who have the ability to see that note.

RateCount: This field is the number that how many times that note is rated.

NoteDate: The day, month and year information which holds the exact time that the note is submitted on.

- Tags

Tags are kind of notes as well. They both have the same functionalities like involving text and having the ability of being put on anywhere of the web-page. As AQUT we aim to put difference between these two by changing their styles. In other words, Xilent Tags and Xilent Notes will be seen quite different from each other although they have the same specialties. Tags will appear more formal in order to direct the users to use them for the page related issues while the notes appear more like sweet post-its.

TagID: Id of the tag is the primary key of 'Tags' attribute.

TagMember: The text which the user writes on his tagging.

- Website

Xilent has to keep information about the pages because of its independency from the websites.

SiteName: Websites have unique names and Xilent keeps their whole names in the 'SiteName' attribute in order to recognize and manage them.

AverageRating: The users also have the power of evaluating the web pages with Xilent. In this new feature the users can be aware of the evaluating of web pages without searching it on other places with only using Xilent. In 'Average Rating' field the ratings per rating time is kept and served to all Xilent users.

RateCount: In this attribute of the database; the number that how many times the web page is rated kept.

- QAPair

Xilent has question answering agent that helps users to find what they are looking for in webpage they are visiting. This entity stores correct answer to question asked by users.

Question: The field for Question that asked by user.

WebPage: The field for web page where question asked by user.

Answer: The field for answer to question asked by user.

*RELATION DESCRIPTIONS*

- AddToBlackList

Xilent's user can ignore other user that he/she wants. This capability protect users from being disturbed by another user.

Unwanted_UserName: The name of the user who is blocked by specified user.

Pleantive_UserName: The name of the user who blocks the user he/she does not have a connection with him/her.

- Have

Xilent's user can store their friend's user name to their account. This relation stores infornation about user's friends.

Friend_UserName : The name of the user who has specified friend.

Mate_UserName : The name of the user who is added as a friend by another user.

- LeaveNote

A user can leave and read notes to web page he visits. The ability to see the notes is up to the leaver of the note. If the note is left public it is seen and read by every user who visits that page. If the note is left to friends only the friends of the user can see and read that note. If the note is left private only the leaver of the note can see and read the note.

UserName : The name of the user who attachs left specified note.

TagID : The id number of note that is left.

NodeInfo : HTML node information of note that is kept for learning where user leaves this note on web page.

Website : The web page where tag is left.

- RateNot

Xilent serves user to rate notes left by other users. This relation kept information about rating of these notes.

UserName : The name of the user who rates specified note.

NID : The id of note that is rated.

RateValue : Rating given to specified note by a user.

- AttachTag

A user registered to Xilent can attach tags to a paragraph of the web page and image of the web page. Therefore, Database of the system keeps information about attached tags and where it attached.

UserName : The name of the user who attachs specified tag.

TagID : The id number of attached tag.

NodeInfo : HTML node information of tag that is kept for learning where user leaves tag on the web page.

Website : The web page where tag is left.

- RecentlyVisited

Xilent database stores data about web pages that user visits recently.

UserName : The name of the user who visit specified web page recently.

SiteName : The url of the web page that is visited by specified user recently.

- RateWebSite

Xilent has capability for user to evaluate website. This relation stores data about rated web pages.

UserName : The name of the user who rate the web page that kept in the SiteName attribute.

SiteName : The url of the web site that is rated by user.

RateValue : Float value gived by user to specified web page.

- Ask

This relation holds data about user and his question for question answering agent. This Relation holds questions whose answers are found.

UserName: The name of the user who asks question.

Question:  The field for question asked by the user.

WebPage: The field for web page where question asked by user.

### 3.5.3. DATABASE TABLES

*createDatabase.sql*

```
\i User.sql
\i Notes.sql
\i WebSite.sql
\i Tags.sql
\i AddToBlackList.sql
\i Have.sql
\i LeaveNote.sql
\i RateNote.sql
\i AttachTag.sql
\i RecentlyVisited.sql
\i RateWebSite.sql
\i QAPair.sql
\i Ask.sql
```

*deleteDatabase.sql*

```
DROP TABLE User
DROP TABLE Notes
DROP TABLE WebSite
DROP TABLE Tags
DROP TABLE AddToBlackList
```

```
DROP TABLE Have
DROP TABLE LeaveNote
DROP TABLE RateNote
DROP TABLE AttachTag
DROP TABLE RecentlyVisited
DROP TABLE RateWebSite
DROP TABLE QAPair
DROP TABLE Ask
```

- ✓ *User.sql*

```
CREATE TABLE User (
UserName        VARCHAR(20) NOT NULL,
Password        VARCHAR(20) NOT NULL,
Email           VARCHAR(40) NOT NULL,
Age             INTEGER,
Gender          VARCHAR(10),
Admin           BOOLEAN NOT NULL,
UNIQUE (Email),
PRIMARY KEY (UserName)
);
```

- ✓ *Notes.sql*

```
CREATE TABLE Notes(
NID                     INTEGER NOT NULL,
NoteContent             VARCHAR(100) ,
AverageRating           FLOAT,
```

```
RateCount           INTEGER,
NoteDate            TIMESTAMP NOT NULL,
Permission          VARCHAR(10),
PRIMARY KEY(NID)
);
```

- ✓ *Tags.sql*

```
CREATE TABLE Tags(
TagID               BIGINT NOT NULL,
TagMember           VARNAME(20) NOT NULL,
PRIMARY KEY(TagID)
);
```

- ✓ *WebSite.sql*

```
CREATE TABLE WebSite(
SiteName            VARCHAR(50) NOT NULL;
AverageRating       FLOAT,
RateCount           INTEGER,
PRIMARY KEY(SiteName)
);
```

- ✓ *QAPair.sql*

```
CREATE TABLE QAPair (
Question            VARCHAR (50) NOT NULL,
WebPage             VARCHAR (50) NOT NULL,
Answer              VARCHAR (50) NOT NULL,
```

```
PRIMARY KEY (Question, WebPage)
);
```

- ✓ *AddToBlackList.sql*

```
CREATE TABLE AddToBlackList(
Unwanted_UserName        VARCHAR(20) NOT NULL,
Pleantive_UserName       VARCHAR(20) NOT NULL,
PRIMARY KEY(Unwanted_UserName, Pleantive_UserName),
FOREIGN KEY (Unwanted_UserName) REFERENCES Users (UserName),
FOREIGN KEY (Pleantive_UserName) REFERENCES Users (UserName)
);
```

- ✓ *Have.sql*

```
CREATE TABLE Have(
Friend_UserName          VARCHAR(20) NOT NULL,
Mate_UserName            VARCHAR(20) NOT NULL,
PRIMARY KEY(Friend_UserName, Mate_UserName)
FOREIGN KEY (Friend_UserName) REFERENCES Users (UserName),
FOREIGN KEY (Mate_UserName) REFERENCES Users (UserName)
);
```

- ✓ *LeaveNote.sql*

```
CREATE TABLE LeaveNote(
UserName                 VARCHAR(20) NOT NULL,
NID                      INTEGER NOT NULL,
NodeInfo                 VARCHAR(40) NOT NULL,
```

```
WebSite                 VARCHAR(50) NOT NULL,
PRIMARY KEY(UserName, NID),
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (NID) REFERENCES Notes (NID)
);
```

- ✓ *RateNote.sql*

```
CREATE TABLE RateNote(
UserName                VARCHAR(20) NOT NULL,
NID                     INTEGER NOT NULL,
RateValue               INTEGER,
PRIMARY KEY(UserName, NID),
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (NID) REFERENCES Notes (NID)
);
```

- ✓ *AttachTag.sql*

```
CREATE TABLE AttachTag(
UserName                VARCHAR(20) NOT NULL,
TagID                   BIGINT NOT NULL,
NodeInfo                VARCHAR(40) NOT NULL,
WebSite                 VARCHAR(50) NOT NULL,
PRIMARY KEY(UserName, NID),
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (NID) REFERENCES Tags (TagID)
);
```

- ✓ *RecentlyVisited.sql*

```
CREATE TABLE RecentlyVisited(
UserName                VARCHAR(20) NOT NULL,
SiteName                VARCHAR(50) NOT NULL,
PRIMARY KEY(UserName, SiteName),
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (SiteName) REFERENCES WebSite (SiteName)
);
```

- ✓ *RateWebSite.sql*

```
CREATE TABLE RateWebSite(
UserName                VARCHAR(20) NOT NULL,
SiteName                VARCHAR(50) NOT NULL,
RateValue               INTEGER NOT NULL,
PRIMARY KEY(UserName, SiteName),
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (SiteName) REFERENCES WebSite (SiteName)
);
```

- ✓ *Ask.sql*

```
CREATE TABLE Ask (
UserName                VARCHAR (20) NOT NULL,
Question                VARCHAR (50) NOT NULL,
WebPage                 VARCHAR (50) NOT NULL,
PRIMARY KEY (UserName, Question, WebPage)
```

```
FOREIGN KEY (UserName) REFERENCES Users (UserName),
FOREIGN KEY (Question) REFERENCES QAPair (Question),
FOREIGN KEY (WebPage) REFERENCES QAPair (WebPage)
);
```

