Middle East Technical University

Department of Computer Engineering

`A Unified News Exchange Server `

Initial Design Report

Goncagül DEMİRDİZEN

Hilal KARAMAN

Ali Anıl SINACI

Ferhat ŞAHİNKAYA

"NewsAgent"

by

i$T€ Yazılım

Fall, 2006

1 INTRODUCTION

We had specified our requirements in our requirements analysis report and in the light of our requirements analysis reviews, we have prepared the initial design of our project. In initial design of *NewsAgent* we have understood the details and different aspects of the project more clearly and the system began to visualize in our minds precisely. In the design process, we aimed to design an efficient and modular system which satisfies all concept of the problem and tried to develop practical and applicable solutions to the problem. According to this purpose, we present the initial design of our system in this report. We aimed to visualize and embody our system with the help of different kinds of diagrams and specifications. We believe that all concept of the project is stable in our minds and we will try to develop the details of the design successfully in the light of our initial approaches to *NewsAgent.*

1.1 Project Scope & Definition

Communication has always been a significant aspect in human beings' lives. As the time passes and technology evolves, it appears with different usages and new techniques are discovered for serving communication. Accordingly, after Internet has started to be used widely, communication became one of the most important usage areas of it, especially electronic mails and online chat. Nowadays, most people use mailing lists, newsgroups or web forums for communication and reaching data about a specific issue. Definitely, these ways are more practical for now, when compared with searching whole Internet for a specific data. For this reason, handling different access methods to data is very significant for a news server. In fact, that is the reason for developing *NewsAgent*.

NewsAgent will provide users to reach data through web, tin, e-mail and news clients or via e-mail and RSS options will provide user to reach data in a fast and consistent manner. Furthermore, we can say that when *NewsAgent* takes its place in the market, users will feel the comfortable way of reaching data from different platforms.

1.2 Project Description

NewsAgent will contain several components, each of which will address different methods for communication. Each component will provide a different platform for communication and we can differ each user by the component that he/she used. For this reason, *NewsAgent* users can be named as NNTP user, RSS user, Web user, Mail user and administrator. Here are some general features that will be in *NewsAgent*:

- Administrators will be people who are responsible from the management of newsgroups, users and usergroups. Creating, removing new newsgroups or handling of undesirable articles in any of the newsgroups will be in the scope of his/her responsibilities. Moreover, they also deal with user management. When a candidate user requests to be a user of our system, administrators will be responsible to accept or reject their request and adding, deleting user and modifying user rights will also be responsibilities of administrators.

- Web users will be able to access newsgroups and articles through a graphical user interface. Web user will login to the system and after this authentication they will be able to realize all article-based and newsgroup-based operations according to their access level. An unauthenticated web user will be able to realize only part of these operations since their access levels will cover a small set of these operations. Web component will also provide management facilities for each user such as update user info, change login info etc. and a user-friendly interface will provide user to reach data, quickly.

- NNTP users will be able to access newsgroups through tin or NNTP clients, like Mozilla, Thunderbird or Microsoft Outlook Express. They will also be separated as authenticated and unauthenticated NNTP users. Authenticated NNTP users will be able to realize all article-based and newsgroup-based operations according to their access level. Unauthenticated NNTP users will be able to realize only part of these operations.

- RSS users will be able to receive feeds from newsgroups according to their wishes. We will create separate feeds for every newsgroup and whenever, a new article is posted we will add this article as a new item to our feed tree of the related newsgroup and we will serialize it. We will also delete the old items in the feed and RSS users will be able to access new data via their RSS readers.

- We will present a mailing option for our users and users will be able to set / reset their mailing option and as a result e-mails will be sent to these users if they want to receive post in a newsgroup via e-mail. Mail users will be able to receive mails from different newsgroups according to their wishes. Whenever a new article is posted, e-mails related to that article will be sent to the users who request to receive e-mail from that newsgroup. Moreover, mail users will be able to send posts to newsgroups as a new thread or as a follow-up. When the user sends mail to the system we will check the user is registered and the e-mails from registered users will be insert as articles into newsgroups.

- *NewsAgent* will contain several user groups and each user group will have different access rights. Authentication will specify access rights of each user and user will be able to access different newsgroups according to their rights and newsgroups that they are subscribed. In addition to user groups, also there will be a general access right which will not need authentication and user will be able to access some subset of newsgroups which is specified by the system administrator.

2 NEWSAGENT MODULES

2.1 Web Module

In our web interface, we will display some newsgroups which can be accessed by authenticated and unauthenticated users. Unauthenticated users will only request to read the articles in these groups. If the user is unregistered, a sign up will be requested to get authenticated. If the user is registered, the following functionalities will be provided.

- The user logins to the system by entering username and password and after authentication check, the user group of the user is specified and the user will have the rights according to the user group. Each user group will have different rights and restrictions.
- Users can sign up only through web module and a randomly generated password is sent to the user via e-mail for verification of the candidate user. After the verification, user can start to reach articles on the news server by using his/her username & password.
- Update user info and account info functionalities will be supported and the user will be able to change this information.
- Read, post, update, cancel article functionalities will be provided and the user will have the right to update and cancel only the articles that he/she has posted.
- Mail receiving options will also be adjusted in the web module and a user may request to receive e-mail for the articles of the adjusted newsgroups.
- Listing the newsgroups and sorting the newsgroups and articles according to the specified criteria such as according to names, dates of the articles etc.

2.2 NNTP Module

Similar to the Web module, users are classified as authorized and unauthorized users. Unauthorized users can only reach only some subset of newsgroups, which are specified by system administrator. In fact, that is reasonable, since user group of unauthorized users has access level to only these newsgroups. If user is registered, the following functionalities will be provided to the user:

- The user login to the system by entering his/her username and password. Username and password are controlled for validation from the database. If username-password combination is not valid, display feedback is shown to the user which specifies incorrect username or password and user cannot enter the system. If this is not the case, user can enter the system and an access level is assigned to the user corresponding to the user group.
- User can update his/her account information according to his wishes. Since userid information is hidden from the user, same userid will again specify the user.
- Read, post, update, cancel article functionalities will be provided and the user will have the right to update and cancel only the articles that he/she has posted.
- Mail receiving options will be adjusted in the NNTP Module, setting this option on/off is the users' choice.
- Listing the newsgroups and sorting the newsgroups and articles according to the specified criteria such as according to names, dates of the articles etc.

2.3 Mail Module

- When our system receives an e-mail, first of all the system controls whether the sender is an authenticated mail client or not. If the sender is authenticated then the e-mail is converted to the article format and inserted to the database. The article will be added to a newsgroup which is specified in the address field of the mail content.
- User can reach articles in a newsgroup via e-mail depending on whether he/she set his mailing options on. Of course, user will be able to receive mail from only newsgroups which he/she can subscribe corresponding to his/her user group.

2.4 RSS/Atom Module

- If user want to follow a newsgroup periodically, user can subscribe to the RSS feed of this newsgroup and by using an RSS reader, he/she can reach articles that are newly posted to the newsgroup.

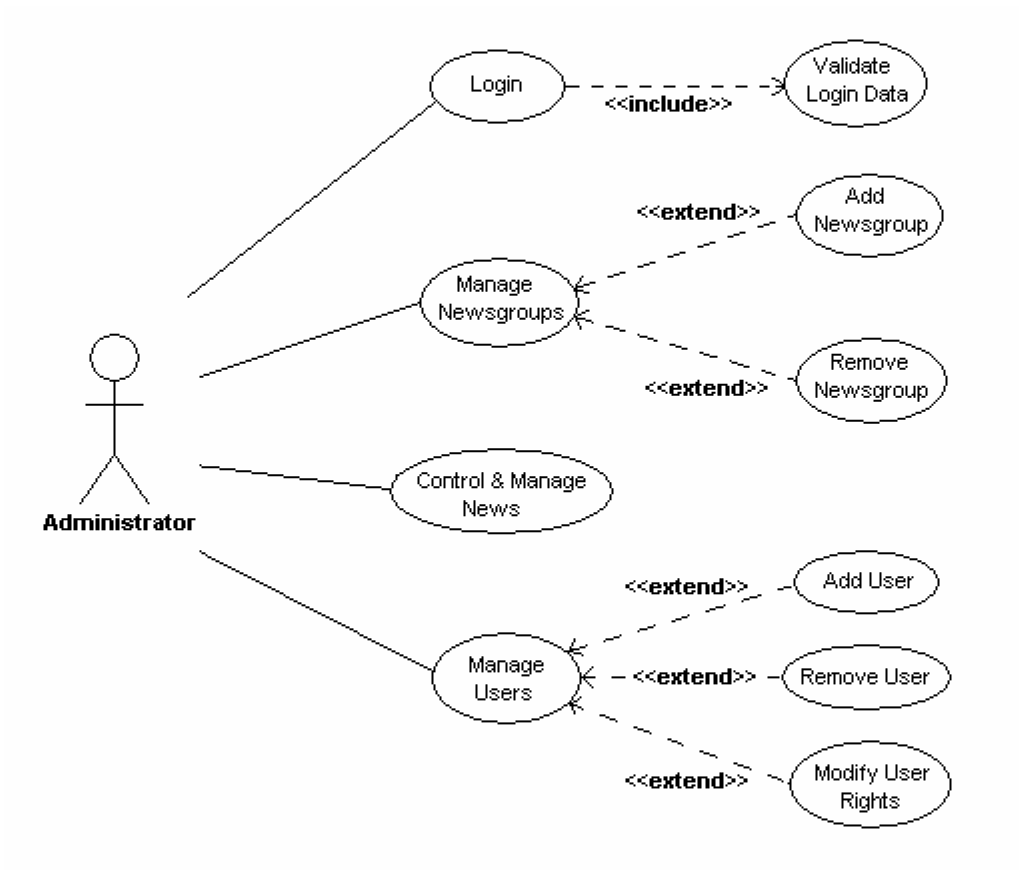
2.5 Authentication Module

- As mentioned in previous modules, each user will be in a user-group which specifies the access level of the user. During authentication username will be checked for specifying whether username is in database or not.
- Username and password will be checked for correspondence between them.
- For security reasons, password will be held in a MD5 (Message-Digest algorithm 5) [references: http://en.wikipedia.org/wiki/MD5] format. This hashing technique will prevent anyone to access passwords of the users, directly.
- User groups will be assigned for the user after his/her authentication. Since user group for each user is stored in the database which is assigned by system administrator, assignment of user groups is not a big deal.
- A user who is not authorized to the system will be able to access only some subset of newsgroups and read only articles in these newsgroups.

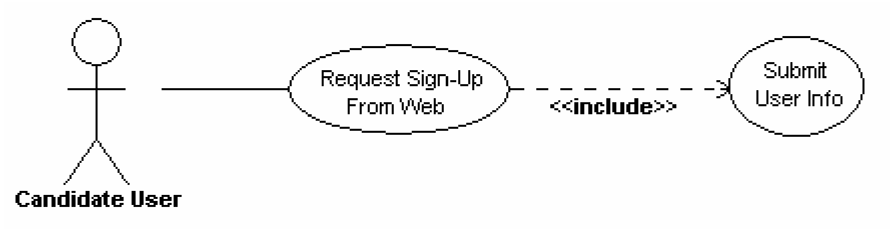
3 USE CASES

3.1 USE CASE DIAGRAMS

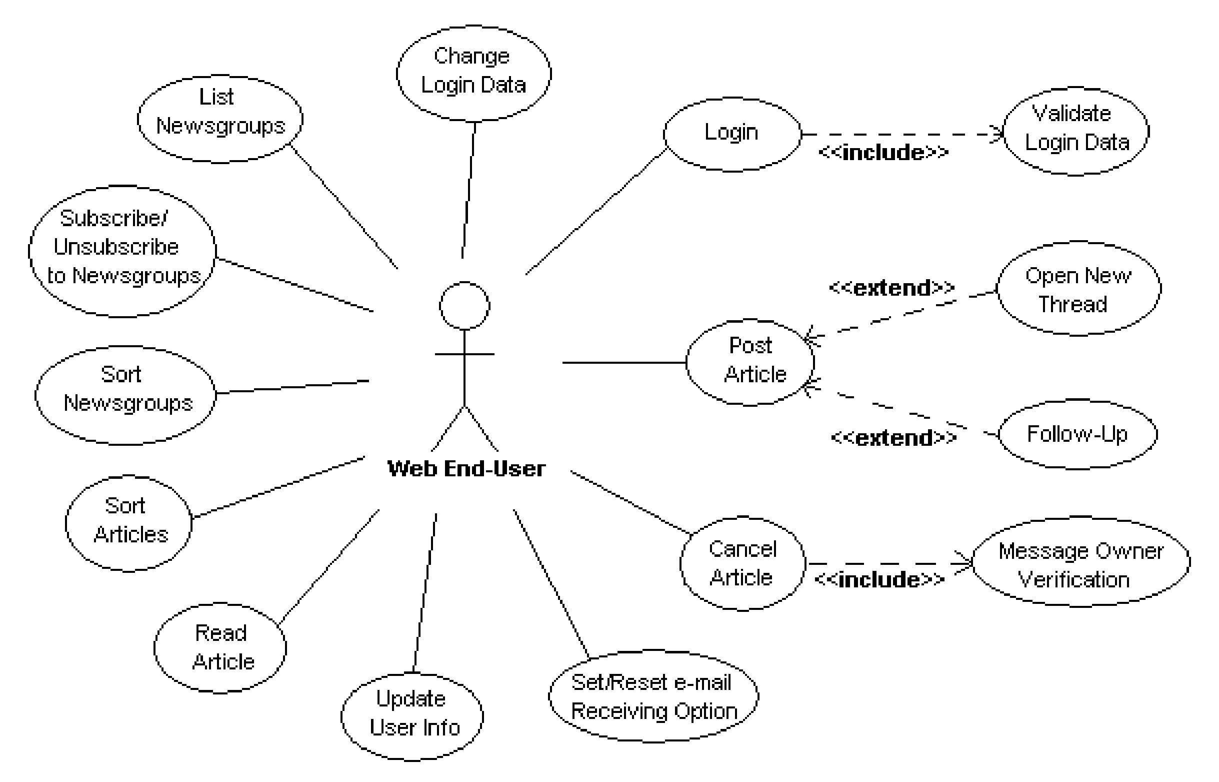
3.1.1 Use Case Diagram for Administrator



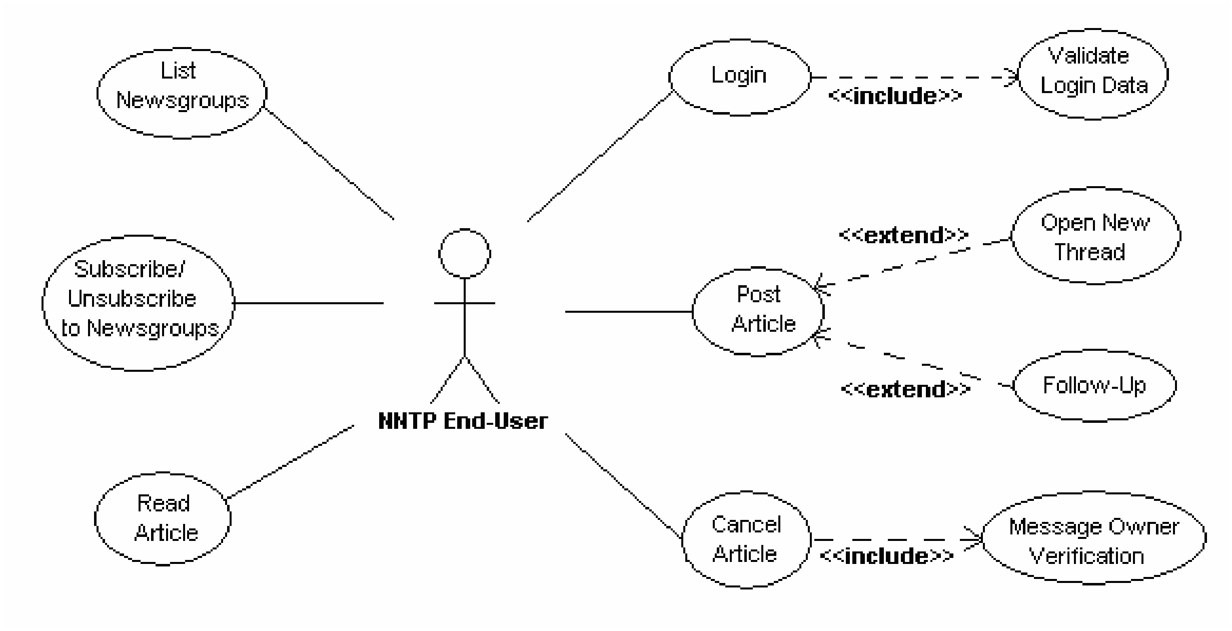
3.1.2 Use Case Diagram for Candidate User