# 4. OBJECT ORIENTED DIAGRAMS

## 4.1. USE CASE DIAGRAMS

- ✓ *USER USE CASE DIAGRAM*

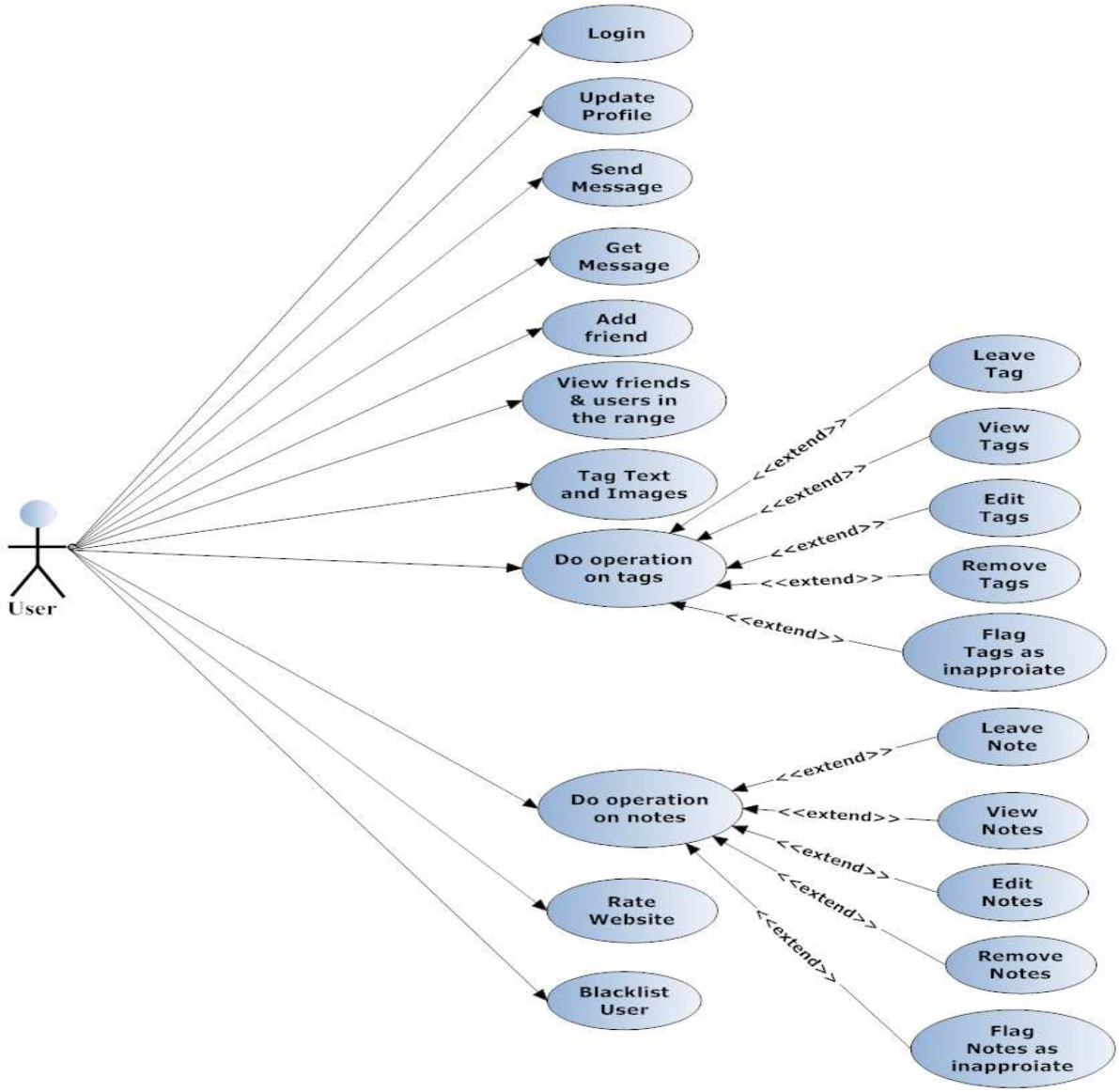



FIGURE 4.1.1: USER USE CASE DIAGRAM

✓ *ADMIN USE CASE DIAGRAM*

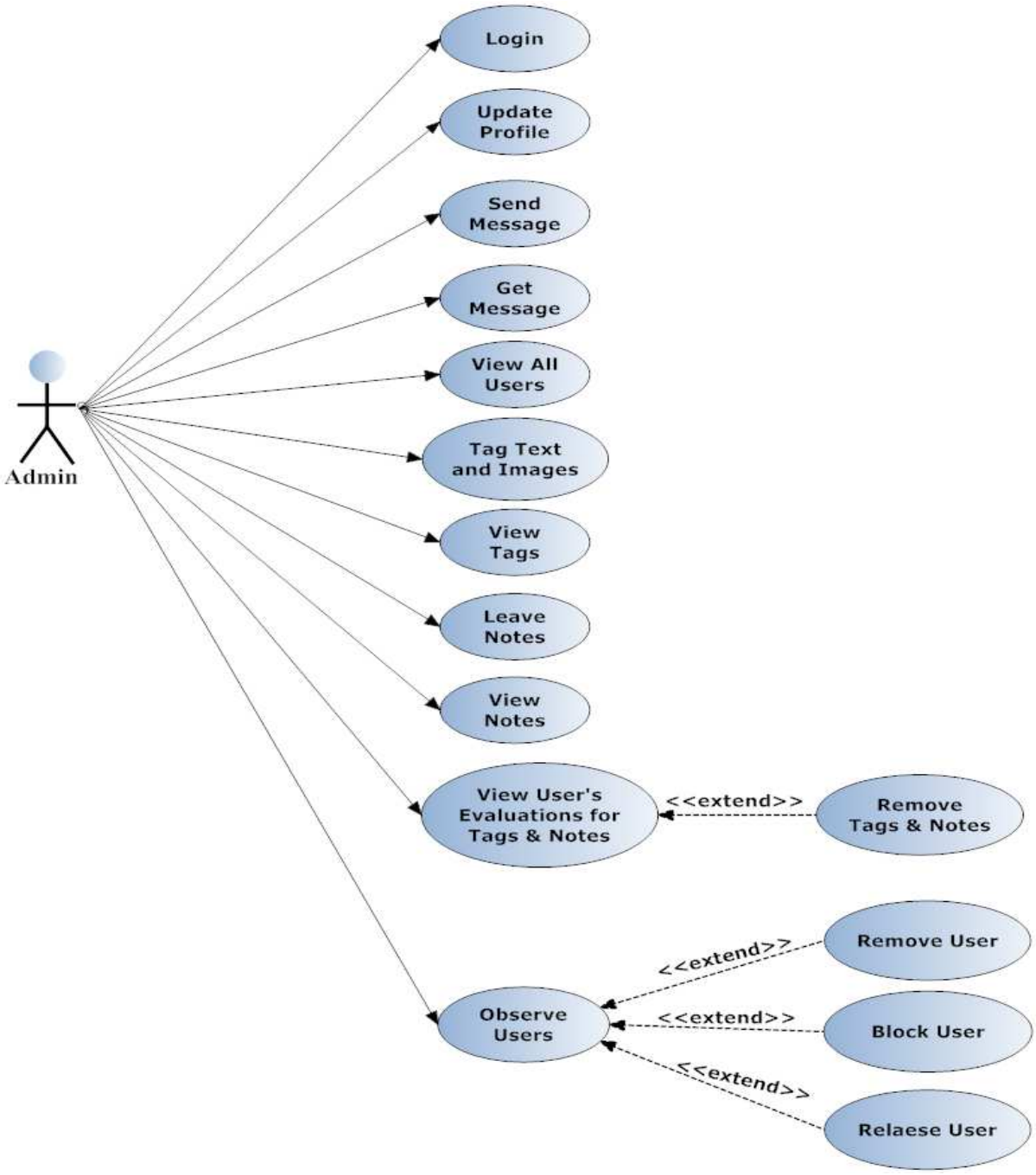



FIGURE 4.1.2: ADMIN USE CASE DIAGRAM

✓ *QA ASSISTANT USE CASE DIAGRAM*

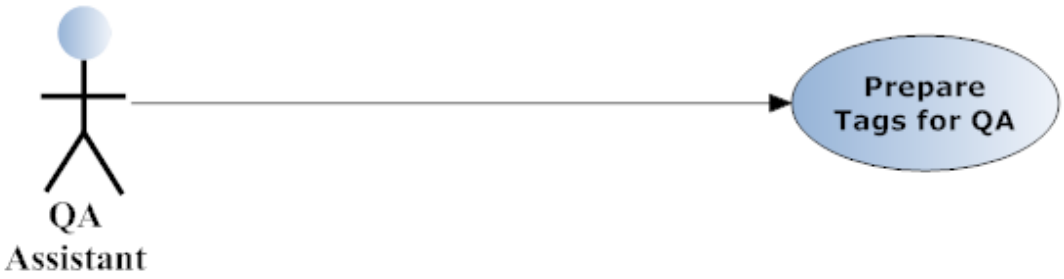



FIGURE 4.1.3: QA ASSISTANT USE CASE DIAGRAM

✓ *QA ASSISTANT USE CASE DIAGRAM*

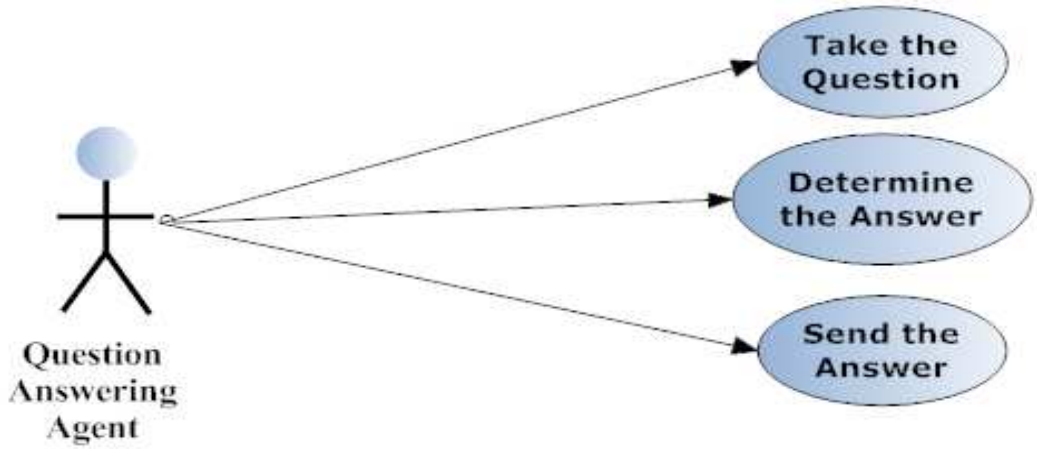



FIGURE 4.1.4: QA USE CASE DIAGRAM

## 4.2. ACTIVITY DIAGRAMS

### 4.2.1. SIGN UP & LOGIN

If it is your first entrance to XILENT, first you will be shown a signup/login page. To sign up an account, user fills the sign-up form and submits this form, so this account becomes active and user is now able to enter to Xilent. On the other hand, user can choose to login. If both username and password are valid, user directly enters to system. In case of user's password being wrong, this user is directed to the login page again but if he or she forgets his/her password, Xilent password renewal process will work.

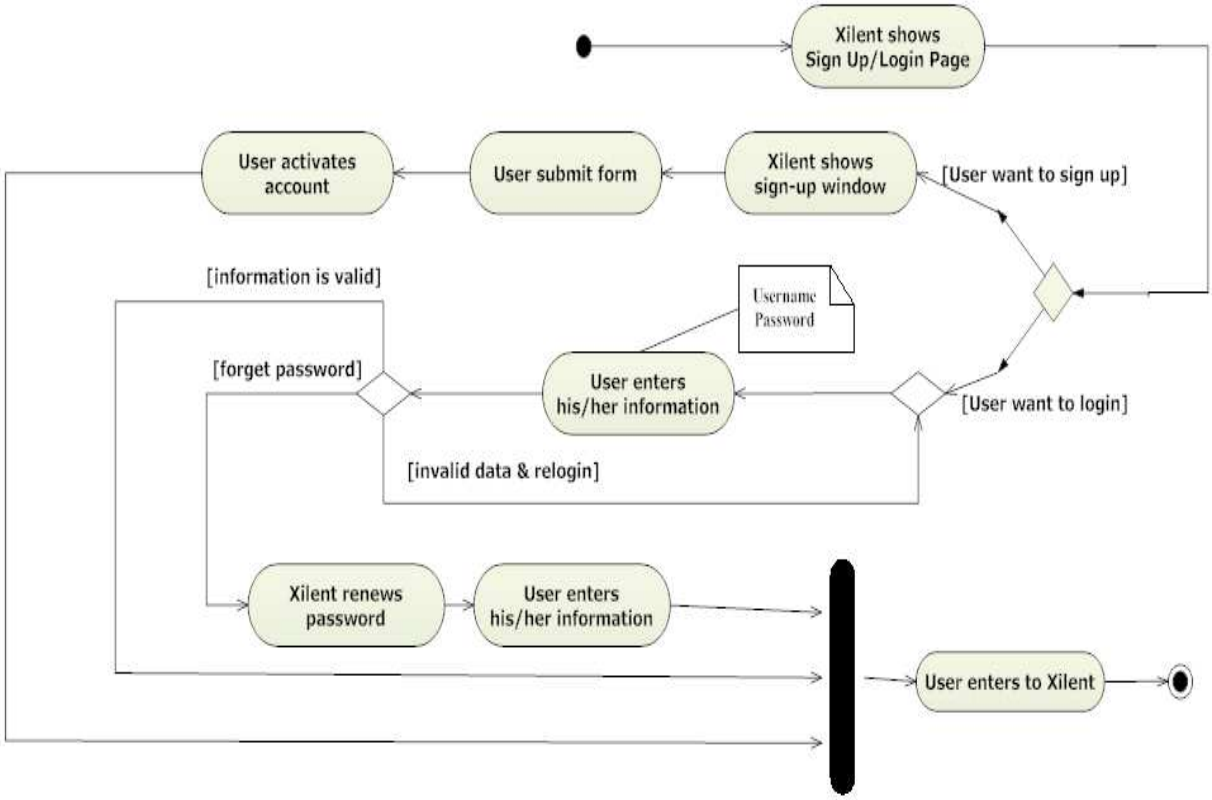



FIGURE 4.2.1.1: SIGN UP & LOGIN ACTIVITY

### 4.2.2. SEND MESSAGE

To send a message user first activates the message window that is on the plug-in by clicking on it. User then starts the conversation according to his/her receiver choice.

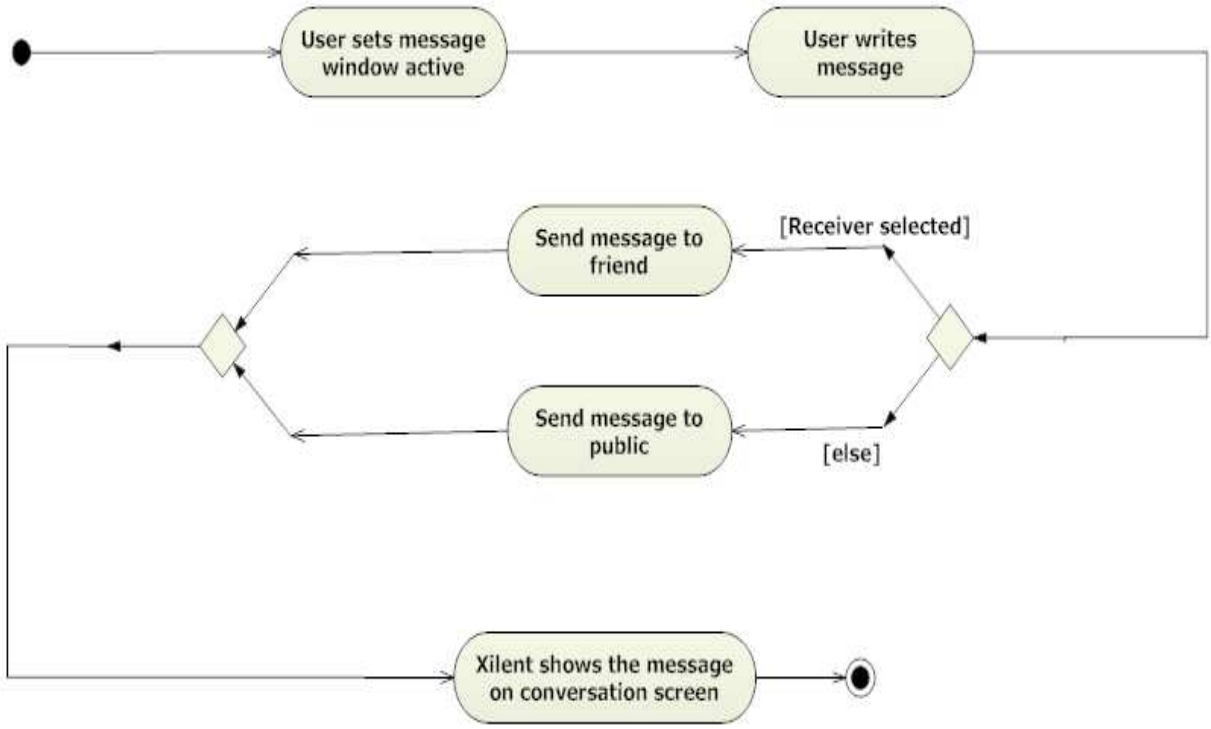



FIGURE 4.2.2.1: SEND MESSAGE ACTIVITY

### 4.2.3. NOTE OPERATIONS

While browsing the web pages, user can leave a note to anywhere on that webpage by just right clicking and selecting “leave note” option from the pop-up menu. User can also edit or remove the notes that is leaved by him/her, by the menu that is located at his/her personal

account page. These operations can also be applied when the note is seen while browsing web pages. But on the other hand, if the user is not the owner of the encountered note, he/she can only flag the note as inappropriate or select not to see the note later on.