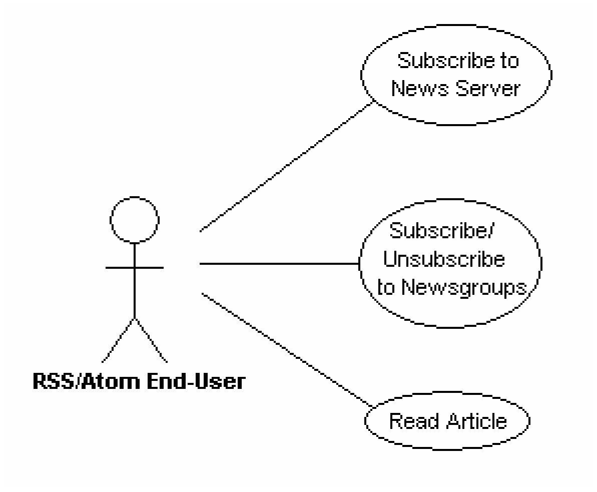
3.1.3 Use Case Diagram for Web End-User



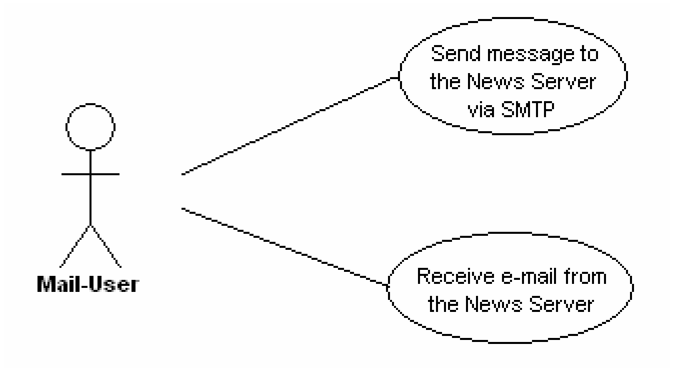
3.1.4 Use Case Diagram for NNTP End-User



3.1.5 Use Case Diagram for RSS/Atom End-User



3.1.6 Use Case Diagram for Mail-User


Send message to the News Server via SMTP
Receive e-mail from the News Server
Mail-User

3.2 USE CASE SCENARIOS

Administrator:

- ***Login:*** An administrator has to login to the system in order to realize administrative roles. There will be a web user interface for administrative roles. After validation of login information, the administrator will be able to manage newsgroups, users and news.
- ***Manage Newsgroups:*** Administrator may add new newsgroups and remove existing newsgroups in the content of the managing newsgroups scenario.
- ***Manage Users:*** Administrator may add and remove users and modify the user rights. Administrator will control users and will be able to restrict the user rights.

There will be specified user roles and rights, however, new rights can be granted to the users and existing rights may be withdrawn.

- ***Control & Manage News:*** An administrator will have the right of controlling and managing the articles. Articles which do not suit the content of the newsgroup may be cancelled. As a result of such a control on news, user roles and rights granted to the users defined more precisely.

<u>**Candidate User:**</u>

- ***Request Sign-up:*** A candidate user is a person who demands to sign up to the system via web interface and as a result of a sign-up request, the candidate user has to submit a user information form and if the administrators accept the request, the candidate user turns out to be a real system user.

<u>**Web End-User:**</u>

The scenarios which are valid for NNTP End-users are also valid for Web End-users. Moreover, Web End-users have extra usage scenarios. The followings are the extra usage scenarios for Web End-users.

- ***Set & Reset Mail Receiving Options:*** The user will be able to request to receive e-mail for the articles posted. The user may want to receive e-mail for specified newsgroups or want to receive e-mail for all newsgroups. Also the user may want to cancel the mail receiving option and then no e-mails will be sent to the user from that newsgroup.
- ***Update User Info:*** The user will be able to update user information such as his/her personal information registered when signing up, e-mail address etc.
- ***Change Login Data:*** The user may change login information. Generally user id of a user is not allowed to be changed for most of the systems however the users may need to change their passwords.

<u>NNTP End-User:</u>

- ***Login:*** The user will login to the system in order to realize user roles. After validation of user login information, the user will be able to list, subscribe/unsubscribe, sort newsgroups and post, read, cancel and sort articles.
- ***List Newsgroups:*** The user will be able to list the newsgroups. In the concept of listing newsgroups scenario, a user may list all newsgroups or the newsgroups that he/she has been subscribed.
- ***Subscribe / Unsubscribe to Newsgroups:*** After listing the newsgroups, the user will be able to subscribe and unsubscribe to the newsgroups.
- ***Post Article:*** The user posts articles. In the concept of posting articles, the user may open a new thread or follow up to an existing article.
- ***Cancel Article:*** The user may cancel the articles that he/she has posted.
- ***Read Article:*** The user reads articles.

<u>RSS/Atom End-User:</u>

- ***Subscribe to News Server:*** RSS/Atom end-users will subscribe to the news server in order to receive feeds from the server.
- ***Subscribe / Unsubscribe to Newsgroups:*** RSS/Atom end-users will be able to subscribe and unsubscribe to specific newsgroups. Each newsgroup will have its own feed so that the user receives only the news from subscribed newsgroups.
- ***Read Articles:*** As all users do, RSS users will read the news.

<u>Mail User</u>

When a user sets receiving mail option from web, that user becomes also a mail user.

- ***Send Message to the News Server:*** Mail users send messages to the server through SMTP protocol. This message appears in the same way as other messages do in the News Server.

- ***Receive e-mail from the News Server:*** When a new message is posted, mail users receive that message as e-mail from the newsgroups if they are subscribed to that group.

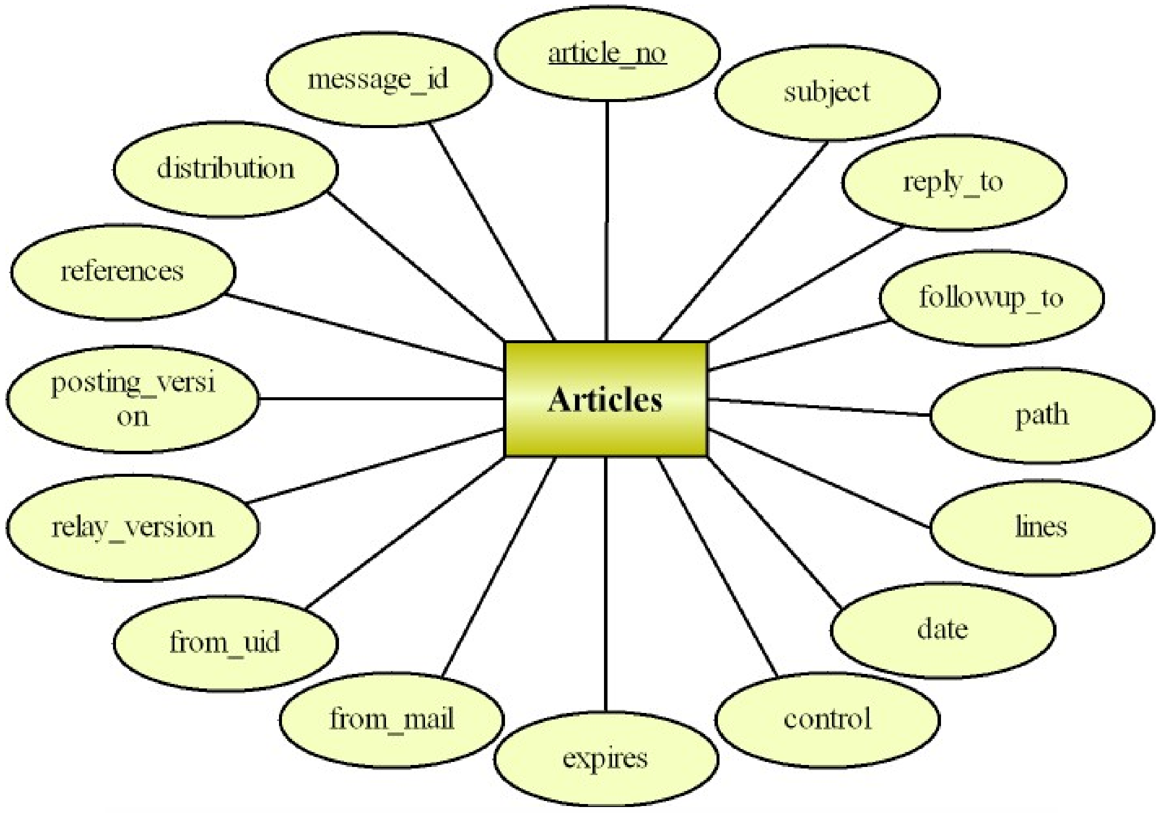
4 MODELLING

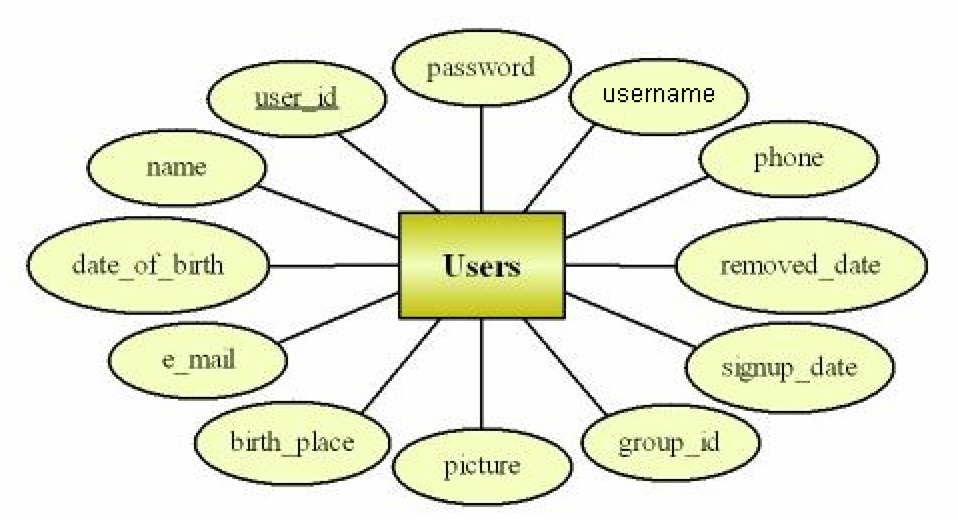
4.1 DATA MODELLING

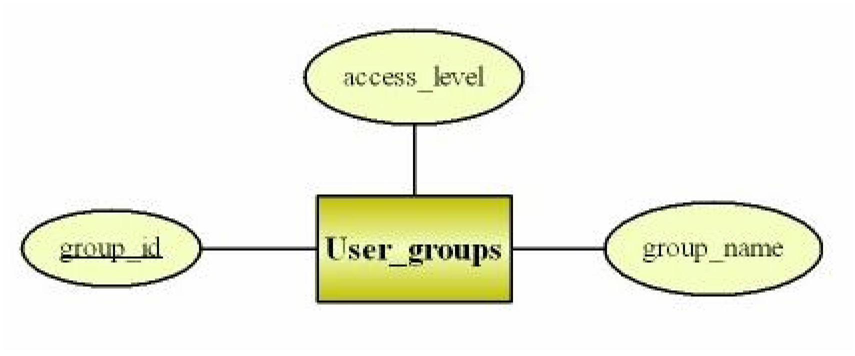
In our system, we will store our data in 2 different databases. The main database will be used to store main data such as articles, users, newsgroups, etc. Other database will be used as an archive to store older articles and deleted newsgroups. These older articles will not be stored in main database anymore. If any client requests an old article which is already moved to the archive database by NewsAgent, system finds the article from the archive database either by the message-id or server specific article number.