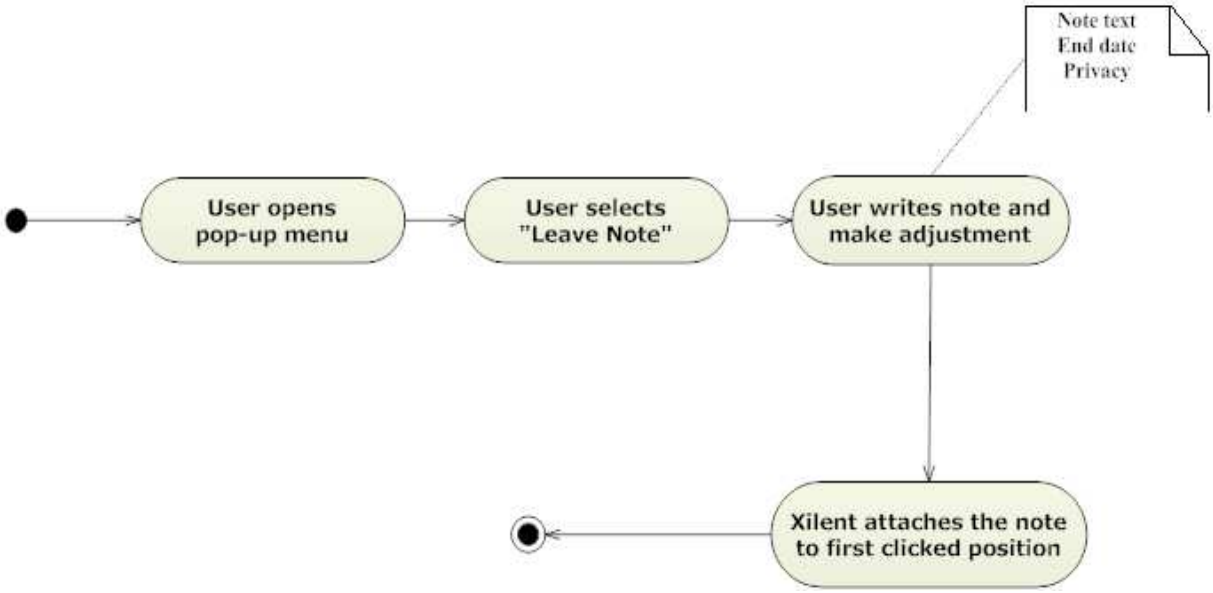



FIGURE 4.2.3.1: LEAVE NOTE ACTIVITY

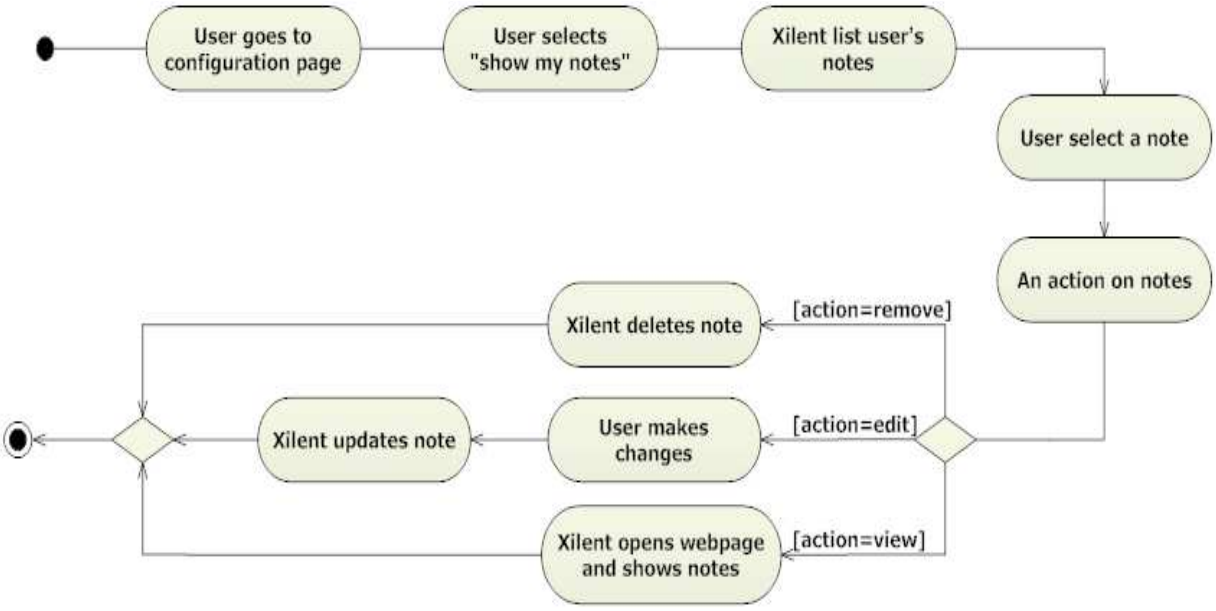



FIGURE 4.2.3.2: NOTE OPERATION ON PERSONAL WEBPAGE ACTIVITY

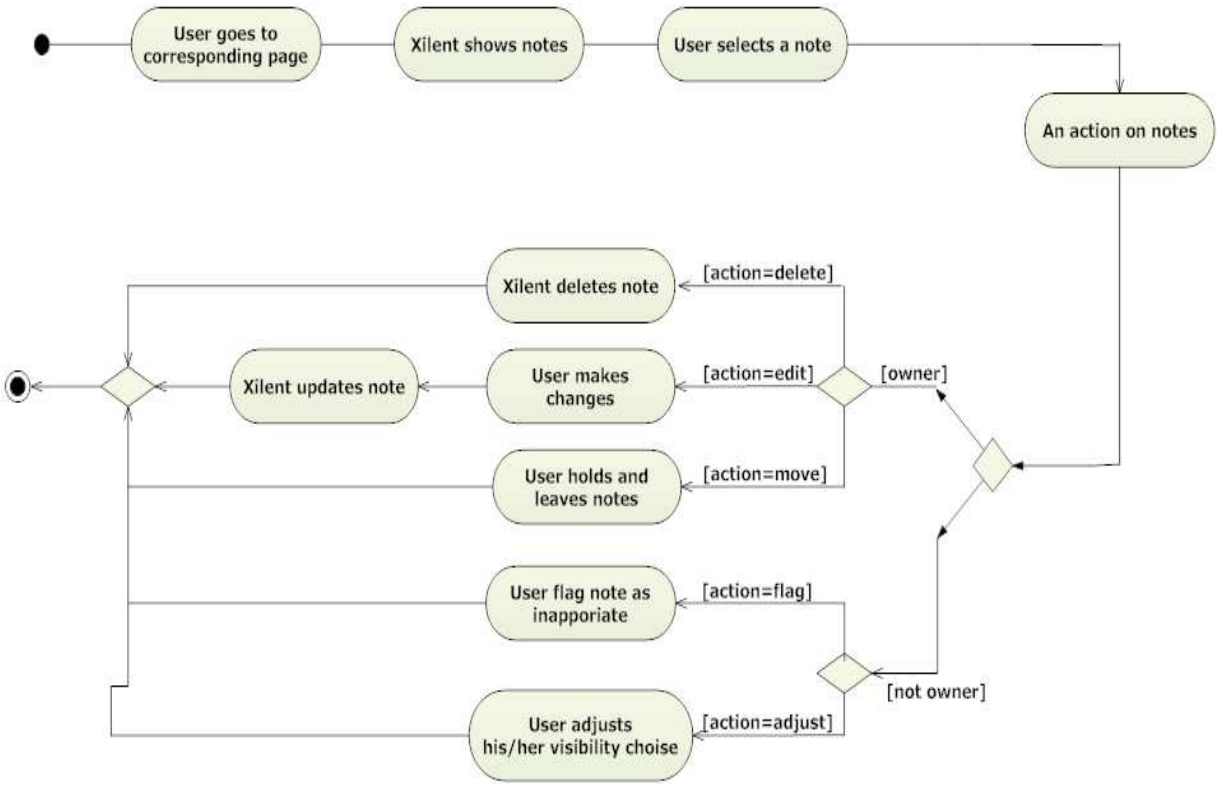



FIGURE 4.2.3.3: NOTE OPERATION ON VISITED WEBPAGE ACTIVITY

## 4.2.4. TAG OPERATIONS

While browsing the web pages, user can tag any information on that webpage by just right clicking and selecting “tag information” option from the pop-up menu. User can also edit or remove the tags that is leaved by him/her, by the menu that is located at his/her personal account page. These operations can also be applied when the tag is seen while browsing web pages if the user is the owner of the encountered tag.

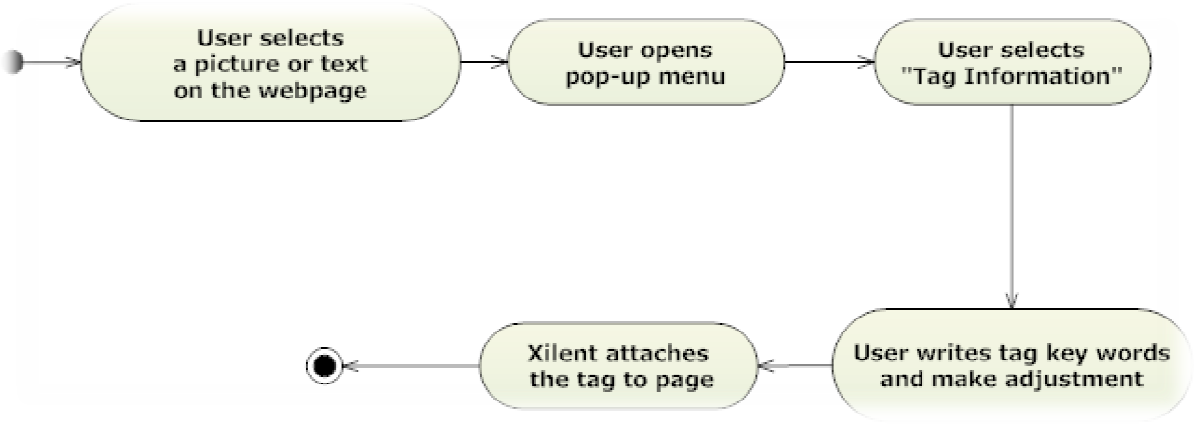



FIGURE 4.2.4.1: TAG INFORMATION ACTIVITY

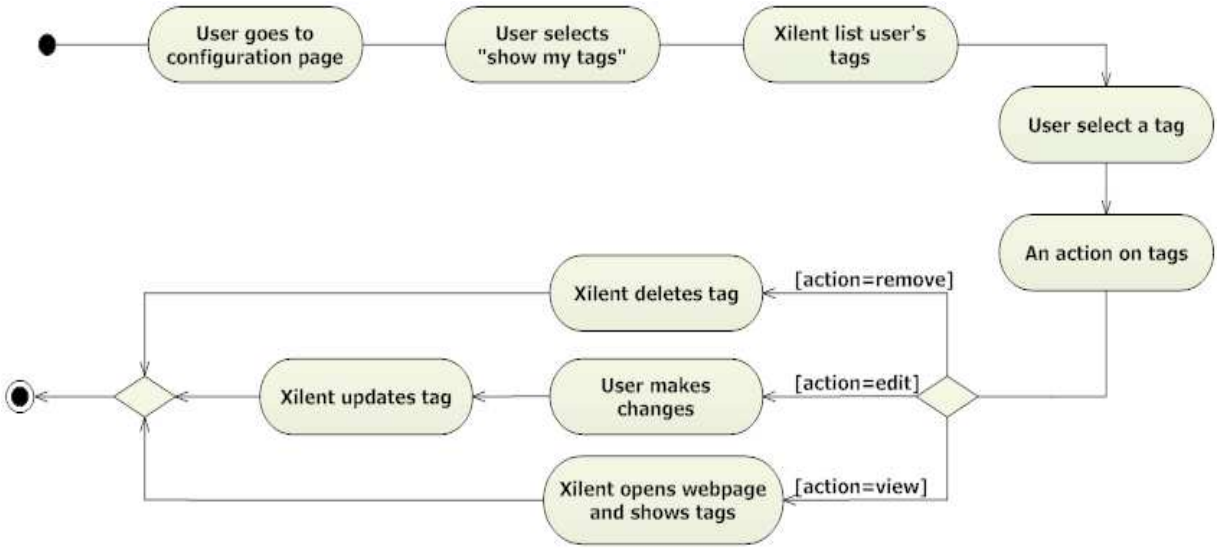



FIGURE 4.2.4.2: TAG OPERATION ON PERSONAL WEBPAGE ACTIVITY

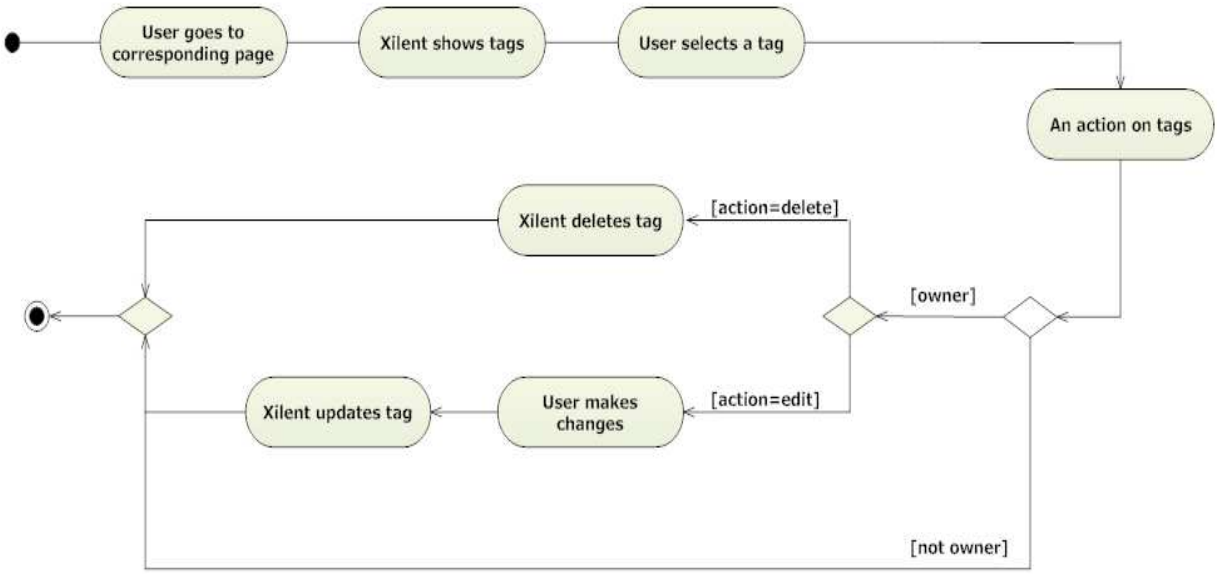



FIGURE 4.2.4.3: TAG OPERATION ON VISITED WEBPAGE ACTIVITY

### 4.2.5. QUESTION ANSWERING AGENT

Xilent will be an application which has a QA service. When user activates the QA window that is on the plug-in by clicking on it, question answering agent will be ready to help the user. User asks for the answer and QA agent finds some appropriate answers from the tagged information and shows one of them to user. If user accepts this answer, this question answer pair will be inserted to the database for future use; otherwise QA agent presents one of the other alternatives.

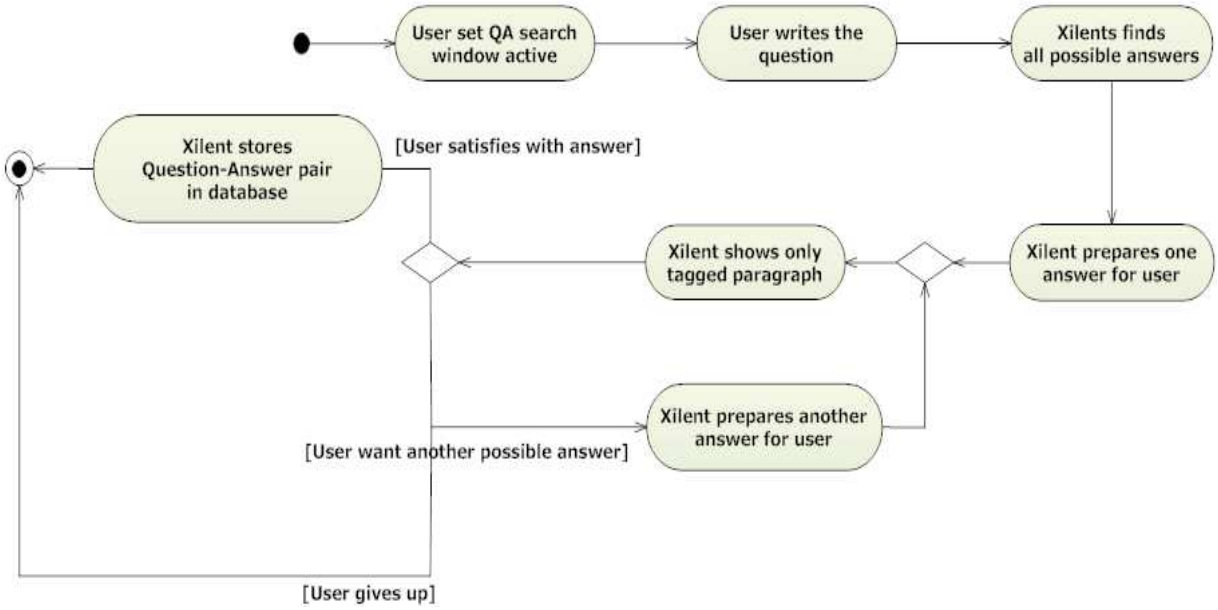



FIGURE 4.2.5.1: QA ACTIVITY

## 4.2.6. RATE WEBPAGE

While surfing on the internet, user can rate the visited webpage by first activating the rate box and then selecting a rate for this webpage.

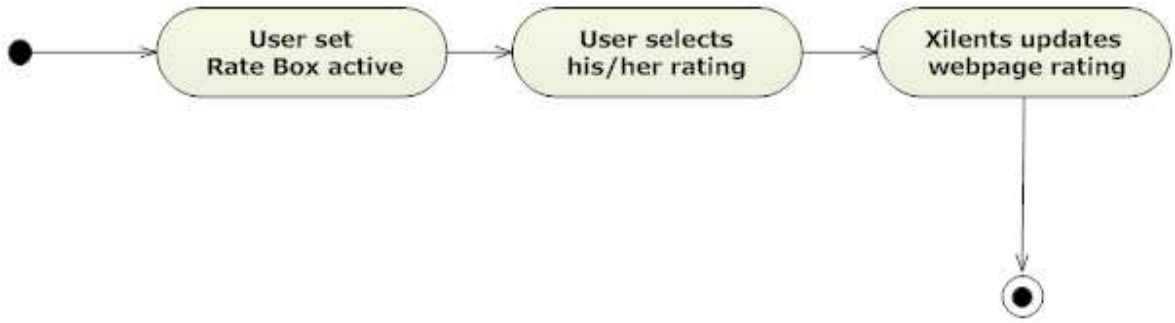



FIGURE 4.2.6.1: RATE ACTIVITY

## 4.2.7. ADD & INVITE FRIENDS

While surfing on the internet, user can see the people that are at the same webpage, from the radar screen which is located on the plug-in. User can make friendship with these people after adding them to his/her friend list. Moreover Xilent will help users to socialize on the internet by providing them to invite their friends that have accounts at other instant messaging services like msn.