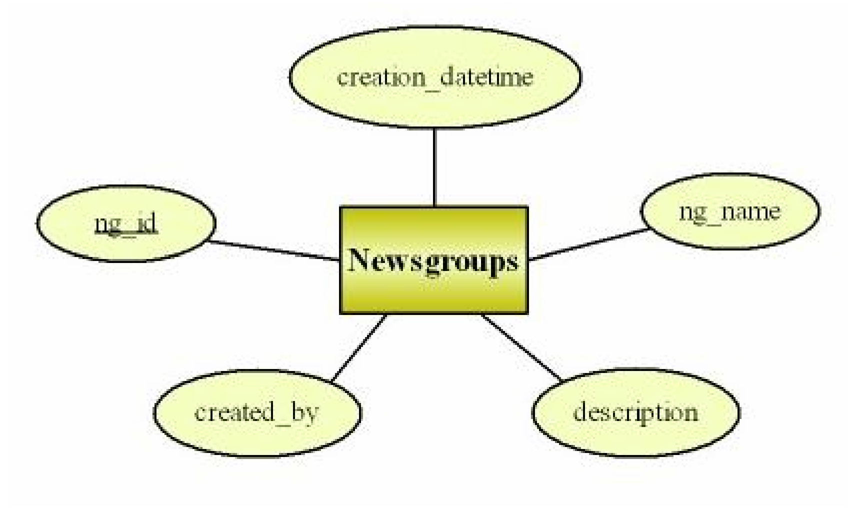
4.1.1 Entity-Relationship Diagrams

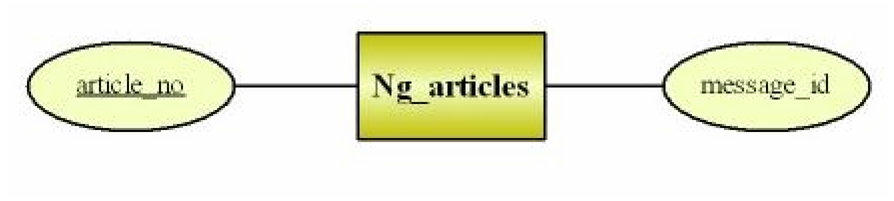
ER Diagrams for Main Database

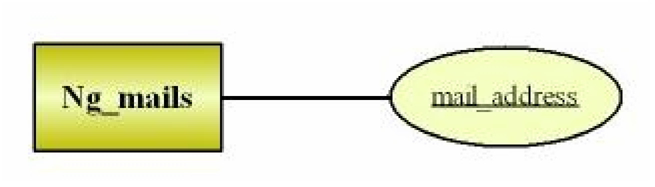


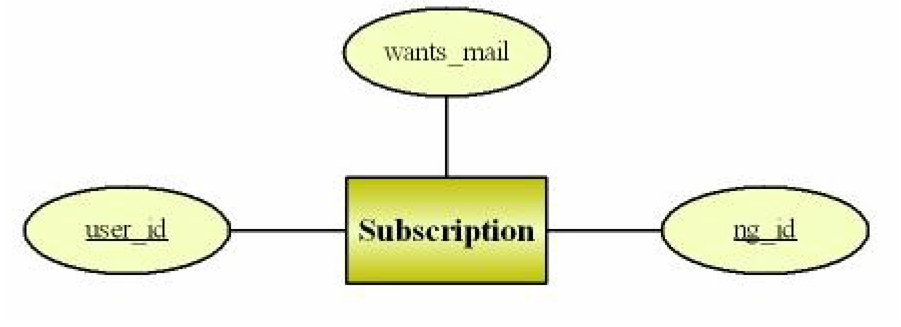


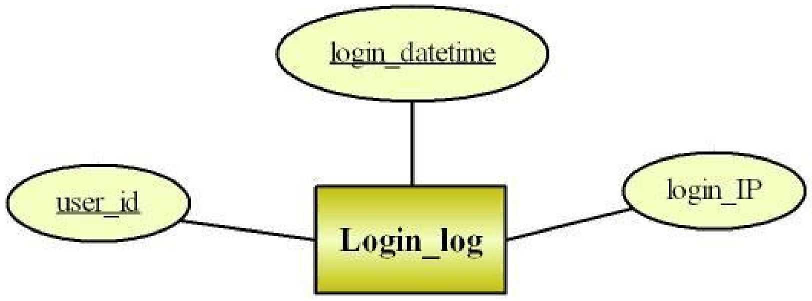

access_level
group_id
User_groups
group_name

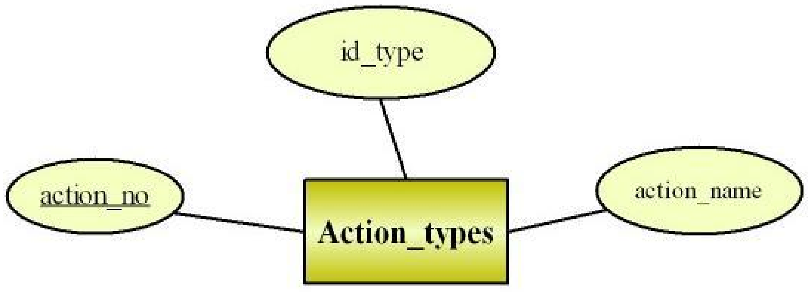

creation_datetime
ng_id
Newsgroups
ng_name
created_by
description

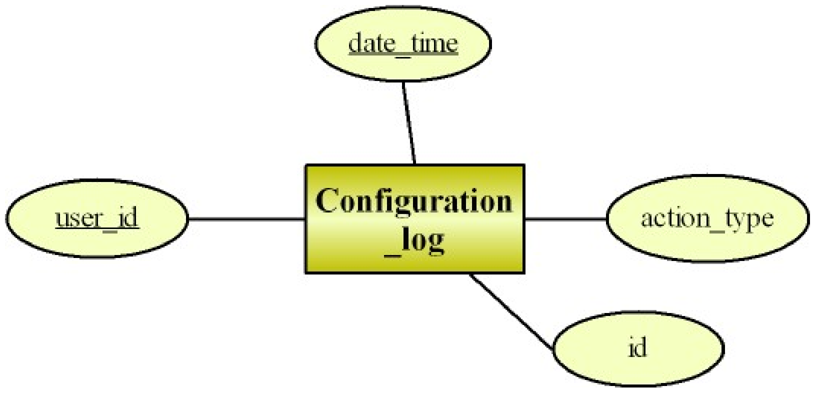

article_no
Ng_articles
message_id


Ng_mails
mail_address

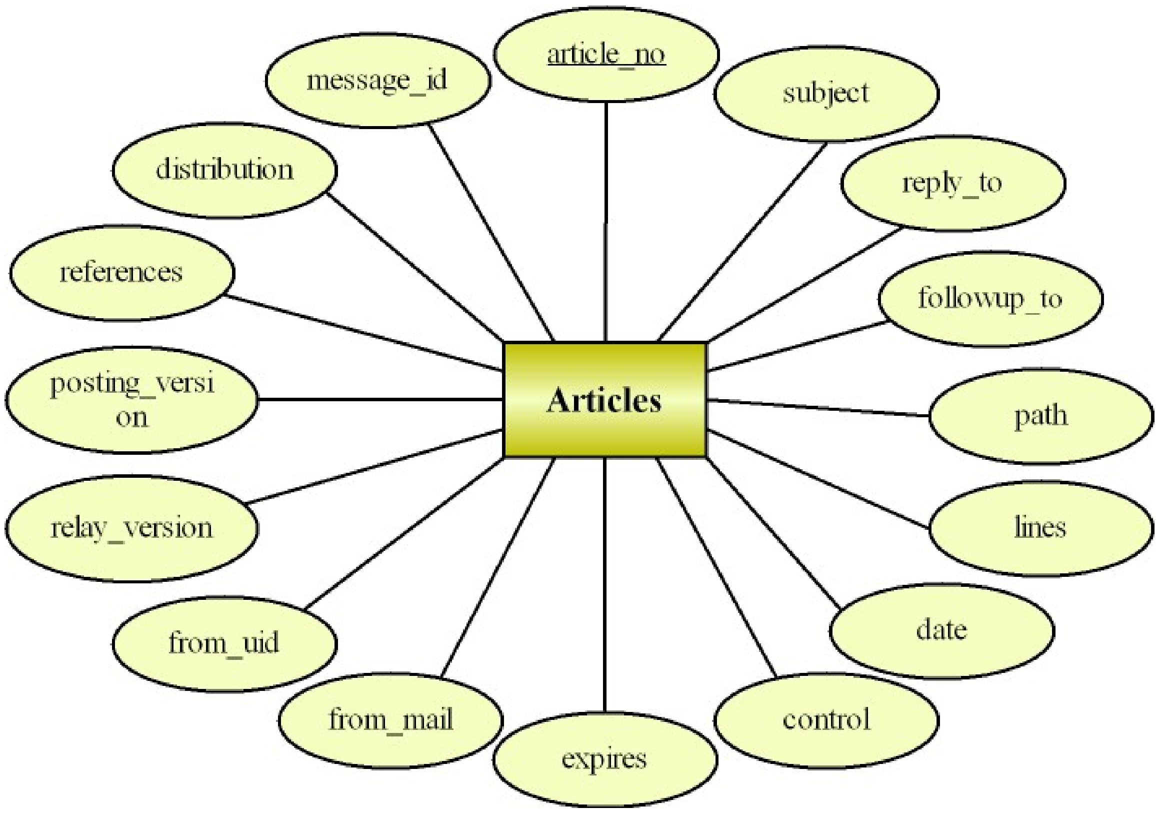

wants_mail
user_id
Subscription
ng_id

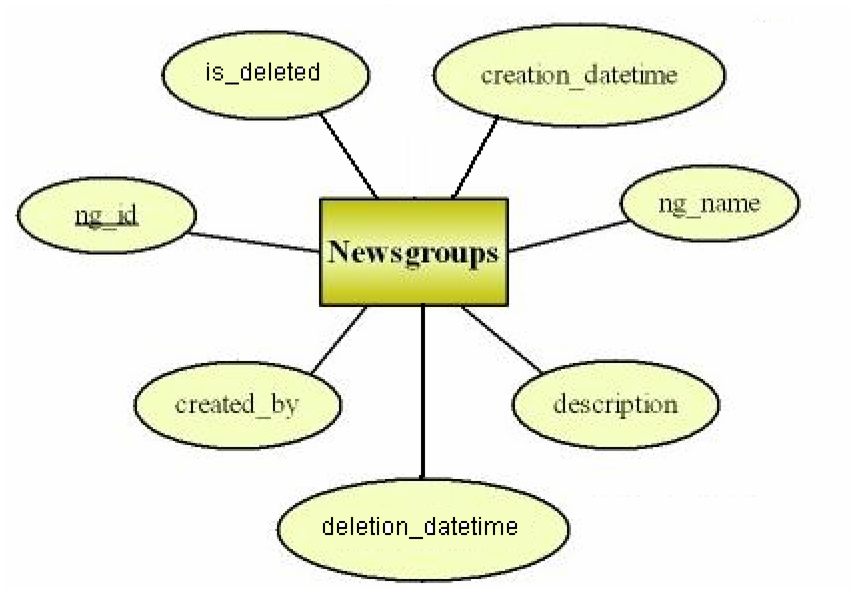

login_datetime
user_id
Login_log
login_IP

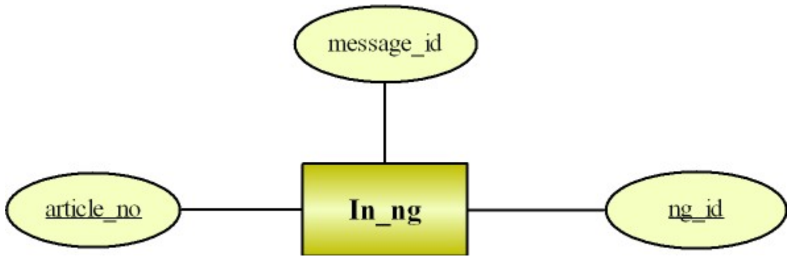

id_type
action_no
Action_types
action_name


date_time
user_id
Configuration _log
action_type
id

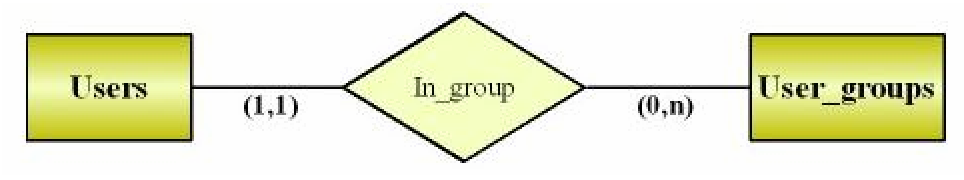
ER Diagrams for Archive Database

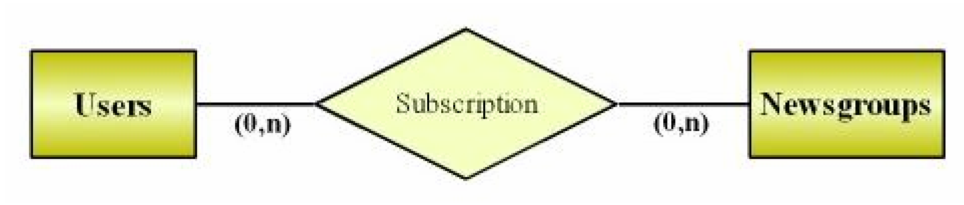


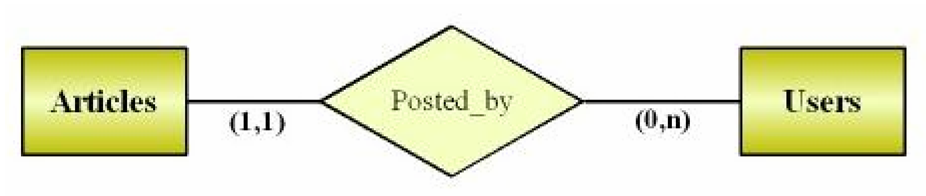


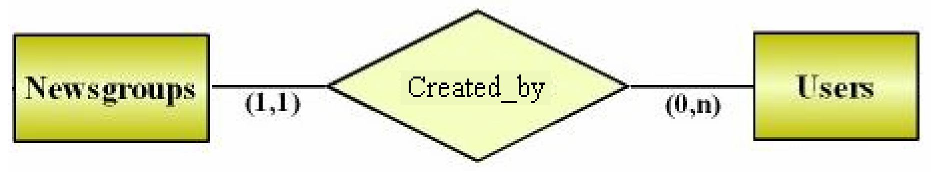

message_id
article_no
In_ng
ng_id

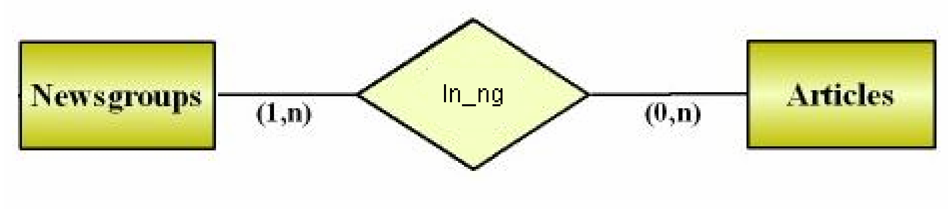
Relations


Users
(1,1)
In_group
(0,n)
User_groups

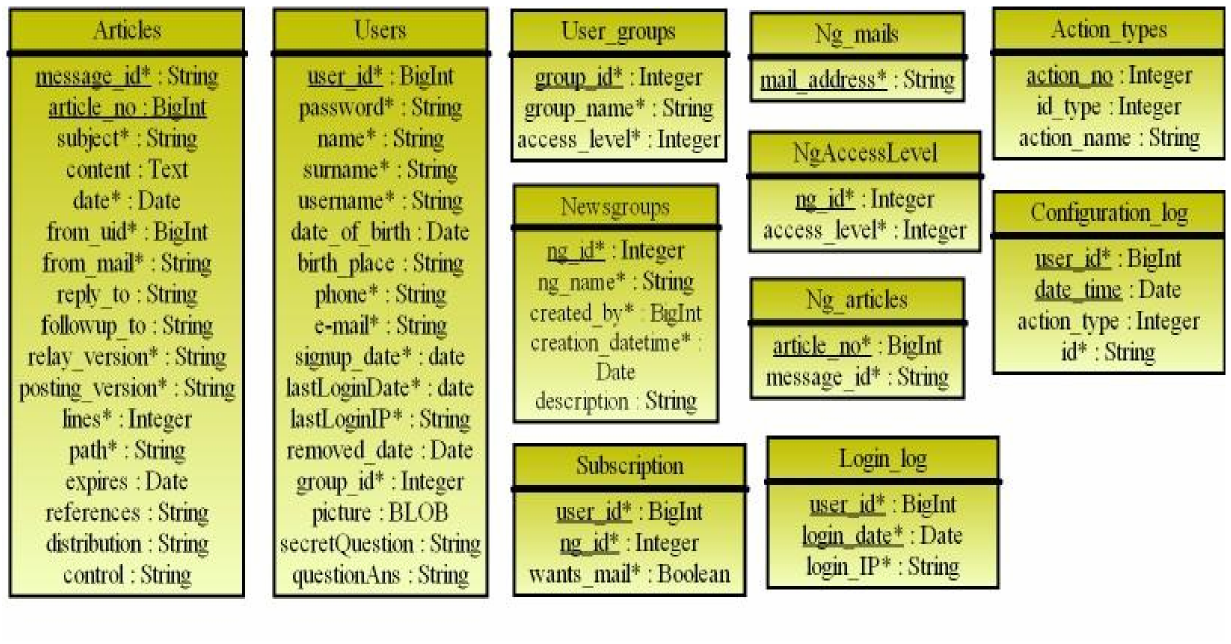

Users
(0,n)
Subscription
(0,n)
Newsgroups

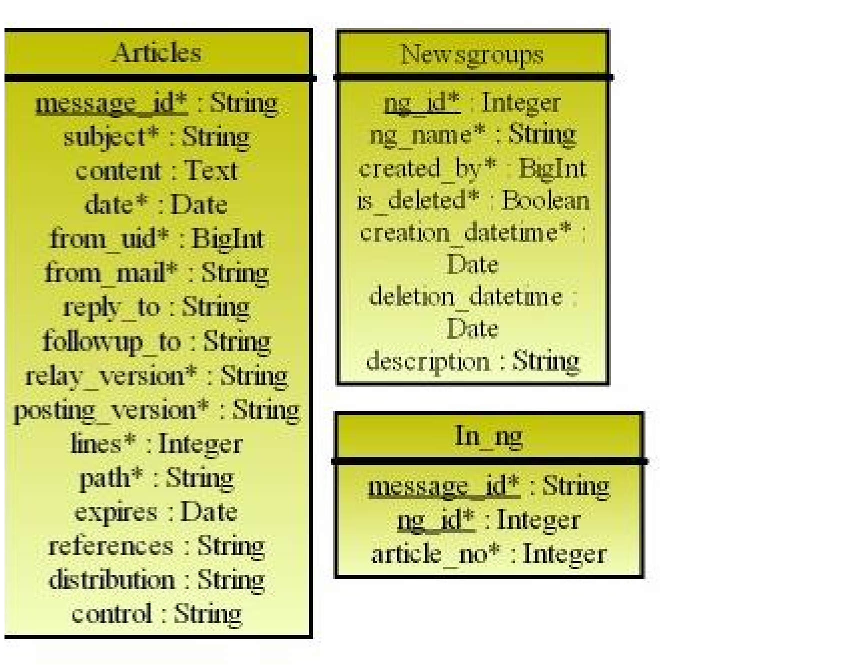

Articles
(1,1)
Posted_by
(0,n)
Users


Newsgroups
(1,1)
Created_by
(0,n)
Users


Newsgroups
(1,n)
In_ng
(0,n)
Articles

Entity Sets


Articles
message_id* : String
article_no : BigInt
subject* : String
content : Text
date* : Date
from_uid* : BigInt
from_mail* : String
reply_to : String
followup_to : String
relay_version* : String
posting_version* : String
lines* : Integer
path* : String
expires : Date
references : String
distribution : String
control : String
Users
user_id* : BigInt
password* : String
name* : String
surname* : String
username* : String
date_of_birth : Date
birth_place : String
phone* : String
e-mail* : String
signup_date* : date
lastLoginDate* : date
lastLoginIP* : String
removed_date : Date
group_id* : Integer
picture : BLOB
secretQuestion : String
questionAns : String
User_groups
group_id* : Integer
group_name* : String
access_level* : Integer
Newsgroups
ng_id* : Integer
ng_name* : String
created_by* : BigInt
creation_datetime* : Date
description : String
Subscription
user_id* : BigInt
ng_id* : Integer
wants_mail* : Boolean
Ng_mails
mail_address* : String
NgAccessLevel
ng_id* : Integer
access_level* : Integer
Ng_articles
article_no* : BigInt
message_id* : String
Login_log
user_id* : BigInt
login_date* : Date
login_IP* : String
Action_types
action_no : Integer
id_type : Integer
action_name : String
Configuration_log
user_id* : BigInt
date_time : Date
action_type : Integer
id* : String



4.1.2 Data Descriptions

The data description function is to deal with the structure of the data. We have taken each entity and relation separately and given each attribute in each entity or relation a type so the data is fully structured.