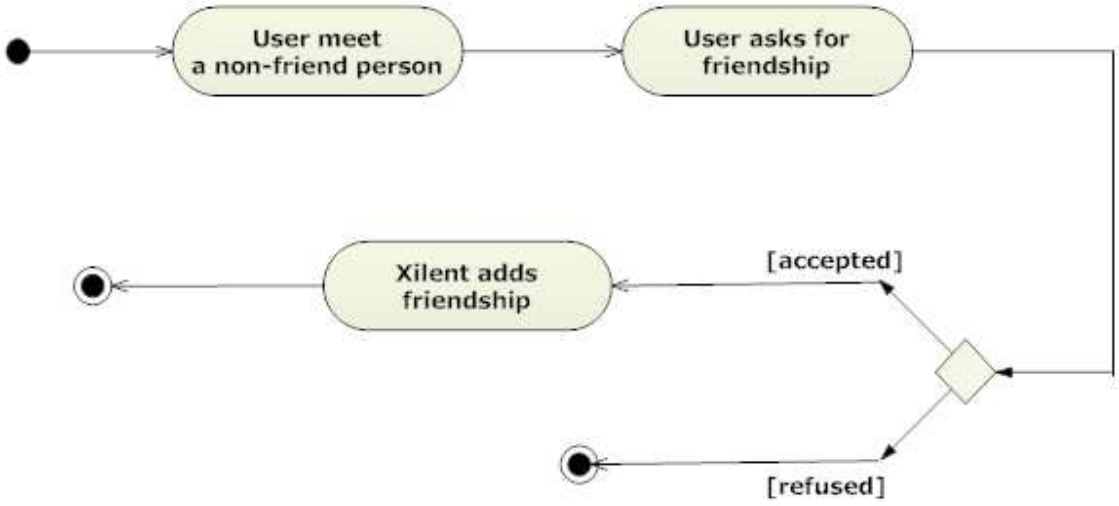



FIGURE 4.2.7.1: ADD FRIENDS ACTIVITY

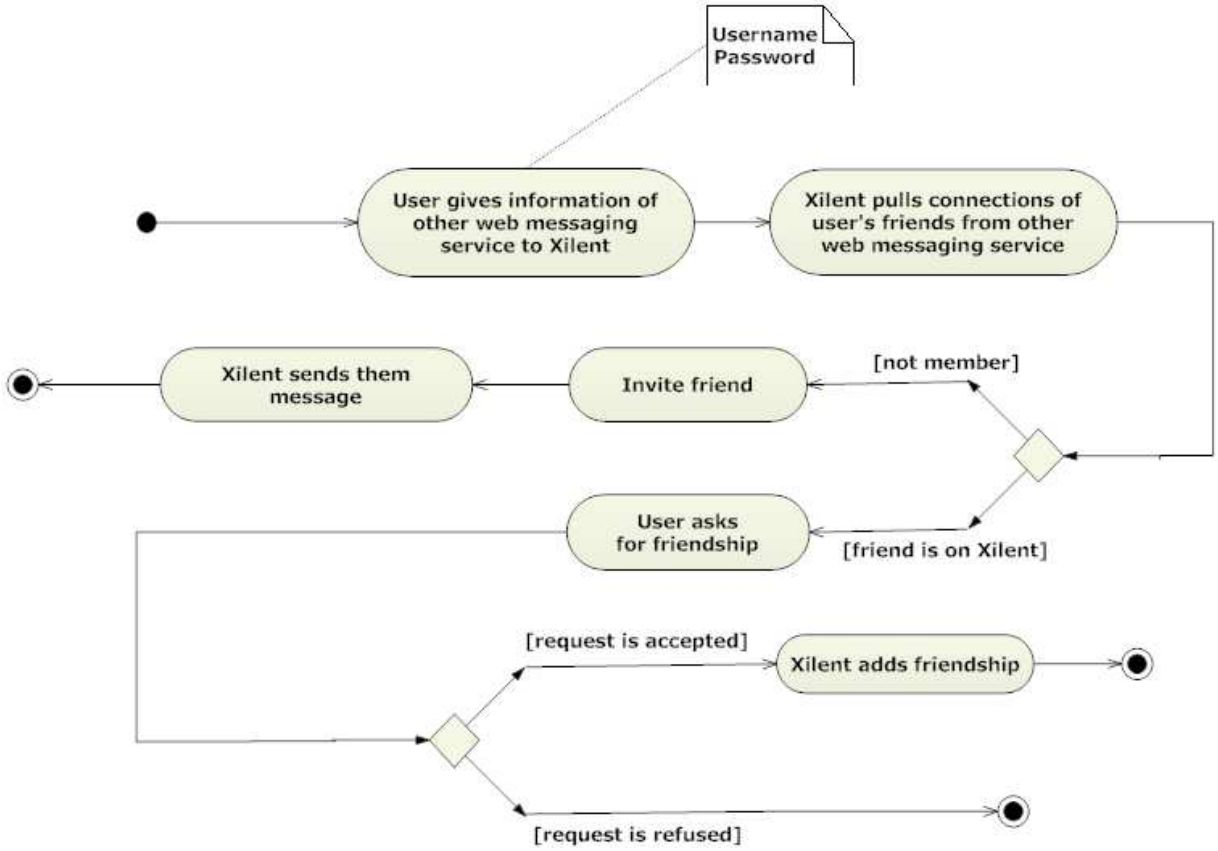



FIGURE 4.2.7.2: INVITE FRIENDS FROM OTHER INSTANT MESSAGING SERVICES ACTIVITY

## 4.3. CLASS DIAGRAMS

### 4.3.1. USER MANAGEMENT MODULE CLASS DIAGRAM


FIGURE 4.3.1.1: CLASS DIAGRAM FOR USER MANAGEMENT MODULE

- **UserAdministration** class handles the administrative operations on users. When a user signups, the validity of given information are checked by calling UserDatabaseAccess class and user has an account after confirming the confirmation mail that is sent after SignupManagement class is called. When these steps are finished User class is created. On the other hand users' login process is handled by LoginManagement class. This class calls UserDatabaseAccess class and checks the validity of the information.
- **SignupManagement** class is responsible from sending a confirmation email to the user who filled the signup form and confirming the registration of the user to the system. This class calls UserDatabaseAccess class for the creation of the new user.
- **LoginManagement** class handless the login process of registered users. First gets the user name and password, and then calls UserDatabaseAccess class to check this information's validity.
- **UserDatabaseAccess** class establishes the connection with the database and executes the queries related with the methods of this class.
- **User** class is responsible from user related issues such as updating and changing login information. User class calls UserDatabaseAccess class to perform these operations.

## 4.3.2. NOTE OPERATIONS MODULE CLASS DIAGRAM


FIGURE 4.3.2.1: CLASS DIAGRAM FOR NOTE OPERATIONS MODULE

- **UserNoteList** class handles the note operations such as adding, deleting, updating and showing notes which are defined for user's note lists. All users have a note list in Xilent. When a user wants to leave a note on a webpage, UserNoteList class creates NoteDefinition class and this class creates WebPageInformation class. User class also calls NoteDatabseAcces class to perform database operations.
- **NoteDefinition** class is created when a user adds a note on a webpage. Note related attributes are set by the methods of this class. NoteDefinition class creates WebPageInformation class.
- **WebPageInformation** class is created after the creation of NoteDefinition class when a user adds a note. Leaved note's webpage information is set by the methods of this class.
- **NoteDatabaseAccess** class establishes the connection with database and does operations such as insert, delete or update. Its methods uses queries to perform these database related operations.

## 4.3.3. TAG OPERATIONS MODULE CLASS DIAGRAM


FIGURE 4.3.3.1: CLASS DIAGRAM FOR TAG OPERATIONS MODULE

- **UserTagList** class handles the tag operations such as adding, deleting, updating and showing tags which are defined for user's tag lists. All users have a tag list in Xilent. When a user wants to tag information on a webpage, UserTagList class creates TagDefinition class and this class creates WebPageInformation class. User class also calls TagDatabaseAcces class to perform database operations.

- **TagDefinition** class is created when a user tags information on a webpage. Tag related attributes are set by the methods of this class. TagDefinition class creates WebPageInformation class.
- **WebPageInformation** class is created after the creation of TagDefinition class when a user tags information. Tagged information's webpage information is set by the methods of this class.
- **TagDatabaseAccess** class establishes the connection with database and does operations such as insert, delete or update. Its methods uses queries to perform these database related operations.

### 4.3.4. RATE MODULE CLASS DIAGRAM

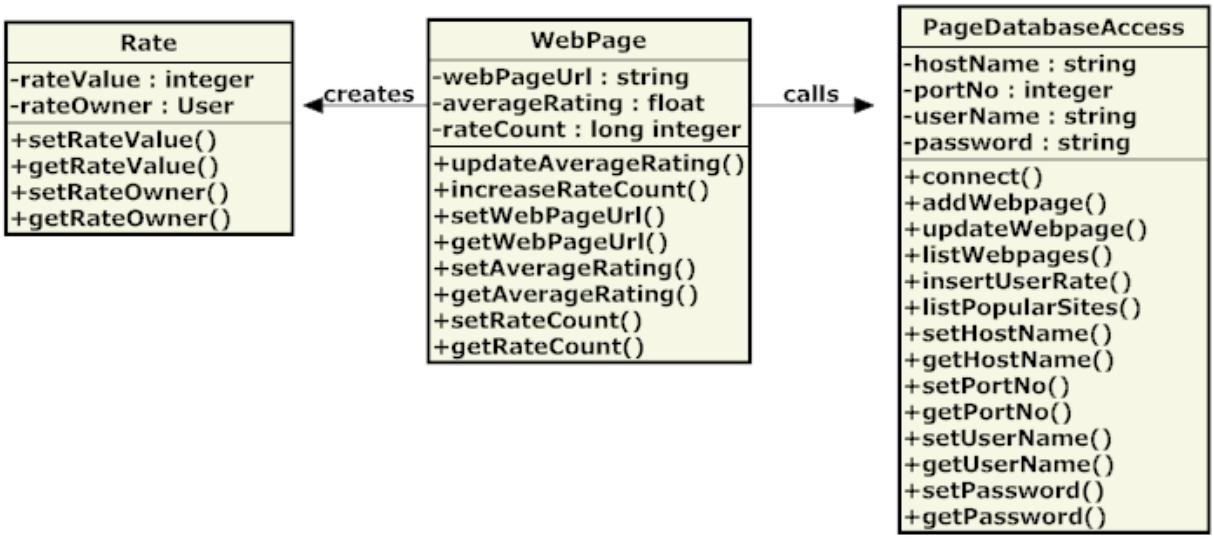



FIGURE 4.3.4.1: CLASS DIAGRAM FOR RATE MODULE

- **WebPage** class is keeping the information such as URL, average rating of the webpage and the number of people that rates the webpage. Webpage class creates Rate class

when a user rates a webpage. It also calls PageDatabaseAccess class to perform database related issues.

- **Rate** class is responsible from keeping the rate of the webpage that is given by the user and user's name.
- **PageDatabaseAccess** class establishes the connection with database and does operations such as insert, delete or update the webpage's rate information. Its methods uses queries to perform these database related operations.

### 4.3.5. INSTANT MESSAGING MODULE CLASS DIAGRAM

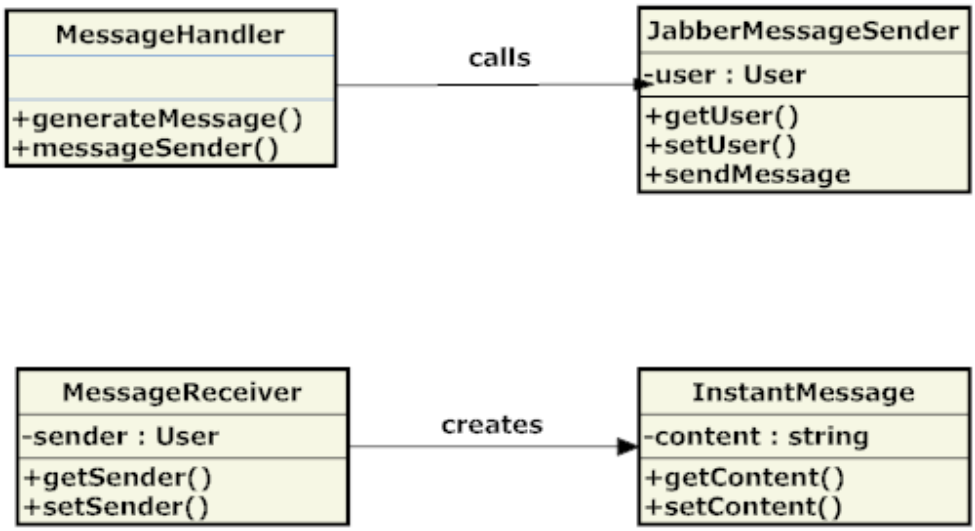



FIGURE 4.3.5.1: CLASS DIAGRAM FOR INSTANT MESSAGING MODULE

- **MessageHandler** class handles the preparation of a user message for JABBER server at message sending phase. Then it calls JabberMessageSender.
- **JabberMessageSender** class is responsible from sending the message to the receiver. Due to JABBER server's ability to send a message to the right receiver by only knowing the username of the receiver (if the user is registered to JABBER, his/her address is username@jabber.org), this class only has user data.

- **MessageReceiver** class keeps the message sender information and creates **InstantMessage** class which handles the preparation of message for receiver that comes from JABBER server.

### 4.3.6. QUESTION ANSWERING MODULE CLASS DIAGRAM

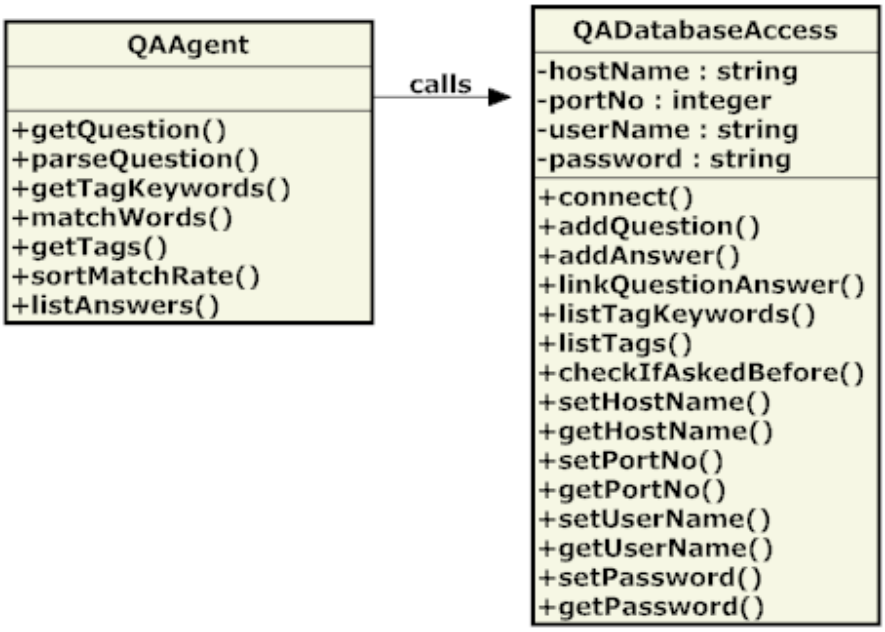



FIGURE 4.3.6.1: CLASS DIAGRAM FOR QA MODULE

- **QAAgent** class handles the question related issues that is asked by the user to the question answering agent. This class prepares the question for QA agent and calls QADatabaseAccess to search the answer among the tags that are at the database. Then gives an answer to the user among the answer list which is formed by this class.
- **QADatabaseAccess** class establishes the connection with database and does operations related with questions and possible answers. Its methods uses queries to perform these database related operations. When the user is satisfied with the given answer, this question and answer pair is inserted to the database.