<u>Note:</u>

- **Data with underlines are primary keys;**
- **Data with star have to be entered absolutely (NOT NULL);**

Data Descriptions for Main Database

<u>Articles</u>

Data	**Type & Size**	**Format**
<u>article_no*</u>	BIGSERIAL	Number (AUTOINC)
message_id*	VARCHAR – 40	Text (UNIQUE)

subject*	VARCHAR – 60	Text
content	TEXT	Text
date*	DATETIME	Date/time
from_uid*	BIGINT	Number
from_mail*	VARCHAR – 40	Text
reply_to	VARCHAR – 40	Text
followup_to	VARCHAR – 40	Text
relay_version*	VARCHAR – 60	Text
posting_version*	VARCHAR – 60	Text
lines*	INTEGER	Number
path*	VARCHAR – 60	Text
expires	DATETIME	Date/time
references	VARCHAR – 60	Text
distribution	VARCHAR – 60	Text
control	VARCHAR – 60	Text

Users

Data	**Type & Size**	**Format**
user_id*	BIGSERIAL	Number (AUTOINC)
password*	VARCHAR – 20	Text is hidden. ********
name*	VARCHAR – 40	Text
username*	VARCHAR – 40	Text
date_of_birth	DATE	Date
birth_place	VARCHAR – 20	Text
phone*	VARCHAR – 40	Text
e-mail*	VARCHAR – 40	Text
signup_date*	DATETIME	Date/time
removed_date	DATETIME	Date/time
group_id*	INTEGER	Number
picture	BLOB	Binary

User_groups

Data	**Type & Size**	**Format**

group_id*	INTEGER	Number
group_name*	VARCHAR – 60	Text
access_level*	INTEGER	Number

Newsgroups

Data	**Type & Size**	**Format**
ng_id*	INTEGER	Number
ng_name*	VARCHAR – 60	Text
created_by*	BIGINT	Number
creation_datetime*	DATETIME	Date/time
description	VARCHAR – 60	Text

Ng_articles

Data	**Type & Size**	**Format**
article_no*	BIGINT	Number
message_id*	VARCHAR – 40	Text

Ng_mails

Data	**Type & Size**	**Format**
mail_address*	VARCHAR – 40	Text

Ng_access_levels

Data	**Type & Size**	**Format**
ng_id*	BIGINT	Number
access_level*	INT	Number

Subscription

Data	**Type & Size**	**Format**
user_id*	BIGINT	Number
ng_id*	INTEGER	Number

wants_mail*	BOOL	Yes/no

Login_Log

Data	Type & Size	Format
user_id*	BIGINT	Number
login_datetime*	DATETIME	Date/time
login_IP*	INET	IP Specific Text

Action_Types

Data	Type & Size	Format
action_no*	INT	Number (AUTOINC)
id_type*	TINYINT	Number
action_name*	VARCHAR - 100	Text

Configuration_Log

Data	Type & Size	Format
user_id*	BIGINT	Number
log_datetime*	DATETIME	Date/time
action_no*	INT	Number
id*	BIGINT	Number

Data Descriptions for Archive Database

Articles

Data	Type & Size	Format
article_no*	BIGSERIAL	Number (AUTOINC)
message_id*	VARCHAR – 40	Text (UNIQUE)
subject*	VARCHAR – 60	Text
content	TEXT	Text
date*	DATETIME	Date/time
from_uid*	BIGINT	Number

from_mail*	VARCHAR – 40	Text
reply_to	VARCHAR – 40	Text
followup_to	VARCHAR – 40	Text
relay_version*	VARCHAR – 60	Text
posting_version*	VARCHAR – 60	Text
lines*	INTEGER	Number
path*	VARCHAR – 60	Text
expires	DATETIME	Date/time
references	VARCHAR – 60	Text
distribution	VARCHAR – 60	Text

Newsgroups

Data	**Type & Size**	**Format**
ng_id*	INTEGER	Number
ng_name*	VARCHAR – 60	Text
created_by*	BIGINT	Number
is_deleted*	BOOLEAN	Yes/no
creation_datetime*	DATETIME	Date/time
deletion_datetime	DATETIME	Date/time
description	VARCHAR – 60	Text

In_ng

Data	**Type & Size**	**Format**
message_id*	VARCHAR – 40	Text
ng_id*	INTEGER	Number
article_no*	BIGINT	Number

4.1.3 Entity Descriptions

Entity & Relation Descriptions for Main Database

Articles

This entity contains all necessary information about articles which are posted to the news server. No matter to which group it is posted, all articles are stored in this table with all

required information. Some attributes are used for holding standard data for USENET messages and some attributes are assigned by us locally for managing articles easily.
In USENET message format, [6] there are some required headers and some optional headers. We hold these required headers and some of the optional headers in our database, in *Articles* entity, in order to obey universal USENET message standards. Below, the table's attributes are explained.

***article_no**:** This number is the identifier of the articles inside NewsAgent. System assigns a unique number each article arrives to the server. Although message-id of the article uniquely identifies each article, system assigns an integer valued identifier to each article to manage them more easily.
message_id*: Required `Message-ID` standard header is held in string *message_id*. This attribute uniquely defines a message. The same message ID may cannot be assigned to another article because this id is created by the clients according to their systems and merging this data with some information of the server.
subject*: Required `Subject` standard header is held in string *subject*. It is assigned by sender and briefly defines what the article is about.
content: This field is held in text format and stores the content of the article.
date*: Required `Date` standard header is held in *date* in date/time format. It is the time that the article is posted to the network.
from_uid*: This is a local assignment that is required to know which user has posted the article. It is a foreign key for this entity referencing *user_id* of *Users* entity.
from_mail*: Required `From` standard header is held in string *from_mail*. It is the mail address of the sender of that article. This is a default mail address and foreign key which references the attribute *e-mail* of *Users* entity.
reply_to: Optional `Reply-To` standard header is held in string *reply_to*. This string holds the optional mail address of the sender if he/she wants to get mail for that article to the specified address instead of *from_mail*.
followup_to: Optional `Followup-To` standard header is held in string *followup_to*. If this is not empty, all follow-ups to the article will be posted to the newsgroups specified

in this field. If it is empty, follow-ups will be posted to the newsgroup(s) that the message was originally posted.

relay_version*: Required \`Relay-Version\` standard header is held in string *relay_version*. This header shows the version of the program that is responsible for the transmission of the article.

posting_version*: Required \`Posting-Version\` standard header is held in string *posting_version*. This header identifies the software that is responsible for passing this message into the network.

lines*: This header is also required and specifies how many lines the article has. It is held in integer format.

path*: Path is a required header and shows the way that the article followed until reaching the system. Path is held in string format and when a system forwards this article, it concatenates its name to the path.

expires: This field is in date/time format and optional. If it exists, the article expires in specified date and time.

references: This field is optional and held in string format consisting of article ID\`s which prompt the submission of this article. For instance, in a follow-up article, the parent article exists in this field.

distribution: This field is held in string format and lists the newsgroups that the article should be sent. This field alters the original newsgroup distribution.

<u>Users</u>

This entity contains all required information about the users which can be authorized or unauthorized. Administrators are also users.

<u>user_id*:</u> This number specifies each user uniquely; hence *user_id* is the primary key of the *Users* entity.

name*: This string field holds the name of the user.

username*: This string field holds the username of the user.

password*: This string field is the matched password for the username of the user .

date_of_birth: This date typed attribute holds the birth date of the user.

birth_place: This string typed attribute holds the birth place of the user.
phone*: This string field holds the cell phone number of the customer.
e-mail*: This text field holds the mail address of the customer.
signup_date*: This field holds the date and time that the user has signed up. This field is of type date/time.
removed_date: This field is usually empty but if a user is removed from the database, this field holds the date and time that the user is removed from the system.
group_id*: Group id specifies which user group the user belongs to. This is a foreign key referencing *group_id* attribute of *User_groups* entity.
picture: Users can upload their pictures to the system. This picture is held in *picture* field in BLOB format.

User_groups

This entity holds information about user groups. Users are assigned to user groups according to their access rights.
group_id*: This number specifies each user group uniquely; hence *group_id* is the primary key of the *User_groups* entity.
group_name*: This string field holds the name of the usergroup.
access_level*: This integer field holds the access level of the user. For instance, if it is 1, it means full access.

Newsgroups

This entity holds information about newsgroups. When a newsgroup is added, listed information about that group is added to the table.
ng_id*: This number specifies each newsgroup uniquely; hence n*g_id* is the primary key of the *Newsgroups* entity.
ng_name*: This string field holds the name of the newsgroup.
created_by*: This big integer typed field holds information about who created this newsgroup. This is a foreign key of this entity referencing *user_id* attribute of *Users* entity.

creation_datetime*: This field holds the date and time that the newsgroup is created. This field is of type date/time.
description: This string field holds a brief description about what the newsgroup is about.

<u>Ng_articles</u>

Ng_articles is a general name for lots of possible tables. When a new newsgroup is created, an article table is created for that newsgroup with a specifying name. For example, if a group named `Music` is created, a table named `Music_articles` is also created. This table does not hold all information about the articles belonging to that table. It only holds little information about articles posted to that group for referencing the articles from main *Articles* table. This way is chosen in order to prevent the database from multiple storage of same article when it is posted to different groups at the same time.
<u>article_no*</u>: This number specifies each article in the server uniquely; hence article_no is the primary key of the *Ng_articles* entity. This is a foreign key referencing to the Articles table.
message_id*: This field is also held with article number because news readers may want request any article by means of the universal message-ids.

<u>Ng_mails</u>

Ng_mails is also a general name for lots of possible tables. When a new newsgroup is created, a mails table is created for that newsgroup with a specifying name. For example, if a group named `Cinema` is created, a table named `Cinema_mails` is also created. This entity is formed in order to store mail addresses of people who subscribed to receive the articles that are posted to the specified newsgroup as e-mail.
mail_address*: This string field holds the mail addresses of the users who want to receive e-mails from the specified newsgroup.

<u>Ng_access_levels</u>

Data	**Type & Size**	**Format**

ng_id*	BIGINT	Number
access_level*	INT	Number

This table specifies access levels of each newsgroup to determine the user groups which will be able to access to which newsgroup in the news server.

ng_id*: This field is the id specifies the newsgroups uniquely. This is a foreign key for this relation referencing *ng_id* attribute of *Newsgroups* entity.

access_level*: This attribute stores an integer which specifies the access level of newsgroups.

- ng_id, itself, the primary key of this table, since each newsgroup will be stored once in this table.

Subscription

This table specifies a relation among users and newsgroups. Users can be subscribed to newsgroups. Required information about this subscription is held in this table.
user_id*: This field is the id of the user who subscribed to the newsgroup. This is a foreign key for this relation referencing *user_id* attribute of *Users* entity. This field is a subset of primary key.
ng_id*: This field is the id of the newsgroup which is subscribed by the user. This is a foreign key for this relation referencing *ng_id* attribute of *Newsgroups* entity. This field is also a subset of primary key.

- ng_id and user_id are primary key of the relation together.

wants_mail*: This Boolean type is hold to know whether the user wants e-mail from this newsgroup or not.

Login_Log

This table stores information about each log in of users. When for each log in to the system, a row is inserted to this table which includes user_id of user, date and time of the login and ip of the computer that user login to the system. Storing this information is significant for a news server, like NewsAgent, since security is a key point. Also, specifying the computer that user logged in to the system in his previous login is a smart feature.

user_id*: This number specifies each user uniquely. This is a foreign key referencing to the Users table.
login_datetime*: This timestamp attribute stores the date and time of the login.
login_IP*: This attribute stores the ip address of the computer that user logged in to the system.

- User_id and login_datetime together forms the primary key of this table, since we consider that any user can login to the system once at any specified time.

Action_Types

This table, in fact, is stored for specifying the configuration actions of users which are stored in Config_log table. In fact, this table is mostly a static table, since there will be no major change on this table when all action types have already been specified. Only a small number of insertions, deletions and updates may be applied on this table when an action type will be inserted, deleted or updated, respectively.

action_no*: This number specifies each action type uniquely.
id_type*: This attribute specifies one of article_no, user_id, ng_id. This id is the specification for on which type of data, the configuration can be done.
action_name*: action_name is just an attribute to specify the name of the action_type. For instance, update of article may be a possible name for an action_name.

- Action_no is the primary key of this table. It will be auto incremented when a action is inserted to this table.

Configuration_Log

This table stores all configurations of users on database. When an insertion, deletion or update is done, a row is inserted to the configuration_log table. Like login_log table, this information is significant for security reasons. Storing configuration actions data in the database provide us to control the configurations done on database by each user and when this configuration is done.

user_id*: This attribute specifies the user who does the configuration. This user_id is a foreign key to the Users table.

log_datetime*: This timestamp attribute stores the date and time of the configuration.

action_no*: This integer stores the information of which configuration is done by the user specified by user_id attribute. Since action types table stores all actions can be applied by users, this attribute is a foreign key to action_types table.

id*: id attribute stores the id of the message, newsgroup or user on which configuration is done. Since action_no table is storing whether the configuration is applied on a message, a newsgroup or a user, it is easy to determine the id is related with whether a message, a newsgroup or a user. By using this id and other attributes of this table, a config_log tuple can easily be created.

- User_id and login_datetime together forms the primary key of this table, since we consider that any user can make a configuration on the system once at any specified time.

Entity Descriptions for Archive Database

Articles

This entity is the same as *Articles* entity in main database. Definitions of attributes are as listed there.

<u>Newsgroups</u>

This entity is the same as *Articles* entity in main database except for the *is_deleted and deletion_datetime* attributes of this newsgroups entity. *is_deleted* boolean attribute specifies whether that newsgroup is deleted or not, since a deleted newsgroup can exist in archive database but not main database. *deletion_datetime* attribute specifies the deletion time of the newsgroup if it is deleted. Definitions of other attributes are as listed in definition of main database entity.

<u>In_ng</u>

This table specifies a relation among articles and newsgroups in archive database. Articles belong to newsgroups. We needed this relation only for this database, since in archive database; we do not hold different tables for different newsgroups that list the articles posted to that newsgroup.

<u>article_no*</u>: This number specifies each article in the server uniquely; hence article_no is the primary key of the *Ng_articles* entity. This is a foreign key referencing to the Articles table.

message_id*: This field is also held with article number because news readers may want request any article by means of the universal message-ids.

<u>ng_id*:</u> This field is a foreign key for this relation referencing *ng_id* of *Newsgroups* entity. It defines which newsgroup the message belongs to.

- ng_id and message_id are primary key of the relation together.

4.1.4 Creating NewsAgent Database

createDB.sql

```
\i Users.sql
\i Articles.sql
\i Newsgroups.sql
\i User_groups.sql
\i Ng_articles.sql
\i Ng_mails.sql
\i Ng_access_levels.sql
```

```
\i Subscription.sql
\i Action_types.sql
\i Login_log.sql
\i Config_log.sql
\i Archive_articles.sql
\i Archive_newsgroups.sql
\i Archive_in_ng.sql
```

deleteDB.sql

```
DROP TABLE Archive_in_ng;
DROP TABLE Archive_newsgroups;
DROP TABLE Archive_articles;
DROP TABLE Config_log;
DROP TABLE Login_log;
DROP TABLE Action_types;
DROP TABLE Subscription;
DROP TABLE Ng_access_levels;
DROP TABLE Ng_mails;
DROP TABLE Ng_articles;
DROP TABLE User_groups;
DROP TABLE Newsgroups;
DROP TABLE Articles;
DROP TABLE Users;
```

Login_log.sql

```
CREATE TABLE Login_log (

        user_id         BIGINT,
        login_datetime          TIMESTAMP,
        login_ip        INET,

        PRIMARY KEY (user_id, login_datetime),
        FOREIGN KEY (user_id) REFERENCES Users (user_id)
);
```

Action_types.sql

```
CREATE TABLE Action_types (

        action_no       SERIAL,
        id_type                 SMALLINT,
        action_name     TEXT,

        PRIMARY KEY (action_no)
);
```

Configuration_log.sql

```
CREATE TABLE Configuration_log (

        user_id         BIGINT,
        log_datetime    TIMESTAMP,
        action_no       INTEGER,
        id              VARCHAR(40),

        PRIMARY KEY (user_id, log_datetime),
        FOREIGN KEY (user_id) REFERENCES Users (user_id),
        FOREIGN KEY (action_no) REFERENCES Action_types (action_no)
);
```

Ng_access_levels.sql

```
CREATE TABLE Ng_access_levels (

        ng_id           BIGINT,
        access_level    INTEGER NOT NULL,

        PRIMARY KEY (ng_id, access_level),
        FOREIGN KEY (ng_id) REFERENCES Newsgroups (ng_id)
);
```

Users.sql

```
CREATE TABLE Users (

    user_id                 BIGSERIAL,
    username                VARCHAR(40) NOT NULL,
    password                VARCHAR(20) NOT NULL,
    name                    VARCHAR(40) NOT NULL,
    surname                 VARCHAR(40) NOT NULL,
    date_of_birth           DATE,
    birth_place             VARCHAR(20),
    phone                   VARCHAR(40) NOT NULL,
    e_mail                  VARCHAR(40) NOT NULL,
    signup_date             TIMESTAMP NOT NULL,
    removed_date            TIMESTAMP,
    group_id                INTEGER NOT NULL,
    picture                 BYTEA,
    last_login_date_time    TIMESTAMP,
    last_login_IP             INET,
    secret_question           TEXT,
    secret_question_answer    TEXT,

    UNIQUE (e_mail),
    UNIQUE (username),
    PRIMARY KEY(user_id)
);
```