## 4.3.7. GUI MODULE CLASS DIAGRAM

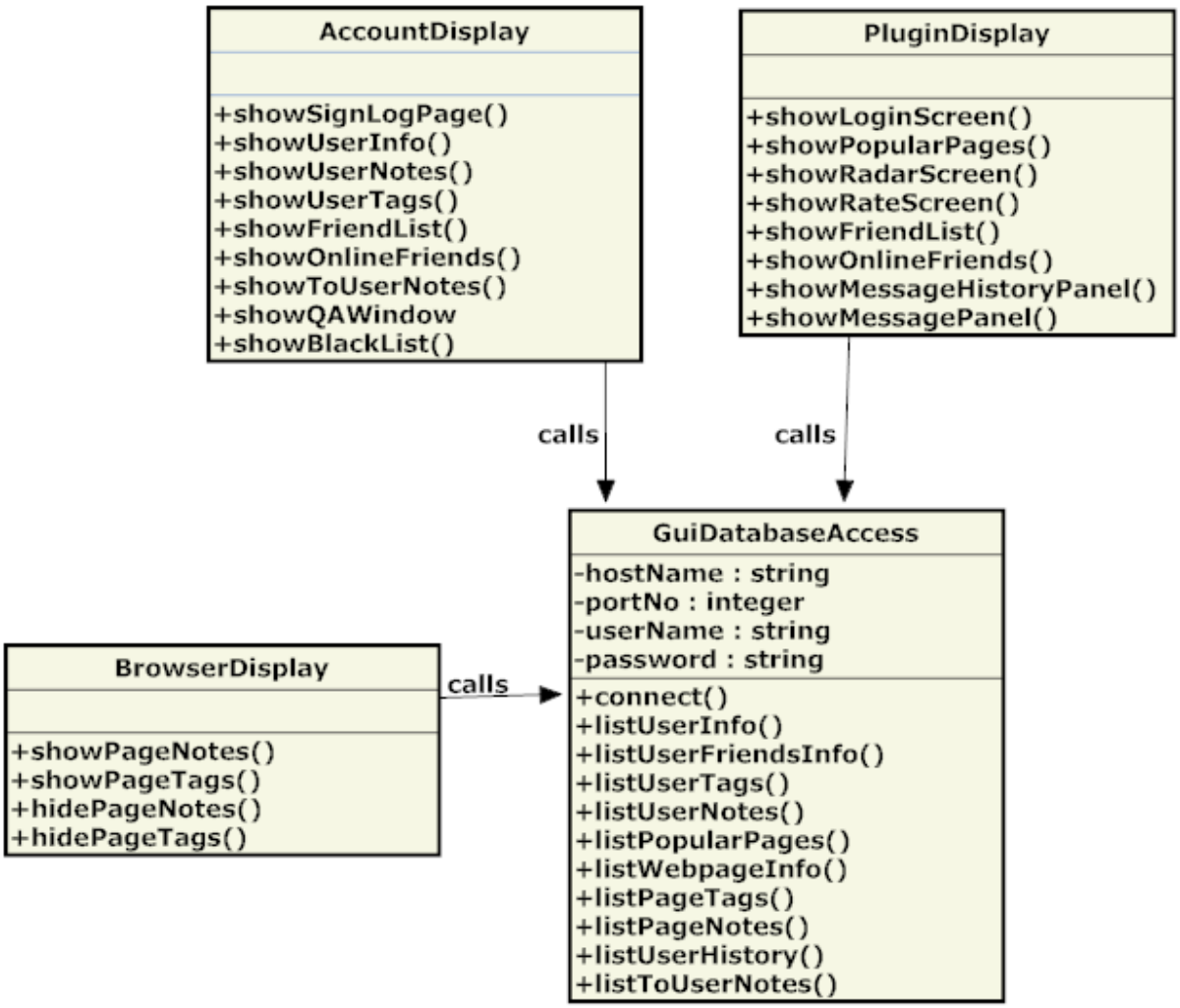



FIGURE 4.3.7.1: CLASS DIAGRAM FOR GUI MODULE

- **AccountDisplay** class handles the user's personal webpage display. The methods of this class are responsible from showing user info, notes, tags, friend and blacklisted people. AccountDisplay class calls GUIDatabaseAccess class.
- **BrowserDisplay** class is responsible from showing and hiding notes or tags that are located on the webpage that is currently being visited. This class also calls GUIDatabseAccess class.

- **PluginDisplay** class handles the display screen for plug-in. Radar screen, rate screen and message screen are some of the different parts of this class from AccountDisplay class. It calls GUIDatabaseAccess to able to show these parts after getting related lists from the database.
- **GUIDatabaseAccess** class establishes the connection with database and lists information such for AccountDisplay, BrowserDisplay and PluginDisplay. Its methods uses select queries to perform these database related operations.