Archive_newsgroup.sql

```
CREATE TABLE Archive_newsgroups (

    ng_id              BIGSERIAL,
    ng_name            VARCHAR(60) NOT NULL,
    created_by         BIGINT NOT NULL,
    creation_datetime  TIMESTAMP NOT NULL,
    description VARCHAR(60),

    PRIMARY KEY (ng_id)
);
```

Archive_in_ng.sql

```
CREATE TABLE Archive_in_ng (

        message_id    VARCHAR(40),
        ng_id         BIGINT,
        article_no    BIGINT NOT NULL,

        PRIMARY KEY (message_id, ng_id)
);
```

Archive_articles.sql

```
CREATE TABLE Archive_articles (

   message_id        VARCHAR(40),
   subject           VARCHAR(60) NOT NULL,
   content           TEXT,
   dateTime          TIMESTAMP NOT NULL,
   from_uid          BIGINT NOT NULL,
   from_mail         VARCHAR(40) NOT NULL,
   reply_to          VARCHAR(40),
   followup_to       VARCHAR(40),
   relay_version     VARCHAR(60) NOT NULL,
   posting_version     VARCHAR(60) NOT NULL,
   lines          INTEGER NOT NULL,
   path                VARCHAR(60) NOT NULL,
   expires             TIMESTAMP,
   reference           VARCHAR(60),
   distribution VARCHAR(60),
   control             VARCHAR(60),

   PRIMARY KEY (message_id),
   FOREIGN KEY (from_uid) REFERENCES Users (user_id)
);
```

User_groups.sql

```
CREATE TABLE User_groups (

    group_id            BIGSERIAL,
    group_name                    VARCHAR(60) NOT NULL,
    access_level        INTEGER NOT NULL,

    PRIMARY KEY (group_id)
);
```

Subscription.sql

```
CREATE TABLE Subscription (

        user_id         BIGINT,
        ng_id           BIGINT,
        wants_mail      BOOLEAN NOT NULL,

        PRIMARY KEY (user_id, ng_id),
        FOREIGN KEY (user_id) REFERENCES Users (user_id),
        FOREIGN KEY (ng_id) REFERENCES Newsgroups (ng_id)
);
```

Ng_mails.sql

```
CREATE TABLE Ng_mails (

    mail_address        VARCHAR(40),

    PRIMARY KEY (mail_address),
    FOREIGN KEY (mail_address) REFERENCES Users (e_mail)
);
```

Ng_articles.sql

```
CREATE TABLE Ng_articles (

    article_no             BIGSERIAL,
    message_id                        VARCHAR(40) NOT NULL,

    UNIQUE (message_id),
    PRIMARY KEY (article_no),
    FOREIGN KEY (message_id) REFERENCES  Articles (message_id)
);
```

Articles.sql

```
CREATE TABLE Articles (

   message_id          VARCHAR(40),
   subject             VARCHAR(60) NOT NULL,
   content             TEXT,
   dateTime            TIMESTAMP NOT NULL,
   from_uid            BIGINT NOT NULL,
   from_mail           VARCHAR(40) NOT NULL,
   reply_to            VARCHAR(40),
   followup_to         VARCHAR(40),
   relay_version       VARCHAR(60) NOT NULL,
   posting_version     VARCHAR(60) NOT NULL,
   lines        INTEGER NOT NULL,
   path                VARCHAR(60) NOT NULL,
   expires             TIMESTAMP,
   reference           VARCHAR(60),
   distribution VARCHAR(60),
   control             VARCHAR(60),

   PRIMARY KEY (message_id),
   FOREIGN KEY (from_uid) REFERENCES Users (user_id)
);
```

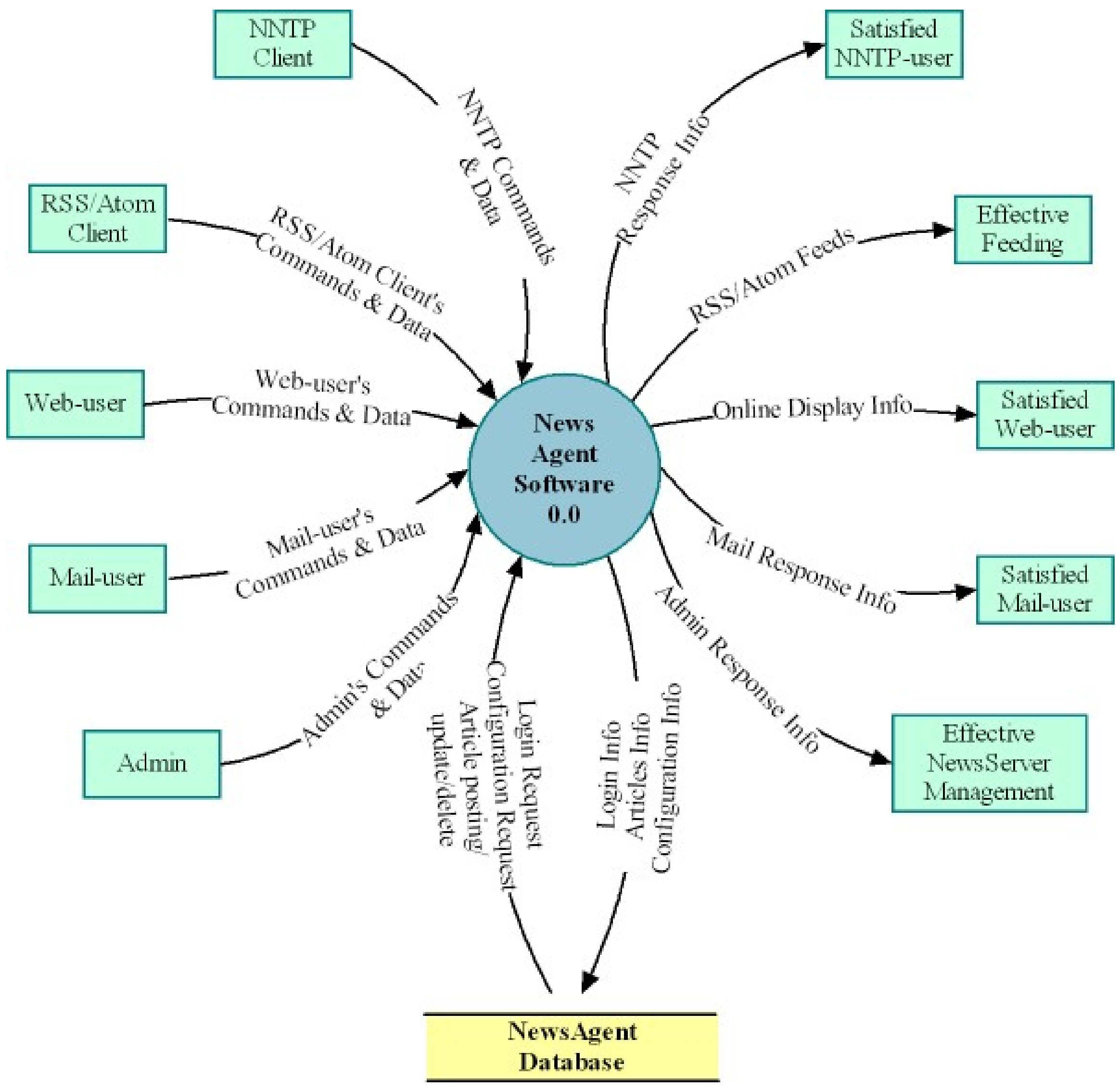
Newsgroups.sql

```
CREATE TABLE Newsgroups (

    ng_id              BIGSERIAL,
    ng_name            VARCHAR(60) NOT NULL,
    created_by         BIGINT NOT NULL,
    creation_datetime  TIMESTAMP NOT NULL,
    description VARCHAR(60),

    PRIMARY KEY (ng_id),
    FOREIGN KEY (created_by) REFERENCES Users (user_id)
);
```

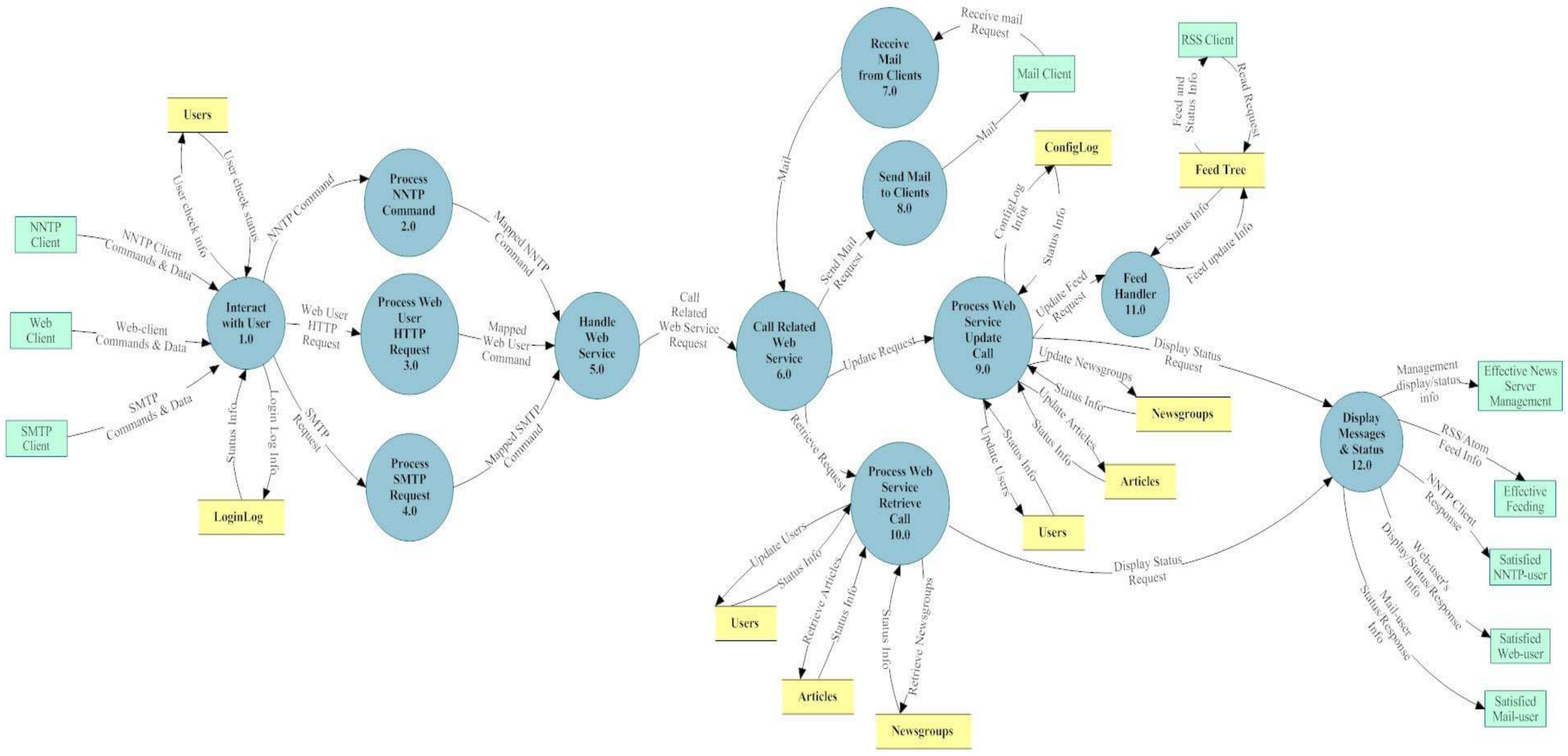
4.2 FUNCTIONAL MODELLING

4.2.1 Data Flow Diagrams

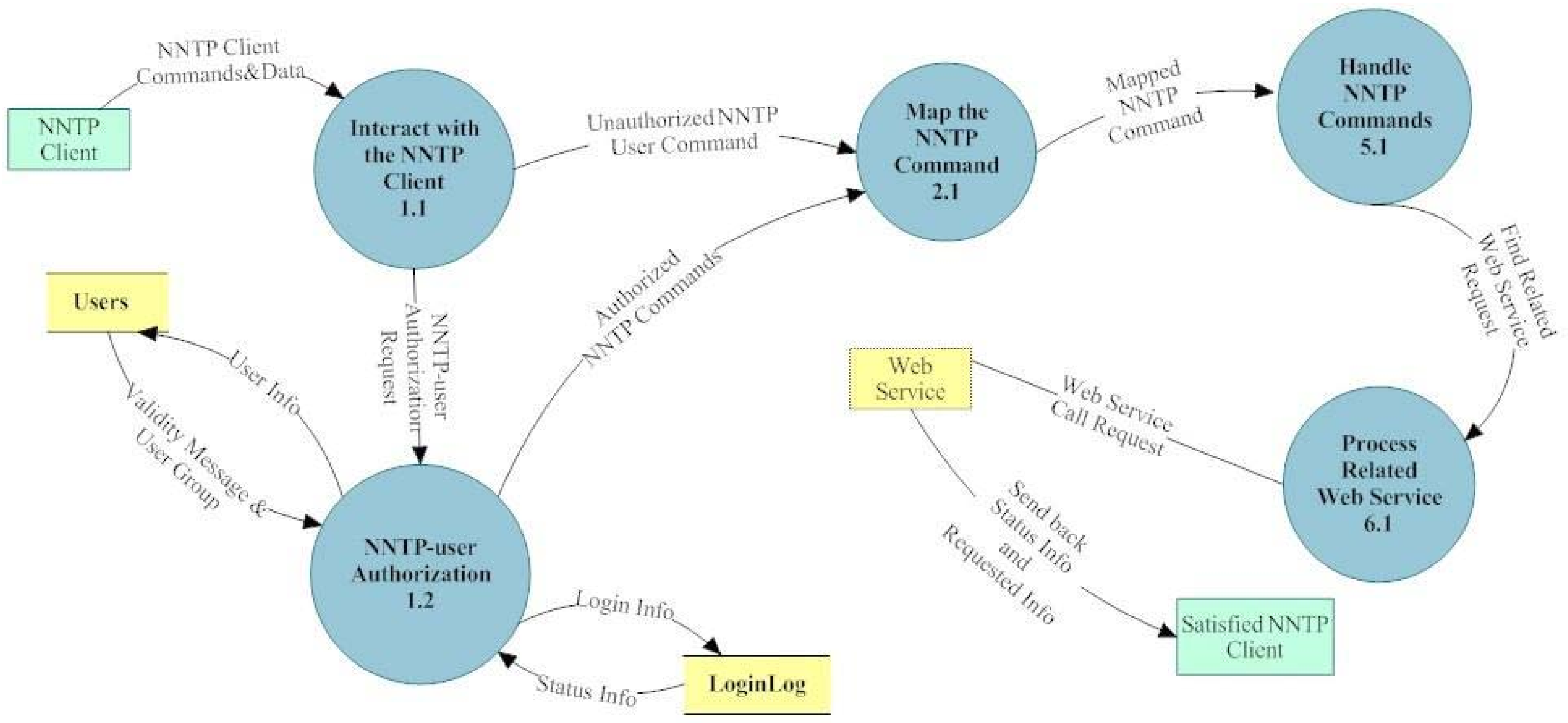
4.2.1.1 LEVEL 0 DATA FLOW DIAGRAM

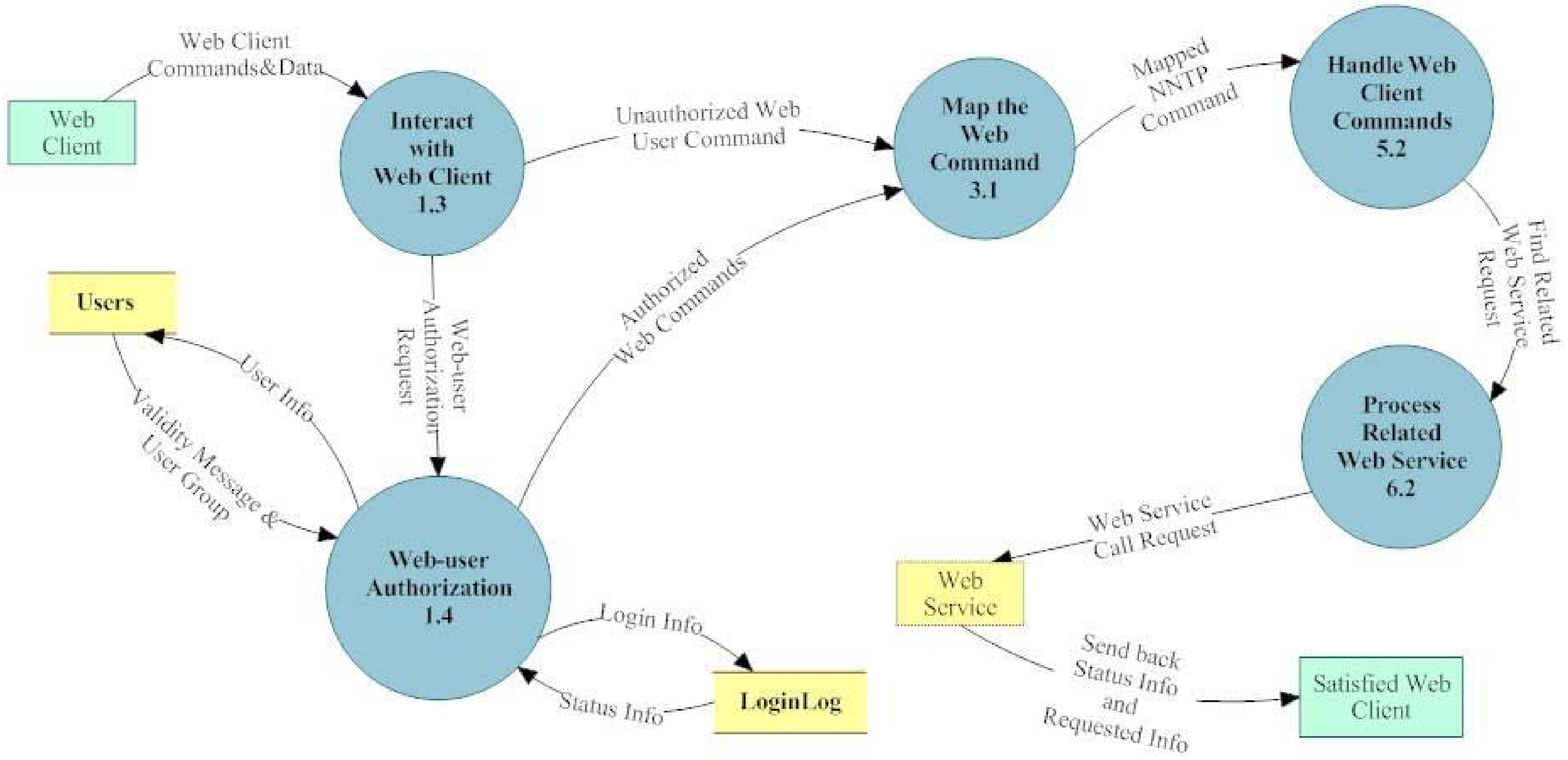


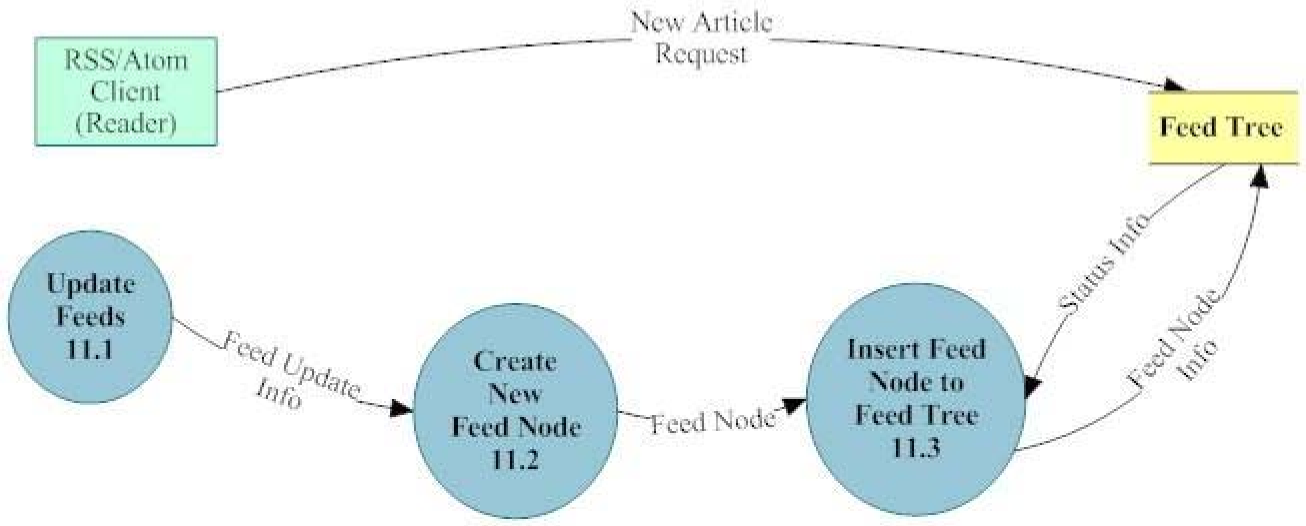
4.2.1.2 LEVEL 1 DATA FLOW DIAGRAM

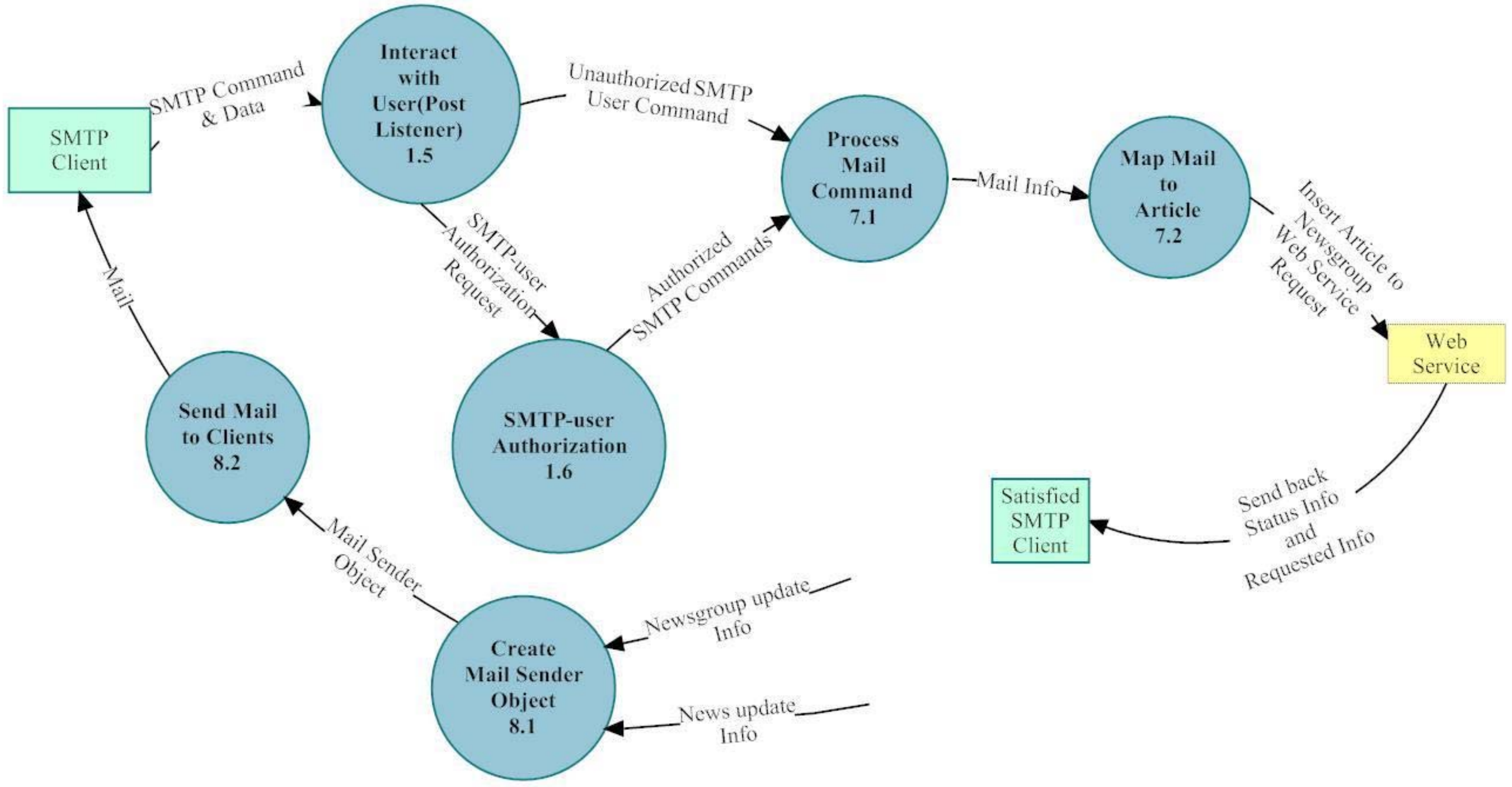


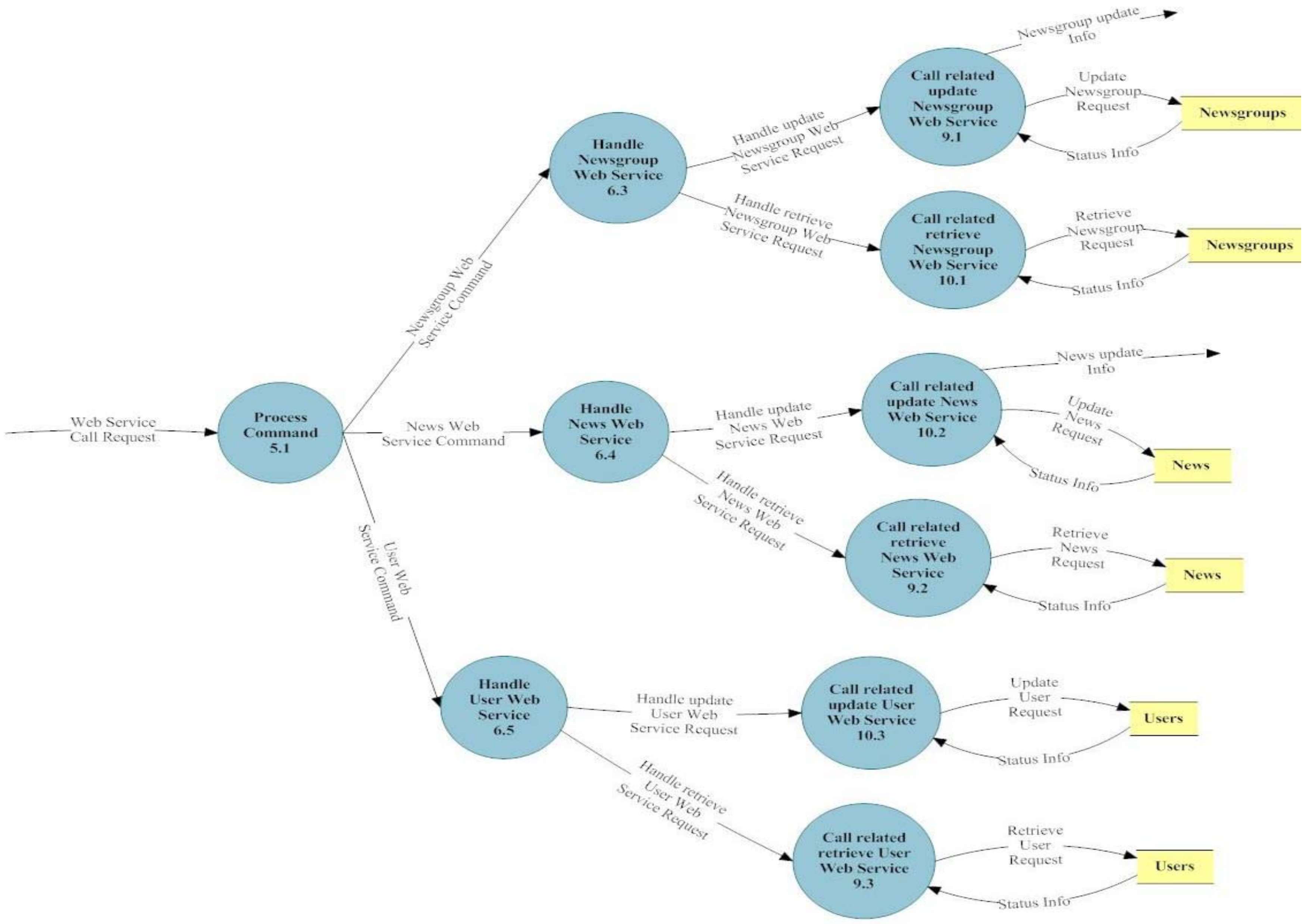
4.2.1.3 LEVEL 2 DATA FLOW DIAGRAMS


NNTP Client
NNTP Client Commands&Data
Interact with the NNTP Client 1.1
Unauthorized NNTP User Command
Map the NNTP Command 2.1
Mapped NNTP Command
Handle NNTP Commands 5.1
Find Related Web Service Request
Users
User Info
Validity Message & User Group
NNTP-user Authorization Request
Authorized NNTP Commands
NNTP-user Authorization 1.2
Login Info
Status Info
LoginLog
Web Service
Web Service Call Request
Process Related Web Service 6.1
Send back Status Info and Requested Info
Satisfied NNTP Client


Web Client
Web Client Commands&Data
Interact with Web Client 1.3
Unauthorized Web User Command
Map the Web Command 3.1
Mapped NNTP Command
Handle Web Client Commands 5.2
Find Related Web Service Request
Process Related Web Service 6.2
Web Service Call Request
Web Service
Send back Status Info and Requested Info
Satisfied Web Client
Users
User Info
Validity Message & User Group
Web-user Authorization Request
Authorized Web Commands
Web-user Authorization 1.4
Login Info
Status Info
LoginLog


RSS/Atom Client (Reader)
New Article Request
Feed Tree
Status Info
Feed Node Info
Update Feeds 11.1
Feed Update Info
Create New Feed Node 11.2
Feed Node
Insert Feed Node to Feed Tree 11.3


SMTP Client
SMTP Command & Data
Interact with User(Post Listener) 1.5
Unauthorized SMTP User Command
Process Mail Command 7.1
Mail Info
Map Mail to Article 7.2
Insert Article to Newsgroup Web Service Request
Web Service
SMTP-user Authorization Request
Authorized SMTP Commands
SMTP-user Authorization 1.6
Mail
Send Mail to Clients 8.2
Send back Status Info and Requested Info
Satisfied SMTP Client
Mail Sender Object
Create Mail Sender Object 8.1
Newsgroup update Info
News update Info


Web Service Call Request
Process Command 5.1
Newsgroup Web Service Command
News Web Service Command
User Web Service Command
Handle Newsgroup Web Service 6.3
Handle News Web Service 6.4
Handle User Web Service 6.5
Handle update Newsgroup Web Service Request
Handle retrieve Newsgroup Web Service Request
Handle update News Web Service Request
Handle retrieve News Web Service Request
Handle update User Web Service Request
Handle retrieve User Web Service Request
Call related update Newsgroup Web Service 9.1
Call related retrieve Newsgroup Web Service 10.1
Call related update News Web Service 10.2
Call related retrieve News Web Service 9.2
Call related update User Web Service 10.3
Call related retrieve User Web Service 9.3
Newsgroup update Info
Update Newsgroup Request
Status Info
Newsgroups
Retrieve Newsgroup Request
Status Info
Newsgroups
News update Info
Update News Request
Status Info
News
Retrieve News Request
Status Info
News
Update User Request
Status Info
Users
Retrieve User Request
Status Info
Users

4.2.2 Data Dictionary

Name:	NNTP Client Commands&Data
Aliases:	NNTP Requests
Where used/how used:	NNTP Client (Output)
	Interact with the NNTP Client 1.1 (Input)

Description:

NNTP Client sends requests as in format stated in RFC-977. It also sends the required article information like server specific article number or universal message id.

Name:	NNTP User Authorization Request
Aliases:	NNTP Authentication
Where used/how used:	Interact with the NNTP Client 1.1 (Output)
	NNTP User Authorization 1.2 (Input)

Description:

If the user wants to access to a field which is not accessible by unauthorized users, system wants the user to send his/her crypted username and password information. Afterwards client sends the authentication request to the system.

Name:	User Info
Aliases:	Username & Password
Where used/how used:	NNTP User Authorization 1.2 (Output)
	Users (Database) (Input)

Description:

To authenticate the user who applied through authentication request, user's username and hashed password is sent to the database. The passwords' encrypted forms are matched to send back validity information.

Name:	Validity Message & User Group
Aliases:	None
Where used/how used:	Users (Database) (Output)
	NNTP User Authorization 1.2 (Input)

Description:

If the password which the user entered matches with the one in the system database, a signal indicating that "the user can go ahead" and his/her user group is returned.

Name:	Login Info
Aliases:	None
Where used/how used:	NNTP User Authorization 1.2 (Output)
	LoginLog (Database) (Input)

Description:

To assure security criteria of NewsAgent, every login action is logged in the system. User's identifier, login date and time, the machine which the user connected to the system and a descriptive text is stored into the database.