## 4.4. SEQUENCE DIAGRAMS

### 4.4.1. SIGN UP

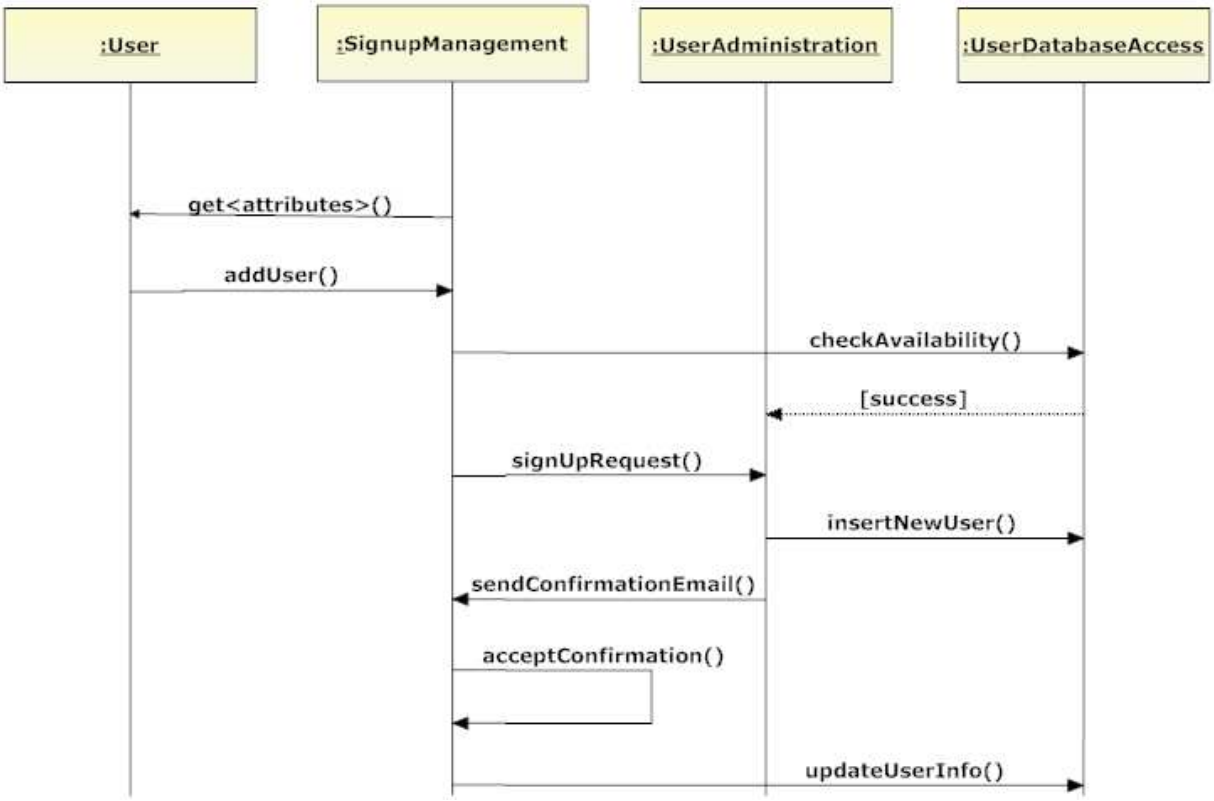



FIGURE 4.4.1.1: SIGN UP SEQUENCE DIAGRAM

## 4.4.2. LOGIN

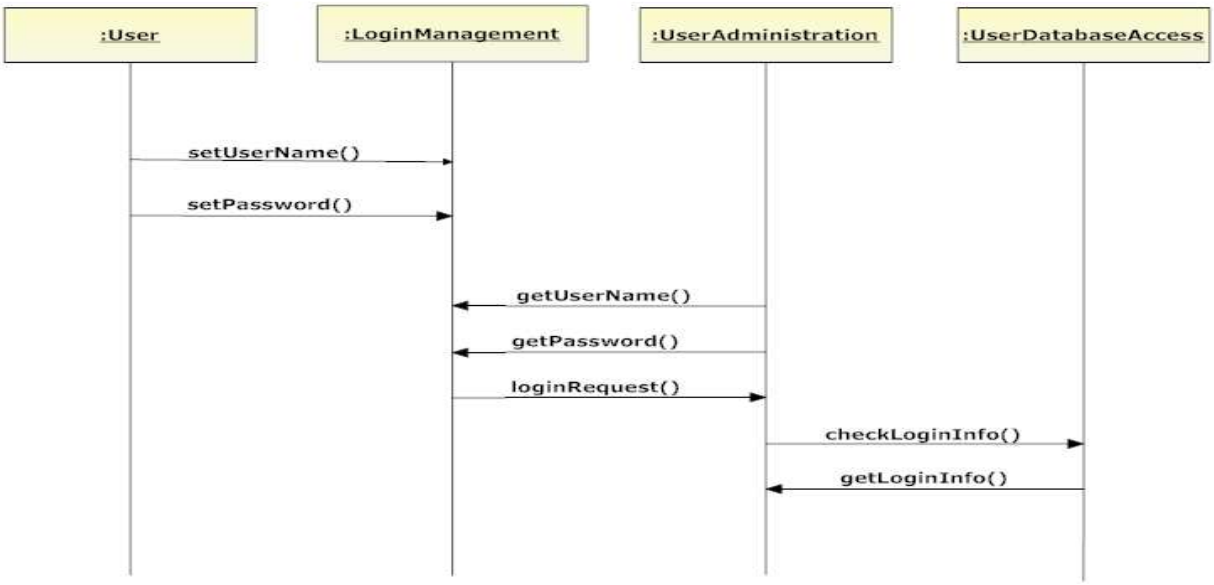



FIGURE 4.4.2.1: LOGIN SEQUENCE DIAGRAM

## 4.4.3. INSTANT MESSAGING

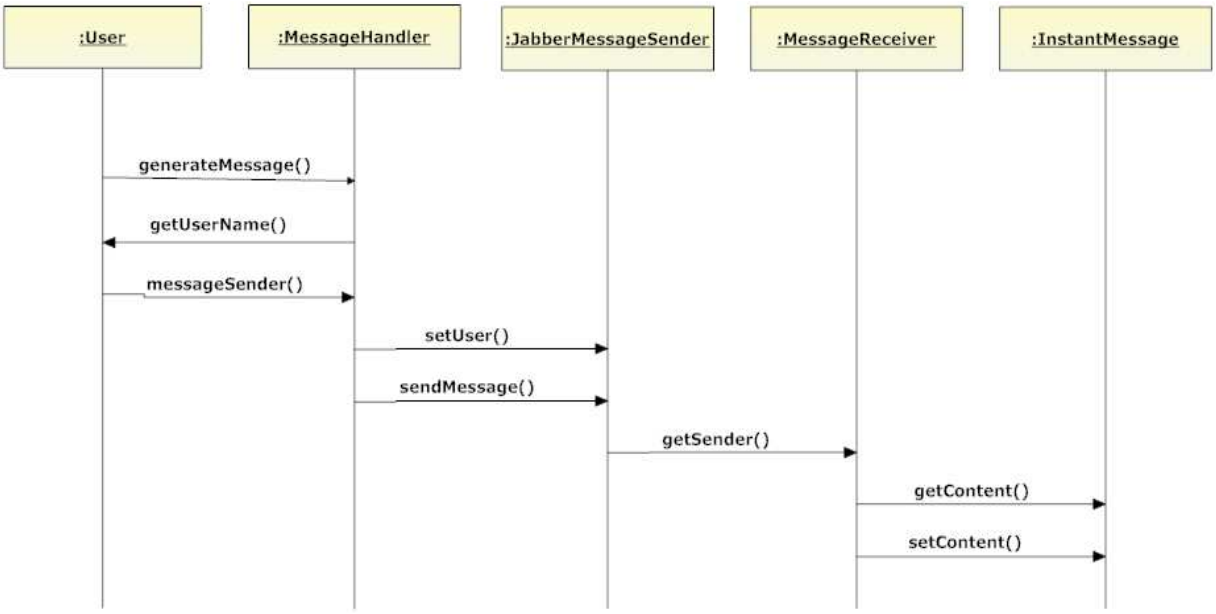



FIGURE 4.4.3.1: INSTANT MESSAGING SEQUENCE DIAGRAM

## 4.4.4. QUESTION ANSWERING

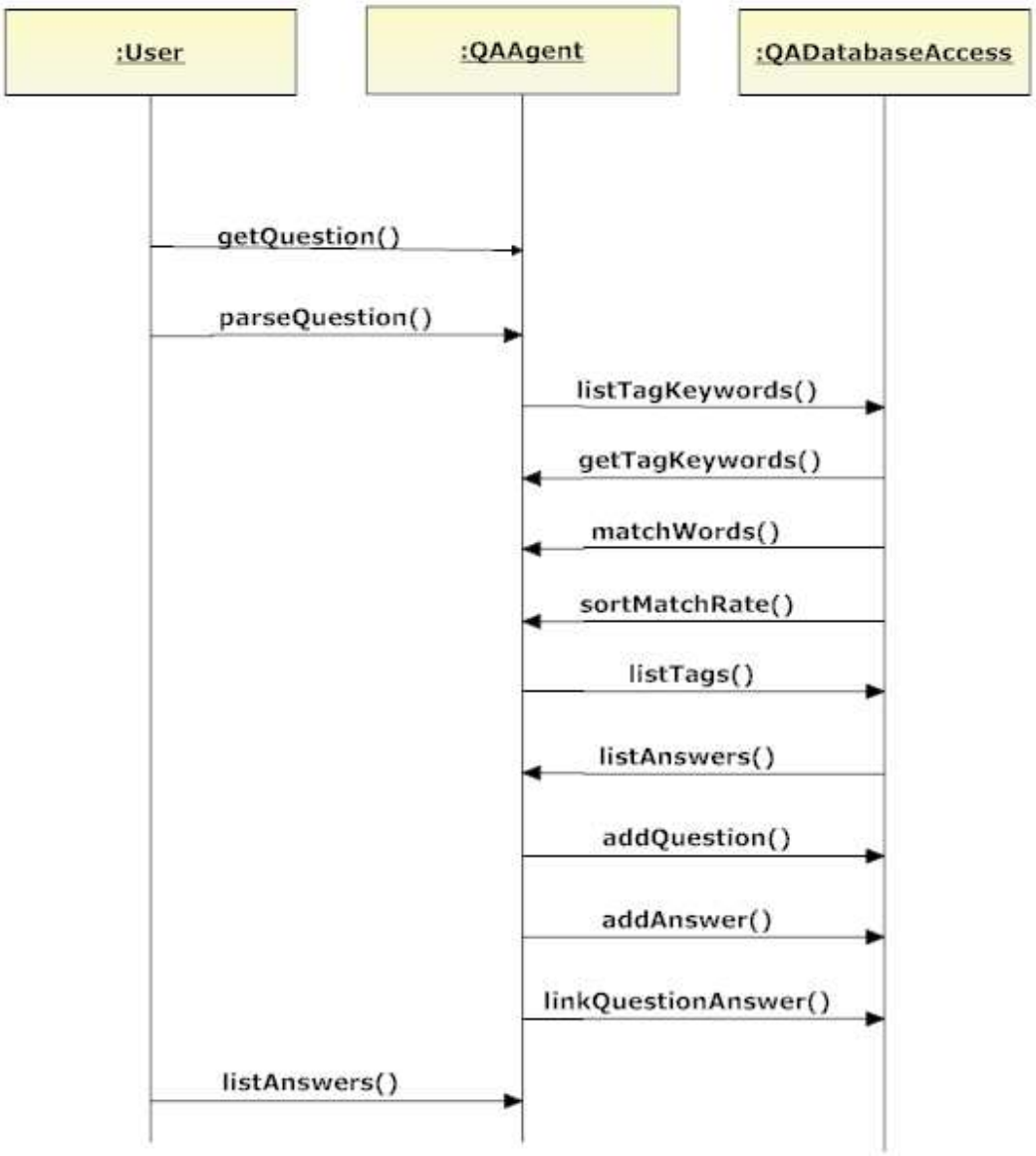



FIGURE 4.4.4.1: QUESTION ANSWERING SEQUENCE DIAGRAM

## 4.4.5. NOTE OPERATIONS


FIGURE 4.4.5.1: NOTE OPERATIONS SEQUENCE DIAGRAM

## 4.4.6. TAG OPERATIONS

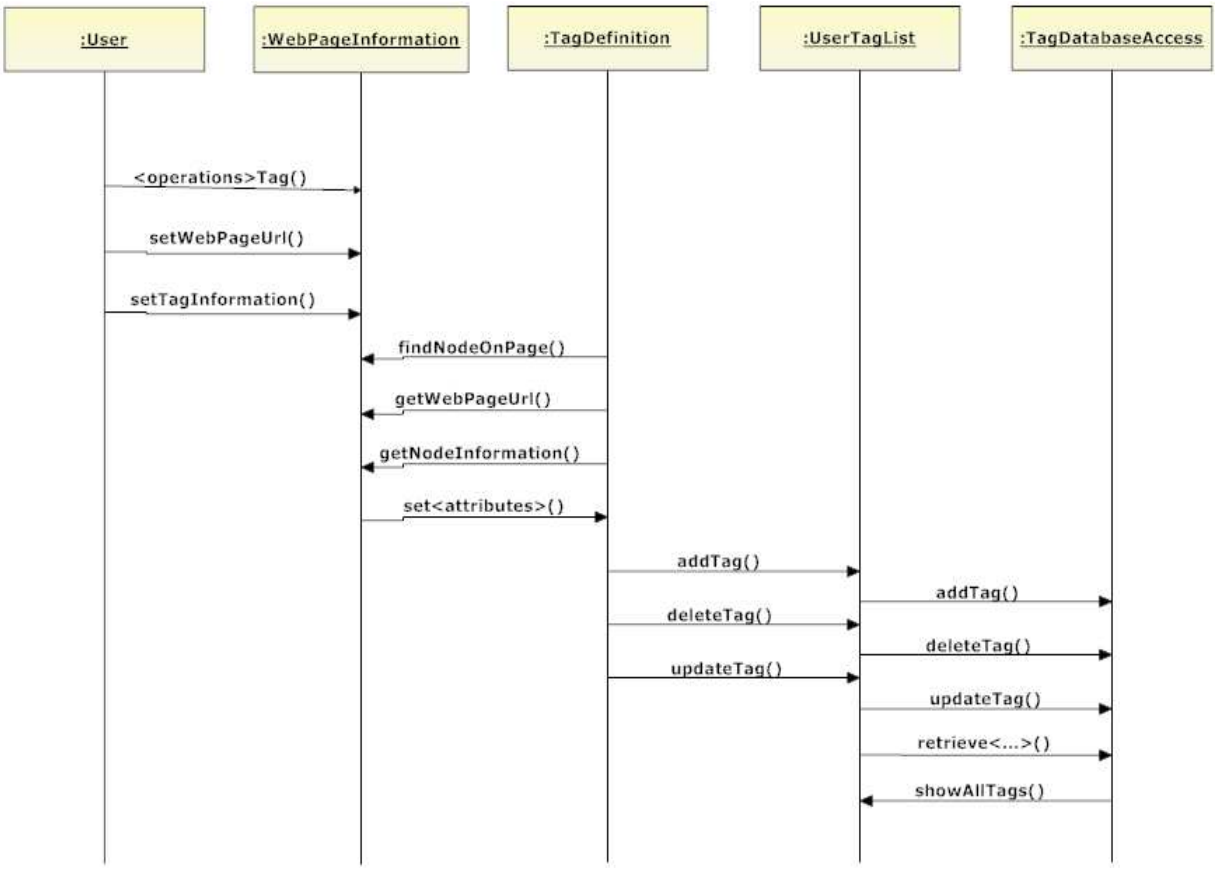



FIGURE 4.4.6.1: TAG OPERATIONS SEQUENCE DIAGRAM

## 4.4.7. RATE

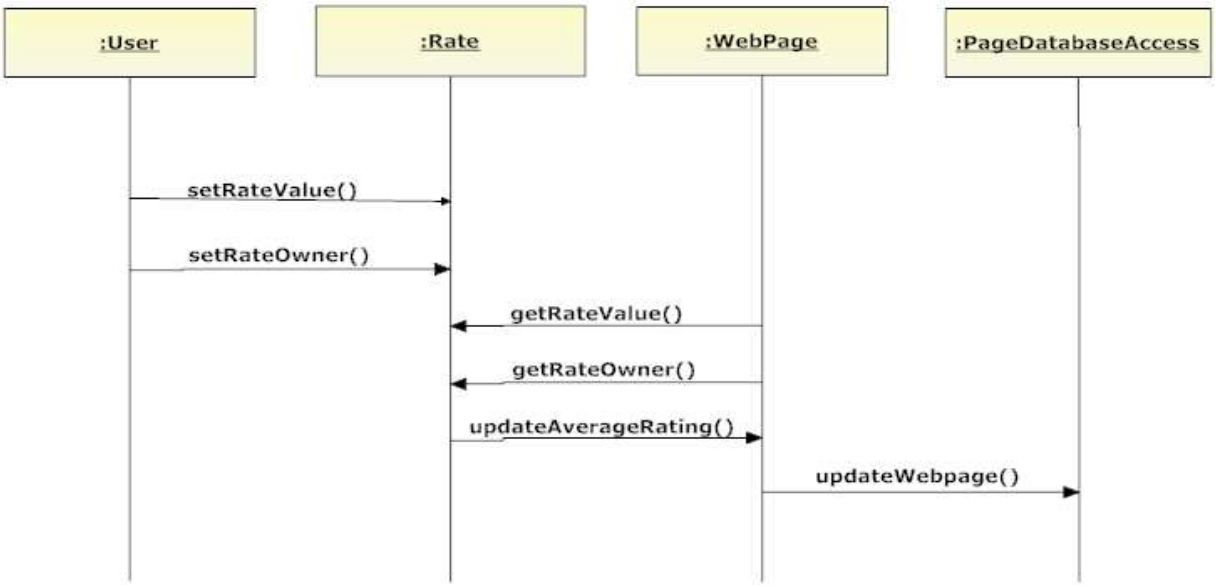



FIGURE 4.4.7.1: RATE SEQUENCE DIAGRAM

## 4.4.8. ACCOUNT DISPLAY

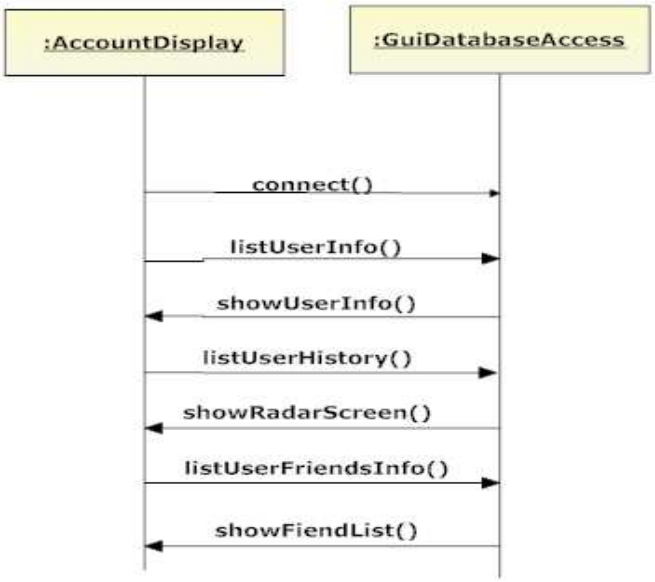



FIGURE 4.4.8.1: PERSONAL ACCOUNT DISPLAY SEQUENCE DIAGRAM

## 4.4.9. PLUG-IN DISPLAY

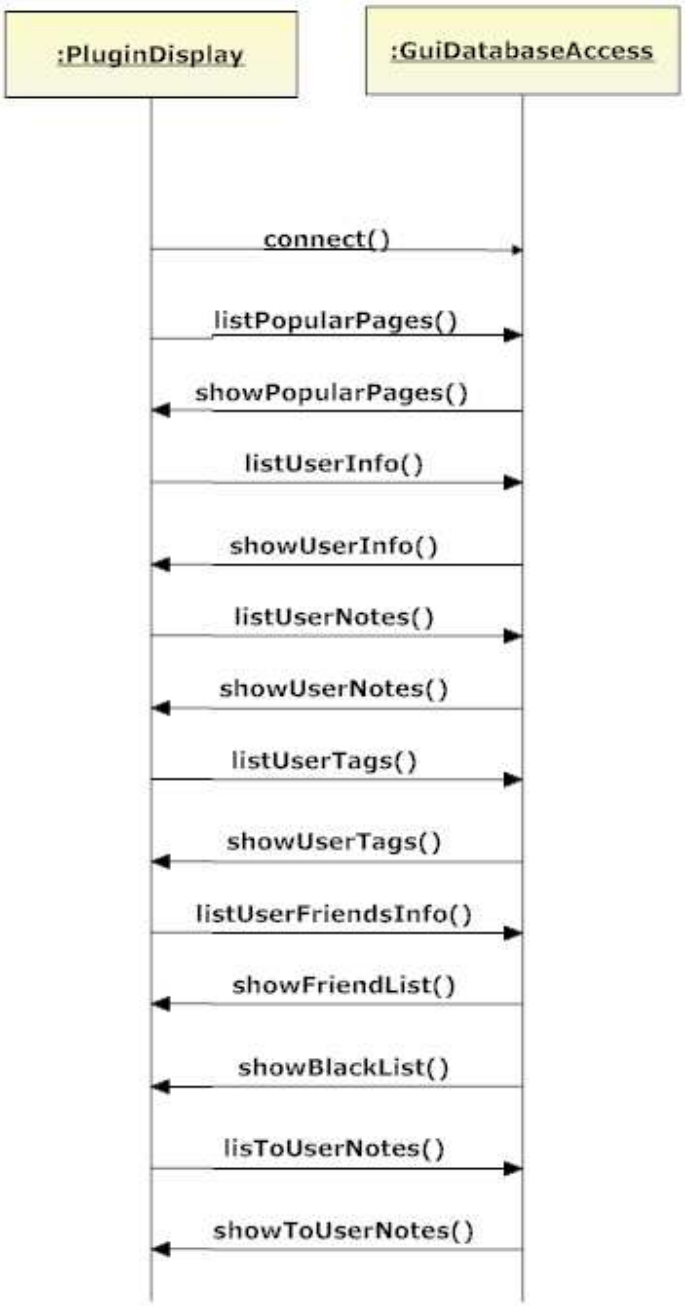



FIGURE 4.4.9.1: PLUG-IN DISPLAY SEQUENCE DIAGRAM

## 4.4.10. BROWSER DISPLAY

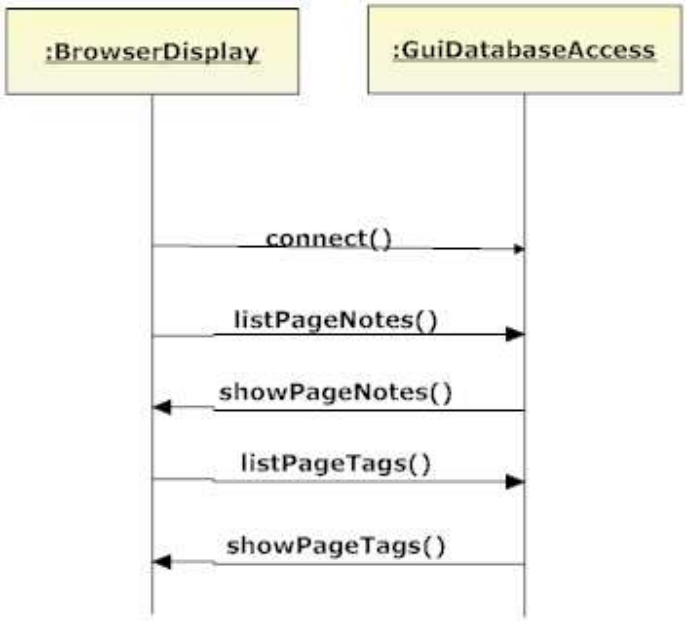



FIGURE 4.4.1.1: BROWSER DISPLAY SEQUENCE DIAGRAM

# 5. USER INTERFACE DESIGN

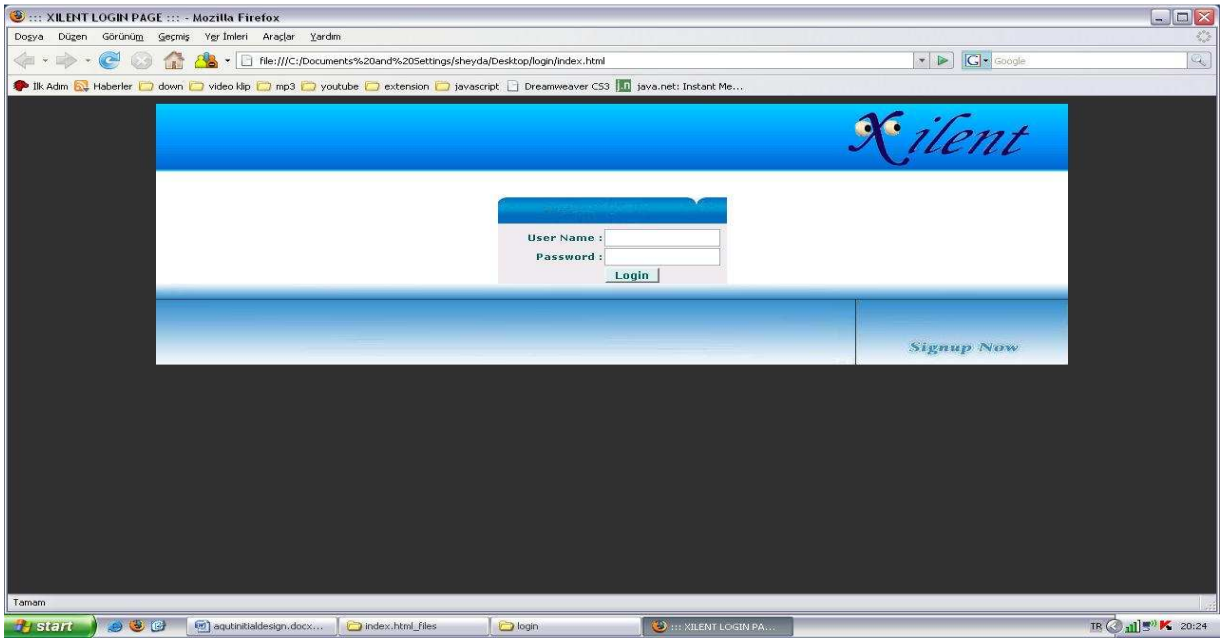



FIGURE 5.1: LOGIN SCREEN

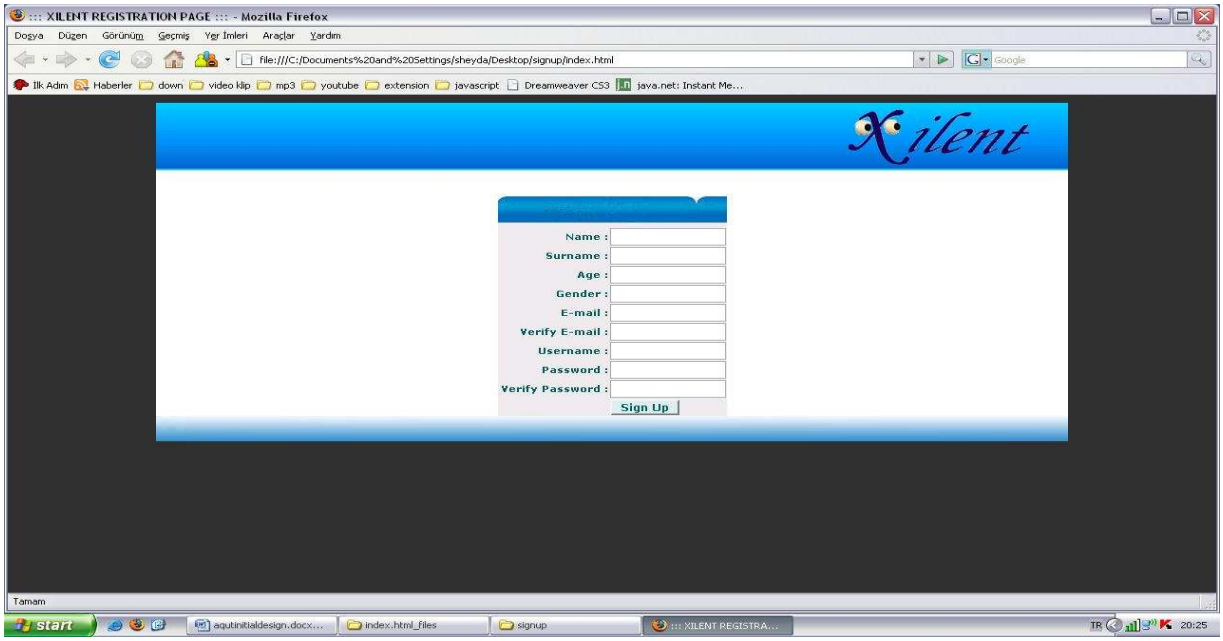



FIGURE 5.2: REGISTRATION SCREEN

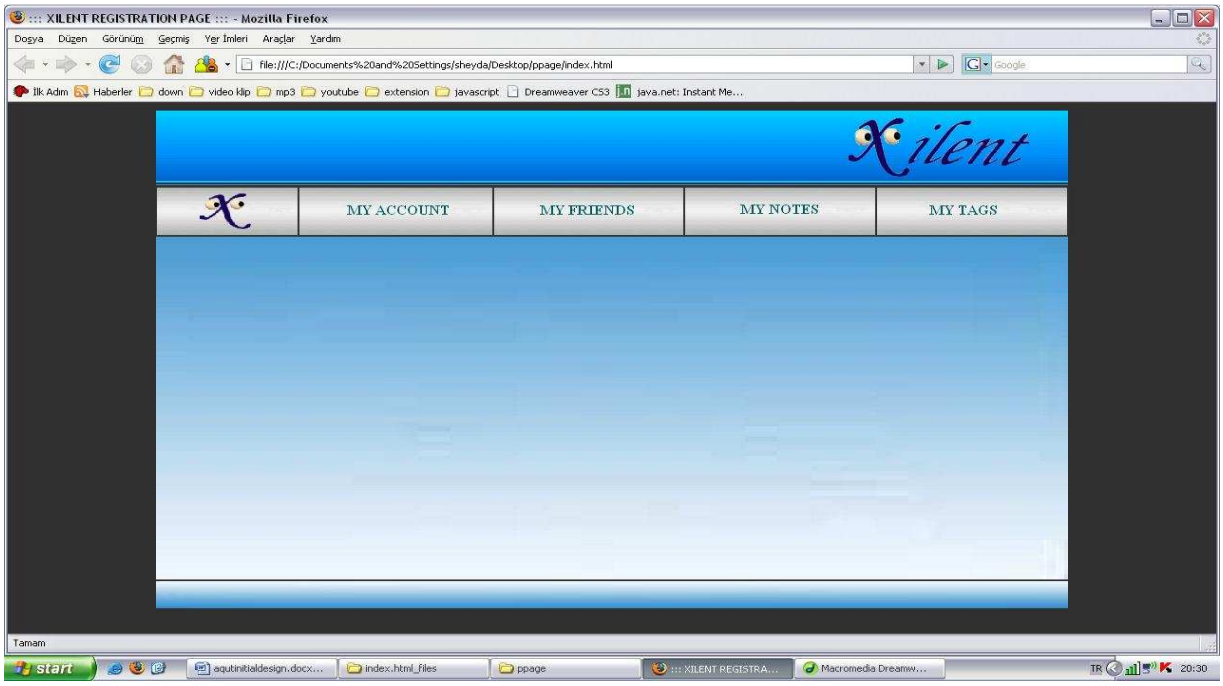



FIGURE 5.3: PERSONAL ACCOUNT SCREEN

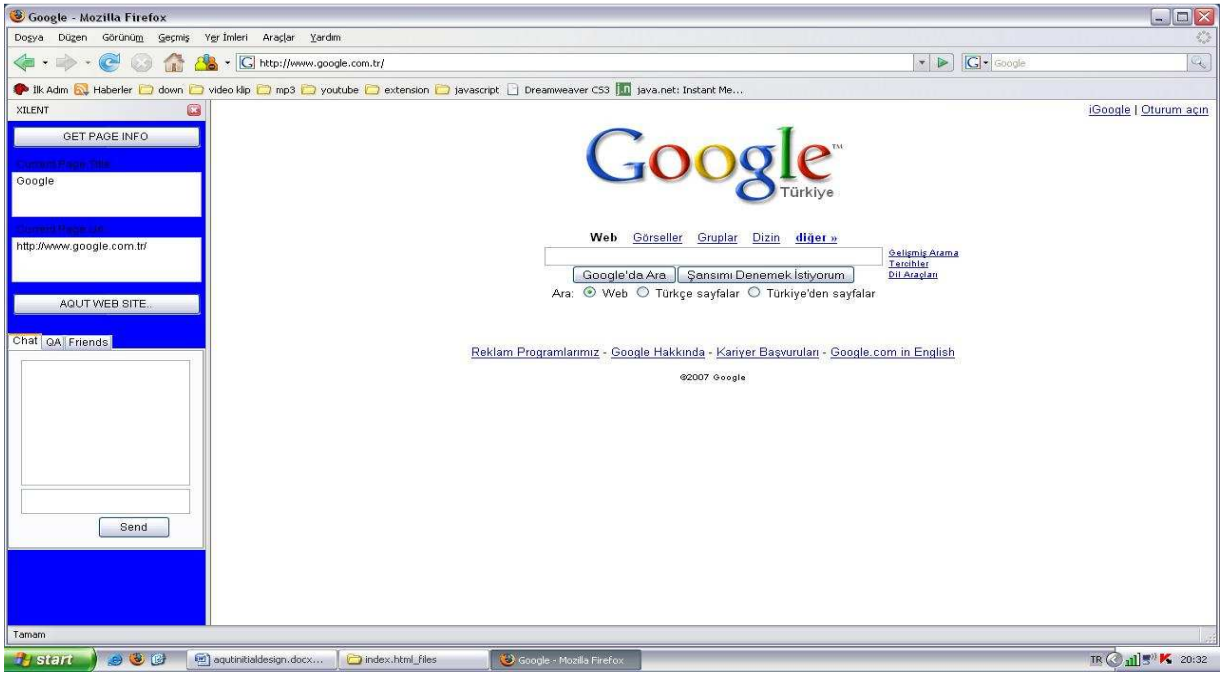



FIGURE 5.4: EXTENSION SCREEN. WHEN THE REGISTERED USER DOWNLOADS THE EXTENSION, A LEFT SIDED TOOLBAR WILL BE LOCATED TO THE BROWSER. THIS EXTENSION CAN GET THE WEBPAGE INFORMATION AND PROVIDE USERS A WEB MESSAGING ENVIRONMENT AS IT IS SEEN

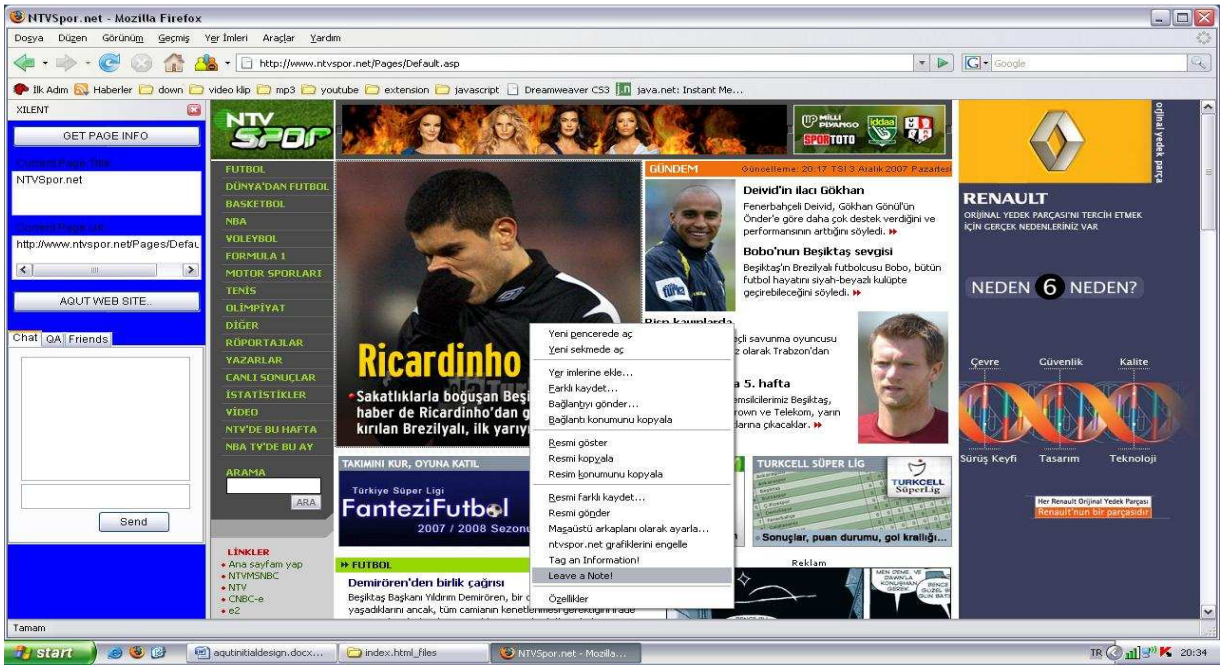



FIGURE 5.5: NOTE LEAVING SCREEN. USER CAN LEAVE A NOTE AT ANYWHERE ON THE WEBPAGE BY JUST RIGHT CLICKING AND SELECTING "LEAVE A NOTE" TAB FROM THE POP UP WINDOW

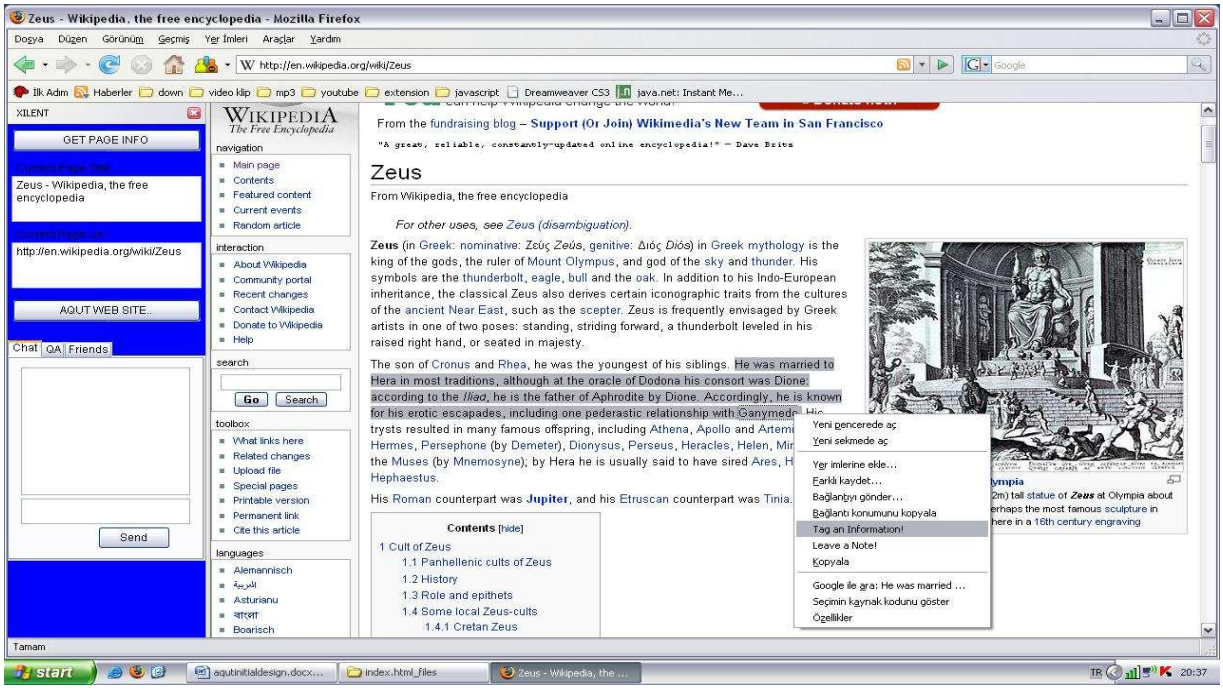



FIGURE 5.6: TAGGING SCREEN. USER CAN TAG ANY INFORMATION ON THE WEBPAGE BY JUST RIGHT CLICKING AND SELECTING "TAG AN INFORMATION" TAB FROM THE POP UP WINDOW

# 6. TESTING

Testing is indispensable phase while project is being developed. Project should be tested professionally before the customer delivery. Mainly, we divide our testing strategy into two parts that are dealing with each module individually and handling errors of the system entirely.

## 6.1. UNIT TESTING

In this step, we will test each module separately to identify errors that arises from them. Tests will be conducted by the code writers of that module. Functionalities will be tested carefully not to have trouble in later phases .We are planning to do this phase first and when we pass to the integration phase, we want to be sure that errors that we will face with are not related with internal structure of the modules.

## 6.2. INTEGRATION TESTING

After testing each module separately, we will integrate modules of the project. After this step, testing procedure may have another point of view since there may be errors that integration phase brings. We will examine relationship of each module with other modules and we will control whether they work within a harmony or not with our testing approach.

## 6.3. VALIDATION TESTING

After the integration step, we still may not be sure about the performance of the product. Therefore, we should go on testing the product by controlling its functionalities. Moreover, we are thinking to serve alpha and beta versions of the product to the customer to be aware of the satisfaction amount of the customer before final package.

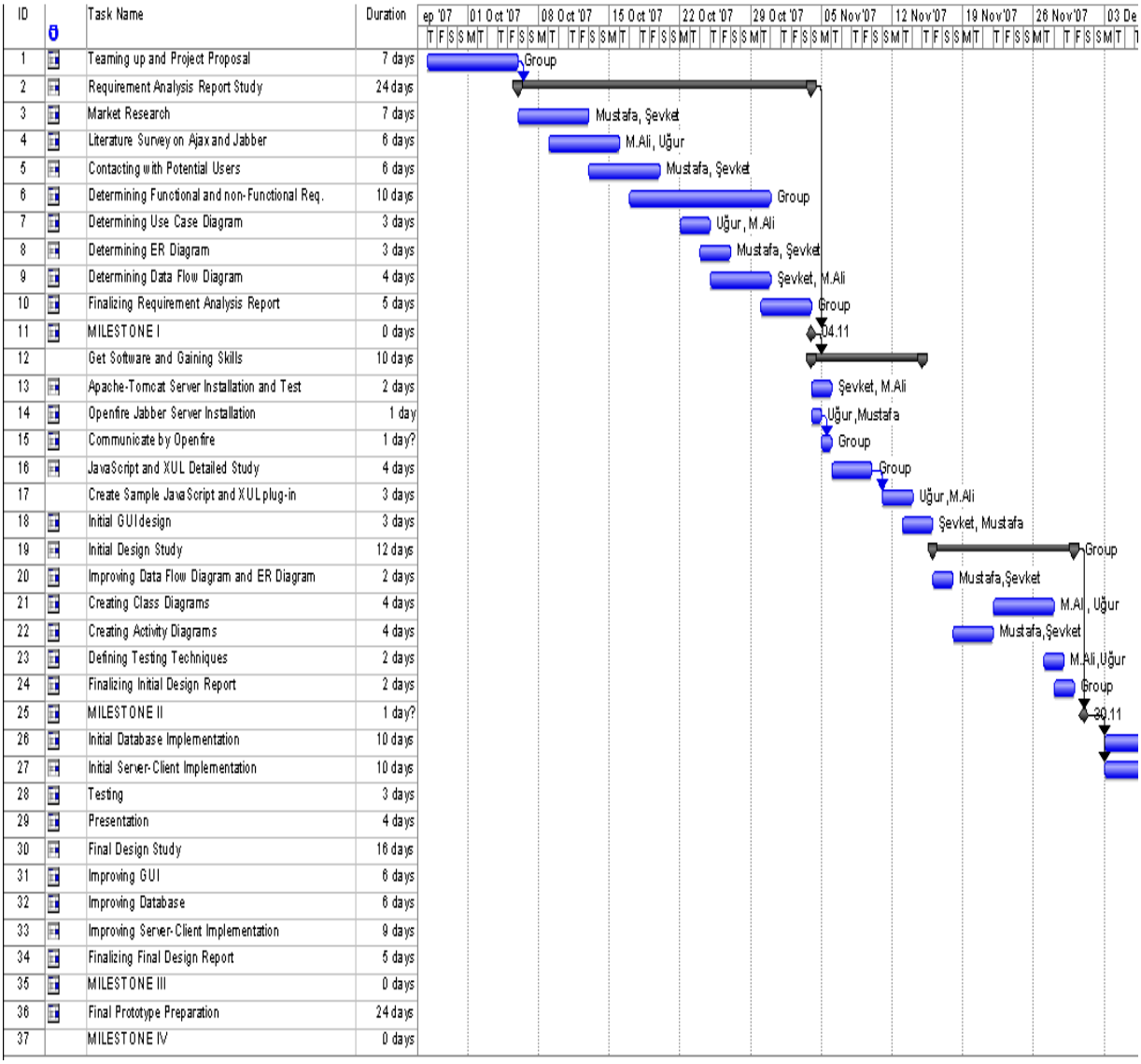
## 6.4. TESTING TECHNIQUES

In the early phases of the implementation, we will use white-box testing to figure out errors arises from implementation of the code. We will test every coded part not to face with bigger problems in later phases.