Name:	Status Info
Aliases:	None
Where used/how used:	LoginLog (Database) (Output)
	NNTP User Authorization 1.2 (Input)

Description:

This data is the result for acknowledgement indicating that the log information is successfully inserted into the database.

Name:	Authorized NNTP Commands
Aliases:	Authenticated NNTP Requests
Where used/how used:	NNTP User Authorization 1.2 (Output)
	Map the NNTP Command 2.1 (Input)

Description:

Authenticated NNTP Commands include all post, read, update etc. The commands that an authenticated user may send.

Name:	Unauthorized NNTP User Commands
Aliases:	Unauthenticated NNTP Commands
Where used/how used:	Interact with the NNTP Client 1.1 (Output)
	Map the NNTP Command 2.1 (Input)

Description:

NewsAgent will be flexible to allow to edit the security preferences. If it is wanted, users may be allowed to access the specified resources, articles from the database through the web services.

Name:	Mapped NNTP Command
Aliases:	None
Where used/how used:	Map the NNTP Command 2.1 (Output)
	Handle NNTP Commands 5.1 (Input)

Description:

The NNTP commands taken through the port are parsed and mapped to the convenient functions of the system. This data is the corresponding function calls of NNTP standard commands.

Name:	Find Related Web Service Request
Aliases:	Look-up for Web Service
Where used/how used:	Handle NNTP Commands 5.1 (Output)
	Process Related Web Service 6.1 (Input)

Description:

This information is used to find the related web service. Actually, this link is used to obey the conventions. UDDI is not used in NewsAgent because we already know which web service does what and their endpoints.

Name:	Web Service Call Request
Aliases:	Invoking the Corresponding Web Service Data
Where used/how used:	Process Related Web Service 6.1 (Output)
	Web Service (Input)

Description:

This data is the SOAP message which is required to invoke web services and carry information between the services and the invokers. The parameters, returning values including primitive types and built-in simple types are carried through SOAP messages.

Name:	Send Back Status Info and Requested Info
Aliases:	None
Where used/how used:	Web Service (Output)
	Satisfied NNTP Client (Input)

Description:

This is the data returned from the invoked web services. This is also a SOAP message as explained above.

Name:	Web Client Commands & Data
Aliases:	Web Client's Requests
Where used/how used:	Web Client (Output)
	Interact with Web Client 1.3 (Input)

Description:

Web Client sends his/her requests to the system through NewsAgent web module.

Name:	Web User Authorization Request
Aliases:	Web User Authentication
Where used/how used:	Interact with the Web Client 1.3 (Output)
	Web User Authorization 1.4 (Input)

Description:

If the user wants to access to a field which is not accessible by unauthorized users, system wants the user to send his/her crypted username and password information. Afterwards client sends the authentication request to the system.

Name:	User Info
Aliases:	Username & Password
Where used/how used:	Web User Authorization 1.4 (Output)
	Users (Database) (Input)

Description:

To authenticate the user who applied through authentication request, user's username and hashed password is sent to the database. The passwords' encrypted forms are matched to send back validity information.

Name:	Validity Message & User Group
Aliases:	None
Where used/how used:	Users (Database) (Output)
	Web User Authorization 1.4 (Input)

Description:

If the password which the user entered matches with the one in the system database, a signal indicating that "the user can go ahead" and his/her user group is returned.

Name:	Login Info
Aliases:	None
Where used/how used:	Web User Authorization 1.4 (Output)
	LoginLog (Database) (Input)

Description:

To assure security criteria of NewsAgent, every login action is logged in the system. User's identifier, login date and time, the machine which the user connected to the system and a descriptive text is stored into the database.

Name:	Status Info
Aliases:	None
Where used/how used:	LoginLog (Database) (Output)
	Web User Authorization 1.4 (Input)

Description:

This data is the result for acknowledgement indicating that the log information is successfully inserted into the database.

Name:	Authorized Web Commands
Aliases:	Authenticated Web Requests
Where used/how used:	Web User Authorization 1.4 (Output)
	Map the Web Command 3.1 (Input)

Description:

Authenticated Web Commands include all post, read, update etc. The commands that an authenticated user may send.

Name:	Unauthorized Web User Commands
Aliases:	Unauthenticated Web Commands
Where used/how used:	Interact with the Web Client 1.3 (Output)
	Map the Web Command 3.1 (Input)

Description:

NewsAgent will be flexible to allow to edit the security preferences. If it is wanted, users may be allowed to access the specified resources, articles from the database through the web services.

Name:	Mapped Web Command
Aliases:	None
Where used/how used:	Map the Web Command 3.1 (Output)
	Handle Web Client Commands 5.2 (Input)

Description:

The Web commands taken through the port are parsed and mapped to the convenient functions of the system.

Name:	Find Related Web Service Request
Aliases:	Look-up for Web Service
Where used/how used:	Handle Web Commands 5.2 (Output)
	Process Related Web Service 6.2 (Input)

Description:

This information is used to find the related web service. Actually, this link is used to obey the conventions. UDDI is not used in NewsAgent because we already know which web service does what and their endpoints.

Name:	Web Service Call Request
Aliases:	Invoking the Corresponding Web Service Data
Where used/how used:	Process Related Web Service 6.1 (Output)
	Web Service (Input)

Description:

This data is the SOAP message which is required to invoke web services and carry information between the services and the invokers. The parameters, returning values including primitive types and built-in simple types are carried through SOAP messages.

Name:	Send Back Status Info and Requested Info
Aliases:	None
Where used/how used:	Web Service (Output)
	Satisfied Web Client (Input)

Description:

This is the data returned from the invoked web services. This is also a SOAP message as explained above.

Name:	New Article Request
Aliases:	None
Where used/how used:	RSS/ Atom Client – Reader, Aggregator (Output)
	Feed Tree (Input)

Description:

RSS/Atom readers needs the endpoint of the feed to subscribe. When they connect to the feed, they can subscribe them easily out of the responsibility of NewsAgent.

Name:	Feed Update Info
Aliases:	None
Where used/how used:	Update Feeds 11.1 (Output)
	Create New Feed Node 11.2 (Input)

Description:

When an article is posted to the system, after insertion to the database a feed entry is prepared automatically to add to the feed. This procedure is also followed when any deletion or update operation.

Name:	Feed Node
Aliases:	Feed Entry
Where used/how used:	Create New Feed Node 11.2 (Output)
	Insert Feed Node to Feed Tree 11.3 (Input)

Description:

This is the newly created or edited feed entry which will be added to the feed tree of the corresponding news group.

Name:	Feed Node Info
Aliases:	None
Where used/how used:	Insert Feed Node to Feed Tree 11.3 (Output)
	Feed Tree (Input)

Description:

After required operations are done on the created or edited Feed Node it is transferred to the tree and added to the tree as a new node.

Name:	Status Info
Aliases:	None
Where used/how used:	Feed Tree (Output)
	Insert Feed Node to Feed Tree 11.3 (Input)

Description:

The result of the add operation of the new node to the tree is returned to inform the system about the success or failure of node operation on the tree.

Name:	SMTP Command & Data
Aliases:	None
Where used/how used:	SMTP Client (Output)
	Interact with User 1.5 – Port Listener (Input)

Description:

Mail Client sends his/her requests to the system through NewsAgent mail module. Actually this an electronic mail which has the address of a newsgroup in the system.

Name:	SMTP-User Authorization Request
Aliases:	SMTP-User E-Mail Address
Where used/how used:	Interact with User – Port Listener 1.5 (Output)
	SMTP-User Authorization 1.6 (Input)

Description:

If the user attempts to send e-mail to a non-public newsgroup, his/her e-mail address is checked if it is already subscribed to that newsgroup's email subscription table. This data is the mail address of the user which is parsed out from the e-mail.

Name:	Authorized SMTP Commands
Aliases:	Authenticated SMTP Requests
Where used/how used:	SMTP-User Authorization 1.6 (Output)
	Process Main Command 7.1 (Input)

Description:

If the user is authorized to send mail to the specified newsgroup it is carried as an authenticated command.

Name:	Unauthorized SMTP Commands
Aliases:	Unauthenticated SMTP Requests
Where used/how used:	Interact with User – Port Listener 1.5 (Output)
	Process Main Command 7.1 (Input)

Description:

If the user is not authorized to send mail to the specified newsgroup it is carried as an unauthenticated command. And it is rejected.

Name:	Mail Info
Aliases:	Node
Where used/how used:	Process Main Command 7.1 (Output)
	Map Mail to Article 7.2 (Input)

Description:

If the mail is decided to be posted to the server, it should be converted to the convenient data type. This information is processed and mapped to an article data.

Name:	Insert Article to Newsgroup Web Service Request
Aliases:	Invoking the Corresponding Web Service Data
Where used/how used:	Map Mail to Article 7.2 (Output)
	Web Service (Input)

Description:

This data is the SOAP message which is required to invoke web services and carry information between the services and the invokers. The parameters, returning values including primitive types and built-in simple types are carried through SOAP messages.

Name:	Send Back Status Info and Requested Info
Aliases:	None
Where used/how used:	Web Service (Output)
	Satisfied SMTP Client (Input)

Description:

This is the data returned from the invoked web services. This is also a SOAP message as explained above.

Name:	Newsgroup Update Info
Aliases:	None
Where used/how used:	Call Related Update Newsgroup Web Service 9.1 (Output)
	Satisfied SMTP Client 8.1 (Input)

Description:

If any change occurs in the database related to the newsgroups this information is also transferred to the mail module to publish this event to the subscribers of the newsgroup. Or if a new newsgroup is created, this event is published to all users of the system to make them aware of the newly created newsgroup.

Name:	News Update Info
Aliases:	None
Where used/how used:	Call Related Update News Web Service 10.2 (Output)
	Satisfied SMTP Client 8.1 (Input)

Description:

If any change occurs in the database related to the articles this information is also transferred to the mail module to publish this event to the subscribers of the newsgroup which the article belongs to. Or if a new article is posted, it is mailed to the subscribers of the corresponding newsgroup.

Name:	Mail Sender Object
Aliases:	None
Where used/how used:	Satisfied SMTP Client 8.1 (Output)
	Send Mail to Clients 8.2 (Input)

Description:

This the mail object which is formed from the article object. This data will be directly converted to the electronic mail to be sent to the mail client.

Name:	Mail
Aliases:	None
Where used/how used:	Send Mail to Clients 8.2 (Output)

	SMTP Client (Input)

Description:

The electronic mail which is sent to the mail client.

Name:	Web Service Call Request
Aliases:	None
Where used/how used:	Map Commands to Web Service Commands (Output)
	Process Command 5.1 (Input)

Description:

The data in Web Service Call Request is a mapped command which specify the web service call that should be processed. All Web service calls are made through this data. Data specified in Web Service Call Request are in fact an interface for a database access.

Name:	Newsgroup Web Service Command
Aliases:	None
Where used/how used:	Process Commands 5.1 (Output)
	Handle Newsgroup Web Service 6.3 (Input)

Description:

Newsgroup Web Service Command specifies Newsgroups table will be accessed in the database. Newsgroup Web Service Handler will manage this data to determine the effect of it on the database, whether it is retrieval or update command.

Name:	Handle Update Newsgroup Web Service Request
Aliases:	None
Where used/how used:	Handle Newsgroup Web Service 6.3 (Output)
	Call Related Update Newsgroup Web Service 9.1 (Input)

Description:

This data is an update command web service for newsgroups. Since update on newsgroups or creation of a new newsgroup will cause updates on the database, namely on Newsgroups table, all update command on a newsgroup will flow through this data. We have considered the creation of a new newsgroup also as an update, since there will be a change on Newsgroups table.

.

Name:	Handle Retrieve Newsgroup Web Service Request
Aliases:	None
Where used/how used:	Handle Newsgroup Web Service 6.3 (Output)
	Call Related Retrieve Newsgroup Web Service 10.1 (Input)

Description:

This data is a retrieve command web service for newsgroups. Retrieval is any access to the database that does not cause any change on database. For this data, it is only retrievals from Newsgroups table in the database. This data should be processed so that which data about any newsgroup will be retrieved. This is done in Call Related Retrieve Newsgroup Web Service process.

Name:	News Web Service Command
Aliases:	None
Where used/how used:	Process Commands 5.1 (Output)
	Handle News Web Service 6.4 (Input)

Description:

News Web Service Command specifies Articles table will be accessed in the database. News Web Service Handler will manage this data to determine the effect of it on the database whether, it is retrieval or update command.

Name:	Handle Update News Web Service Request
Aliases:	None
Where used/how used:	Handle News Web Service 6.4 (Output)
	Call Related Update News Web Service 10.2 (Input)

Description:

This data is an update command web service for articles. Since update on an already posted article or posting a new article will cause updates on the database, namely on Articles table, all update command on Articles table will flow through this data. We have considered posting a new article is also as an update, since there will be a change on Articles table.

Name:	Handle Retrieve News Web Service Request
Aliases:	None
Where used/how used:	Handle News Web Service 6.4 (Output)
	Call Related Retrieve News Web Service 9.2 (Input)

Description:

This data is a retrieve command web service for articles. Retrieval is any access to the database that does not cause any change on database. For this data, it is only retrievals from Articles table in the database. This data should be processed so that which data about any article will be retrieved. This is done in Call Related Retrieve News Web Service process.

Name:	User Web Service Command
Aliases:	None
Where used/how used:	Process Commands 5.1 (Output)
	Handle User Web Service 6.5 (Input)

Description:

User Web Service Command specifies Users table will be accessed in the database. User Web Service Handler will manage this data to determine the effect of it on the database whether, it is retrieval or update command.

Name:	Handle Update User Web Service Request
Aliases:	None
Where used/how used:	Handle User Web Service 6.5 (Output)
	Call Related Update User Web Service 10.3 (Input)

Description:

This data is an update command web service for users. An update on Users table will flow through this data. Although mostly account information of any user may be changed by admin of NewsAgent, users themselves can, of course, change their account information. All these changes on Users table is named as an update in web service of NewsAgent.

Name:	Handle Retrieve User Web Service Request
Aliases:	None
Where used/how used:	Handle User Web Service 6.5 (Output)
	Call Related Retrieve User Web Service 9.3 (Input)

Description:

This data is a retrieve command web service for users. Retrieval is any access to the database that does not cause any change on database. For this data, it is only retrievals from Users table in the database. This data should be processed so that which data about any article will be retrieved. This is done in Call Related Retrieve User Web Service process. Mostly retrieving any user account information will be accessed by admin of NewsAgent.

Name:	Newsgroup Update Info
Aliases:	None
Where used/how used:	Call Related Update Newsgroup Web Service 9.1 (Output)
	Create Mail Sender Object 8.2 (Input)

Description:

This data specifies all changes on Newsgroups table on the database. Any update information for Newsgroups table will flow through this data. Newsgroup name update is an instance of such data. This data specifically used for sending mails to all users who request mails from news server or only users who request mail from this newsgroup. For instance, when a new newsgroup is created, it is reasonable to send mail to all mail users of NewsAgent, however when a name update of a newsgroup is applied, it is reasonable to send mails only to mail users who request mail only from the updated newsgroup.

Name:	News Update Info
Aliases:	None
Where used/how used:	Call Related Update News Web Service 10.2 (Output) Create Mail Sender Object 8.2 (Input)

Description:

This data specifies all changes on Articles table on the database. Any update information for Articles table will flow through this data. Article name update is an instance of such data. This data specifically used for sending mails to all users who request mails from news server or only users who request mail from the newsgroup that the article belongs to.

Name:	Update User Request
Aliases:	None
Where used/how used:	Call Related Update User Web Service 10.3 (Output) Users (Input)

Description:

This data specifies all changes on Users table on the database. Any update information for Users table will flow through this data. User name update by an admin is an instance of such data.

Name:	Retrieve Newsgroup Request
Aliases:	None
Where used/how used:	Call Related Retrieve Newsgroup Web Service 10.1 (Output) Newsgroups (Input)

Description:

This data specifies all retrieves from Newsgroups table on the database. Any retrieval information from Newsgroups table will flow through this data. Newsgroup name retrieval by a user is an instance of such data.

Name:	Retrieve News Request
Aliases:	None
Where used/how used:	Call Related Retrieve News Web Service 9.2 (Output) News (Input)

Description:

This data specifies all retrieves from Articles table on the database. Any retrieval information from Articles table will flow through this data. Article header retrieval by a user is an instance of such data.

Name:	Retrieve User Request
Aliases:	None
Where used/how used:	Call Related Retrieve User Web Service 9.3 (Output)
	Users (Input)

Description:

This data specifies all retrieves from Users table on the database. Any retrieval information from Users table will flow through this data. User name retrieval by another user is an instance of such data.