In later phases, we will use black-box testing techniques. We will test modules by examining fully functional requirements of the system as black-box testing technique states. Therefore, we will try all functionalities of modules to detect errors, missing and incorrect functions as much as possible. We are planning to handle errors of the system with this approach and prepare product for delivery.

Lastly, we are planning to deliver an executable of the project to some testers. We will want these testers to report us the errors that they face with. We will have a chance to test project on different environments with this approach.

# 7. GANTT CHART



| ID | Task Name | Duration |
|---|---|---|
| 1 | Teaming up and Project Proposal | 7 days |
| 2 | Requirement Analysis Report Study | 24 days |
| 3 | Market Research | 7 days |
| 4 | Literature Survey on Ajax and Jabber | 6 days |
| 5 | Contacting with Potential Users | 6 days |
| 6 | Determining Functional and non-Functional Req. | 10 days |
| 7 | Determining Use Case Diagram | 3 days |
| 8 | Determining ER Diagram | 3 days |
| 9 | Determining Data Flow Diagram | 4 days |
| 10 | Finalizing Requirement Analysis Report | 5 days |
| 11 | MILESTONE I | 0 days |
| 12 | Get Software and Gaining Skills | 10 days |
| 13 | Apache-Tomcat Server Installation and Test | 2 days |
| 14 | Openfire Jabber Server Installation | 1 day |
| 15 | Communicate by Openfire | 1 day? |
| 16 | JavaScript and XUL Detailed Study | 4 days |
| 17 | Create Sample JavaScript and XUL plug-in | 3 days |
| 18 | Initial GUI design | 3 days |
| 19 | Initial Design Study | 12 days |
| 20 | Improving Data Flow Diagram and ER Diagram | 2 days |
| 21 | Creating Class Diagrams | 4 days |
| 22 | Creating Activity Diagrams | 4 days |
| 23 | Defining Testing Techniques | 2 days |
| 24 | Finalizing Initial Design Report | 2 days |
| 25 | MILESTONE II | 1 day? |
| 26 | Initial Database Implementation | 10 days |
| 27 | Initial Server-Client Implementation | 10 days |
| 28 | Testing | 3 days |
| 29 | Presentation | 4 days |
| 30 | Final Design Study | 16 days |
| 31 | Improving GUI | 6 days |
| 32 | Improving Database | 6 days |
| 33 | Improving Server-Client Implementation | 9 days |
| 34 | Finalizing Final Design Report | 5 days |
| 35 | MILESTONE III | 0 days |
| 36 | Final Prototype Preparation | 24 days |
| 37 | MILESTONE IV | 0 days |



FIGURE 7.1: GANTT CHART PART I (FIRST SEMESTER)

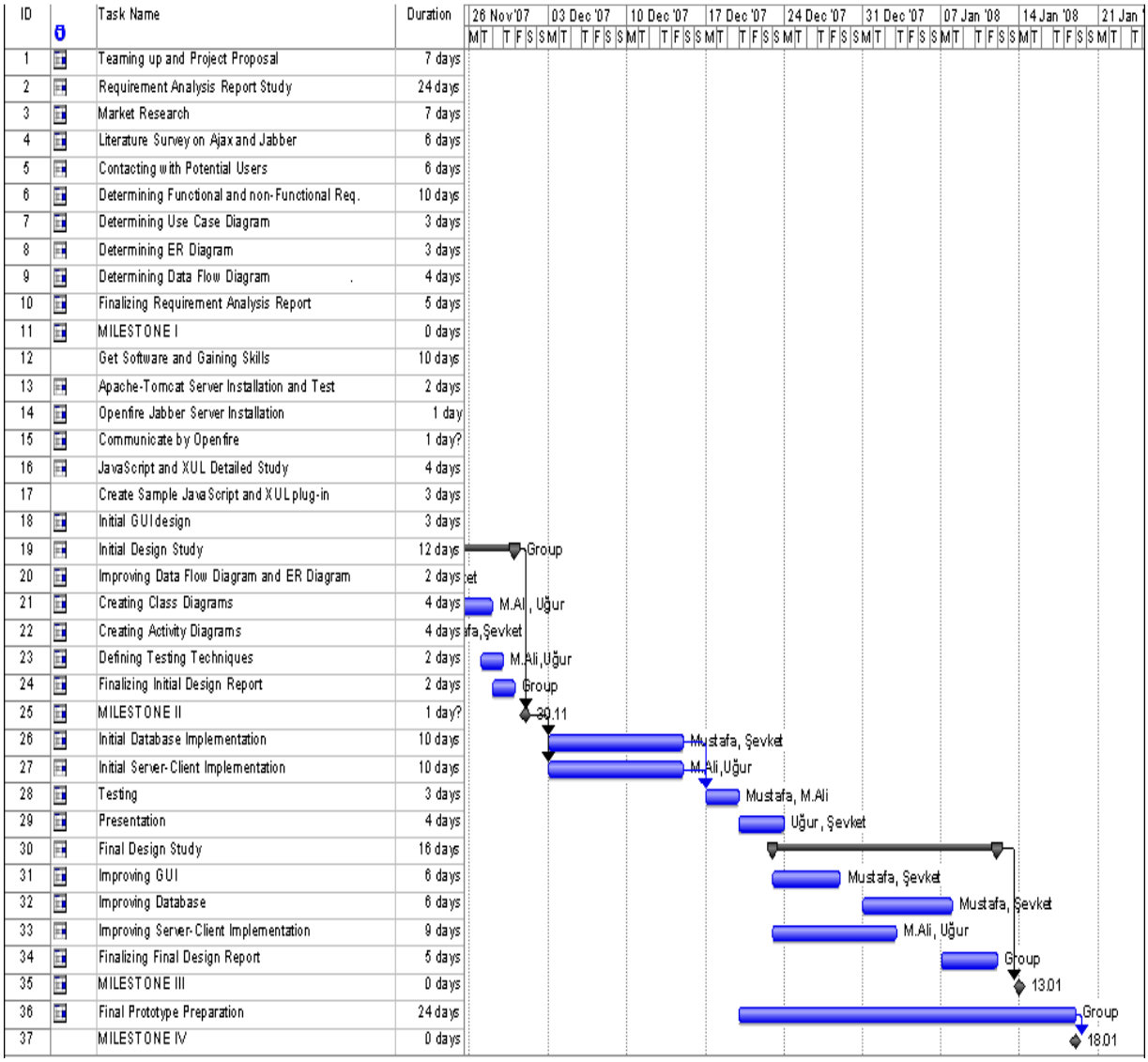



FIGURE 7.2: GANTT CHART PART II (FIRST SEMESTER)

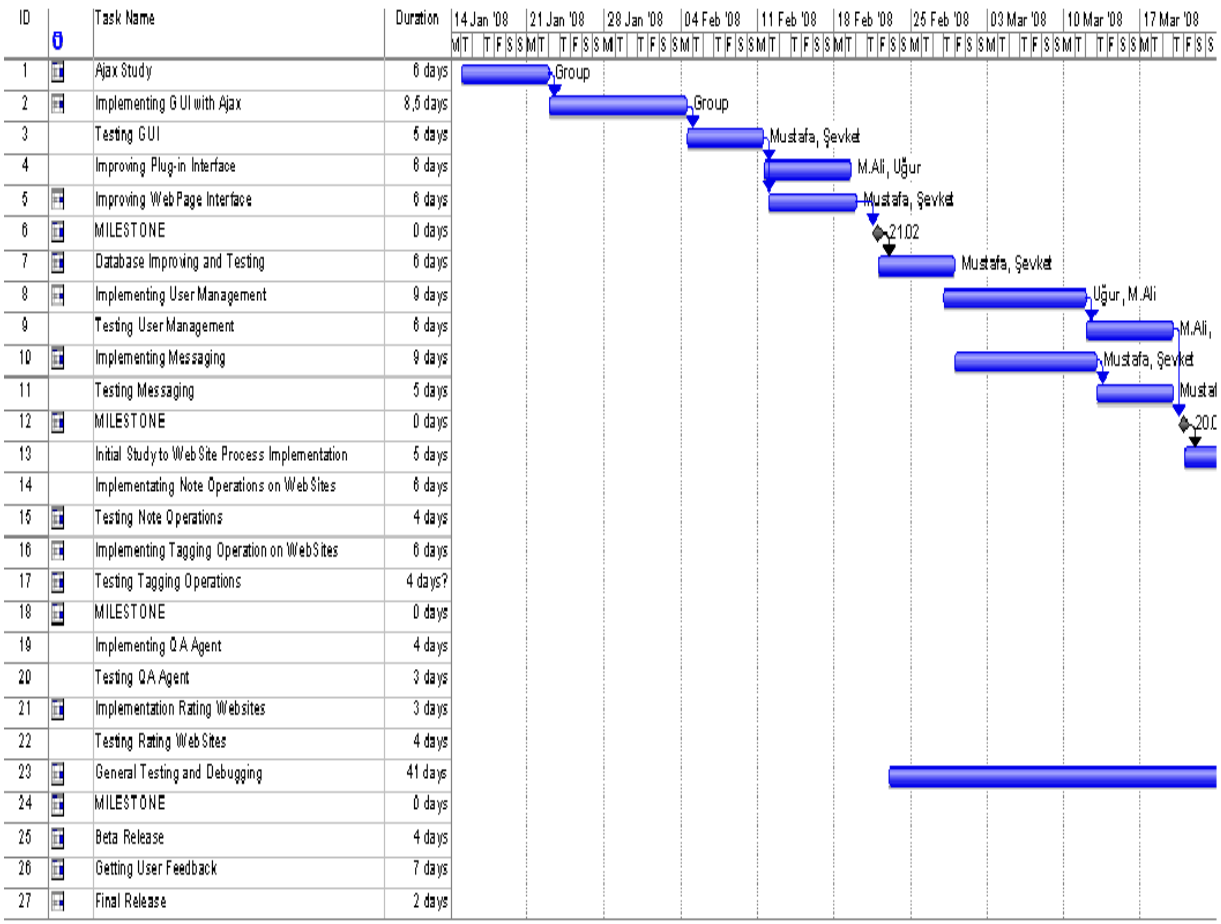



FIGURE 7.3: GANTT CHART PART I (SECOND SEMESTER)

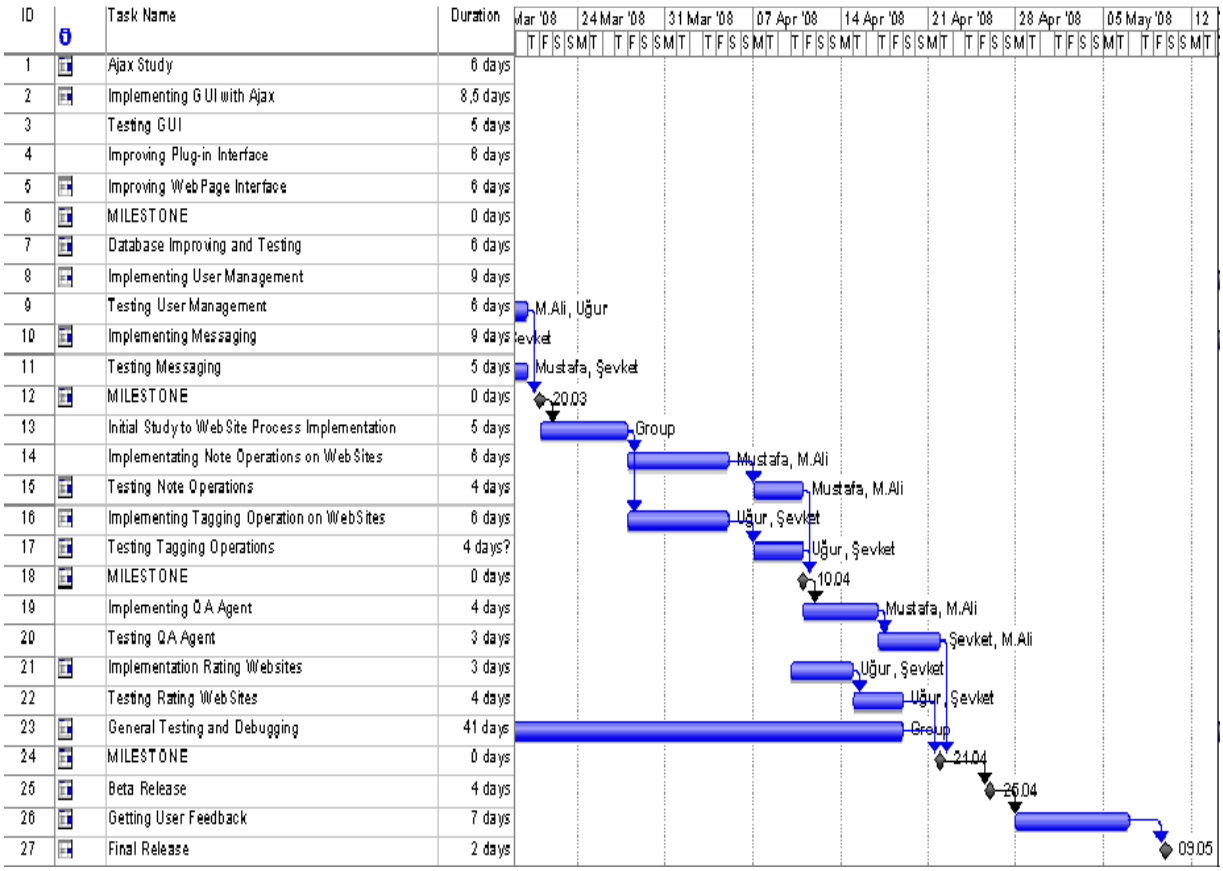



FIGURE 7.4: GANTT CHART PART II (SECOND SEMESTER)

# 8. CONCLUSION

Date from the day that we started to prepare the initial design report, we all know the importance of this report for the later phases. All the team members were aware of the stage being crucial and try to work in a much disciplined way to be successful. Xilent members worked very hard to complete their stuff. Now, as a team, we expect that, all our approaches to solve the problems are well understood and all parts of the report are clear enough. First of all, having drawn the class diagrams, Xilent members have now concrete conceptions for the coding phase of the project. Moreover sequence and activity diagrams are become very valuable from the implementation point of view.

At the end, we believe that giving much importance to design issues, starting from the first day will make our job easier for later phases. This initial design report will be modified when there are more efficient perspectives, and by this way will guide us through the end of the project.

# 9. REFERENCES

Component Oriented Software Engineering, Ali H. Doğru, TheATLAS Publishing 2006

Software Engineering A practitioner's Approach, Roger S. Pressman, McGRAW-HILL INTERNATIONAL EDITION 2001