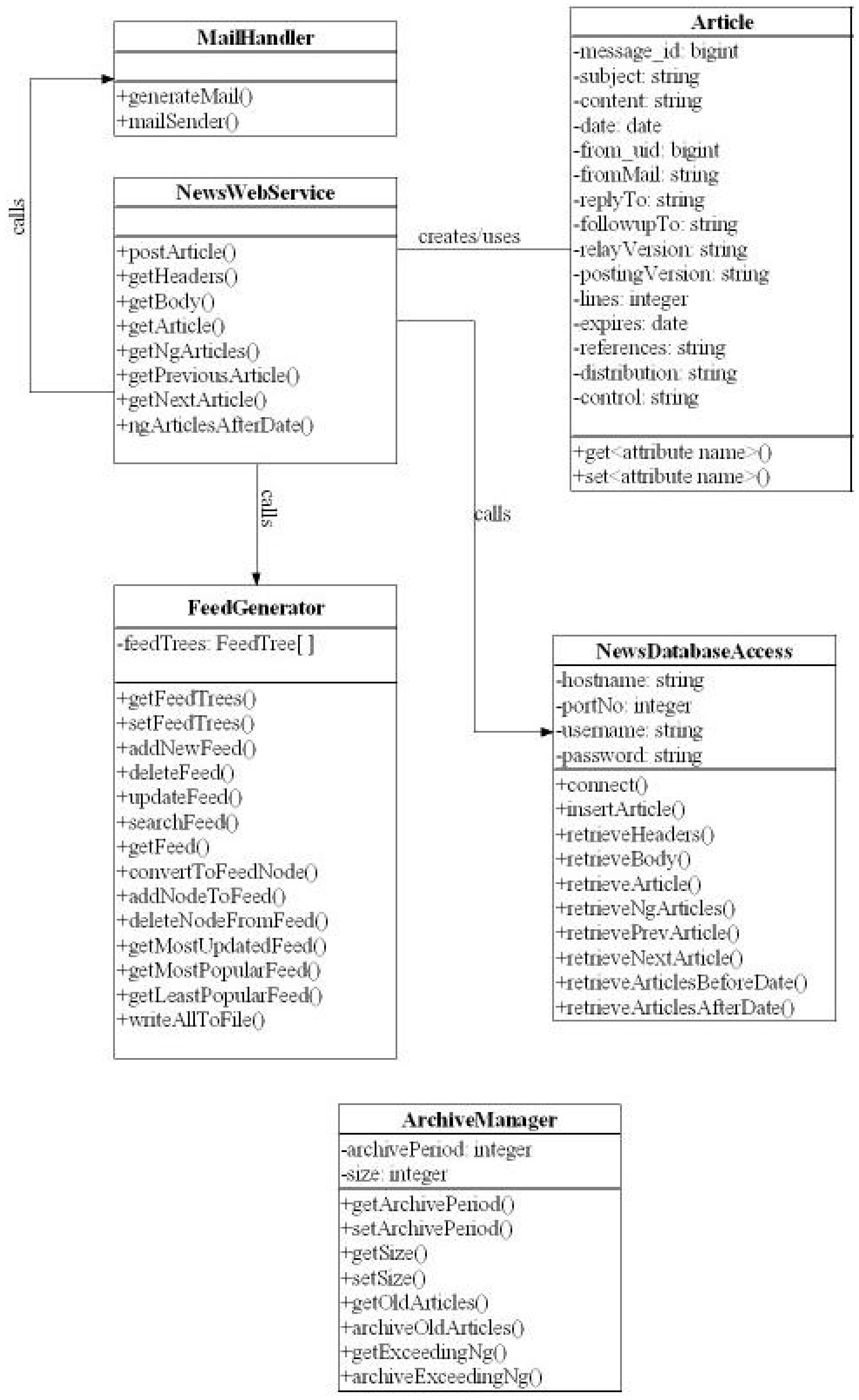
Name:	Status Info
Aliases:	None
Where used/how used:	Database (Output)
	Update/Retrieval Web Service (Input)

Description:

This data specifies whether the update/retrieval is completed successfully or not. In fact, this data is used for controllable database applications.

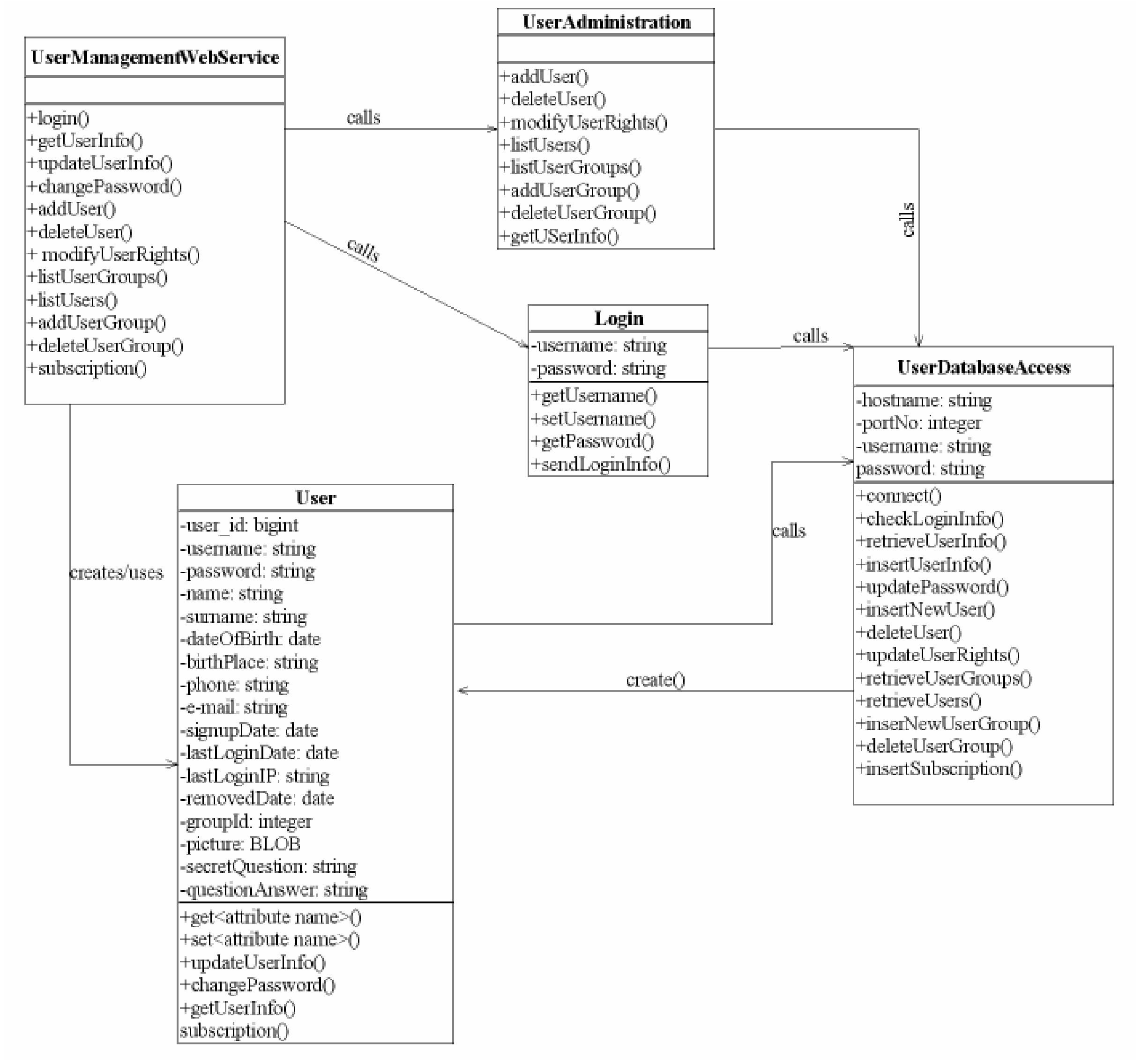
5 CLASS DIAGRAMS

5.1 Article Management Module



- **NewsWebService** class is a web service that maintains all methods required for news management. When it receives post article command, it calls MailHandler class and FeedGenerator class.
- **MailHandler** class sends e-mail to the users who are subscribed to the newsgroups those include that article. It is described in Mailing Module in detail.
- **FeedGenerator** class is called in order to append new article into feed. It is described in Feed Generator module in detail.
- **Article** class is created after a post article command. Created article instance is returned to NewsWebService class and NewsDatabaseAccess is called in order to insert that article to the database.
- **NewsDatabaseAccess** class establishes connection with the database and creates queries in order to retrieve data from database or insert data into database. Its methods use these queries and do all the work related with articles.
- **ArchiveManager** class works on its own an checks whether any newsgroup exceeds the size limit or any articles exceeds time limit. Archiving is done according to these parameters, the user selects which criteria to be used for archiving.

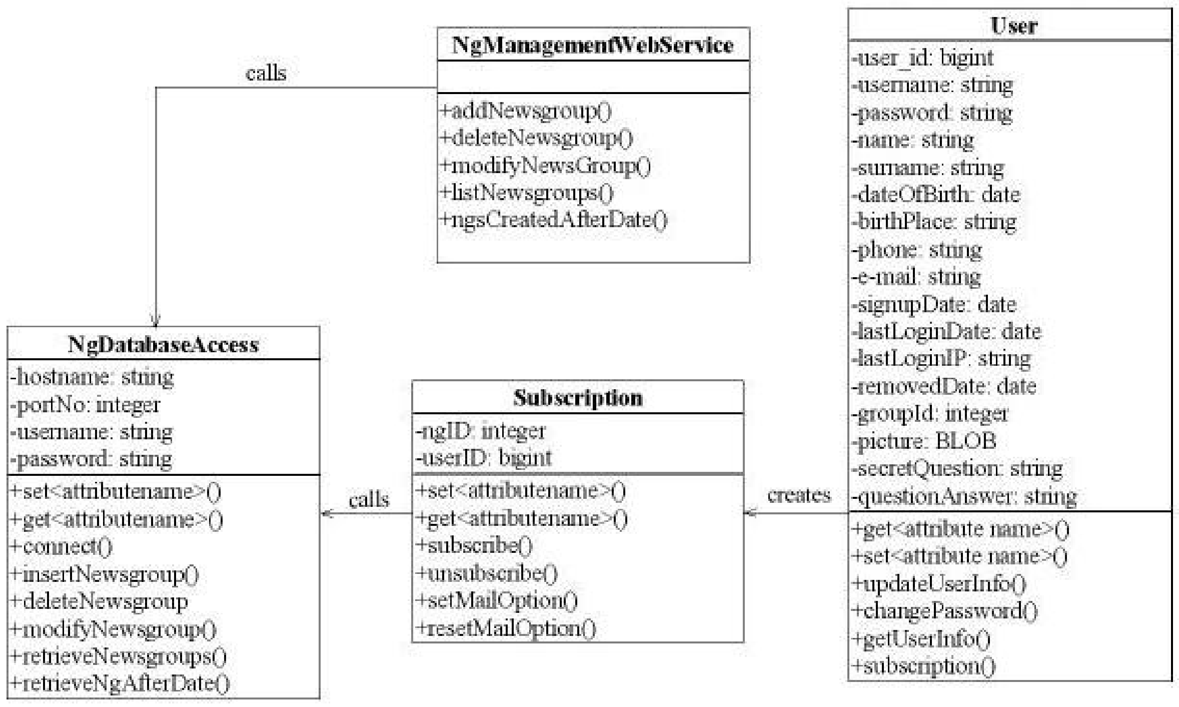
5.2 User Management Module



- **UserManagementWebService** class is a web service that maintains all methods required for user management. It calls UserAdministration, User and Login classes.
- **UserAdministration** class handles the administrative operations on users. When a user wants to add, delete, modify users and usergroups, the related methods are called and the modifications are reflected to the database. It calls UserDatabaseAccess class.
- **Login** class handles the login operation. It gets username and password and send login info to database in order to be checked. It calls UserDatabase Access class.
- **User** class handles the user related operations of the user management such as update user info, change login info, display user info etc. It calls UserDatabase Access class.

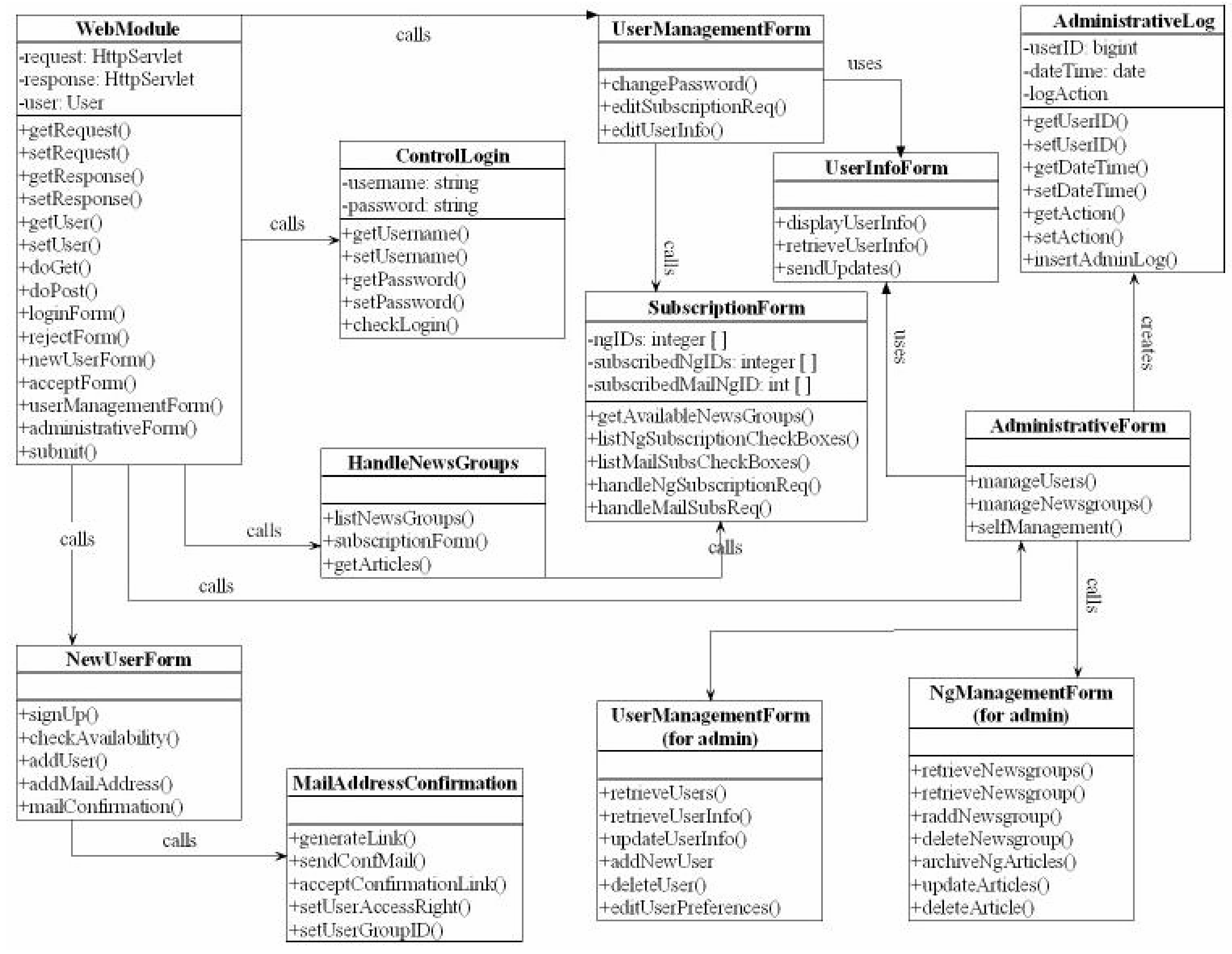
- **UserDatabaseAccess** class establishes connection with the database and creates queries in order to retrieve data from database or insert, delete and modify data into database. Its methods use these queries and do all the work related with users.

5.3 Newsgroup Management Module



- **NgManagementWebService** class is a web service that maintains all methods required for newsgroup management. When system administrators request to list, add, delete and modify a newsgroups, its methods addNewsgroup(), deleteNewsgroup(), modifyNewsgroup() are invoked and the modifications are reflected to the database.
- **NgDatabaseAccess** class establishes connection with the database and creates queries in order to retrieve data from database or insert, delete and modify data into database. Its methods use these queries and do all the work related with newsgroups.
- **Subscription** class handles the user's subscription and mailing option change. When a user wants to subscribe to a newsgroup or unsubscribe from an existing one or request to receive email related to the new posts to the newsgroup or request to cancel the mail receiving option set before, the methods of the Subscription class are activated and NgDatabaseAccess class is called in order to reflect the modifications to the database.

5.4 Web Module

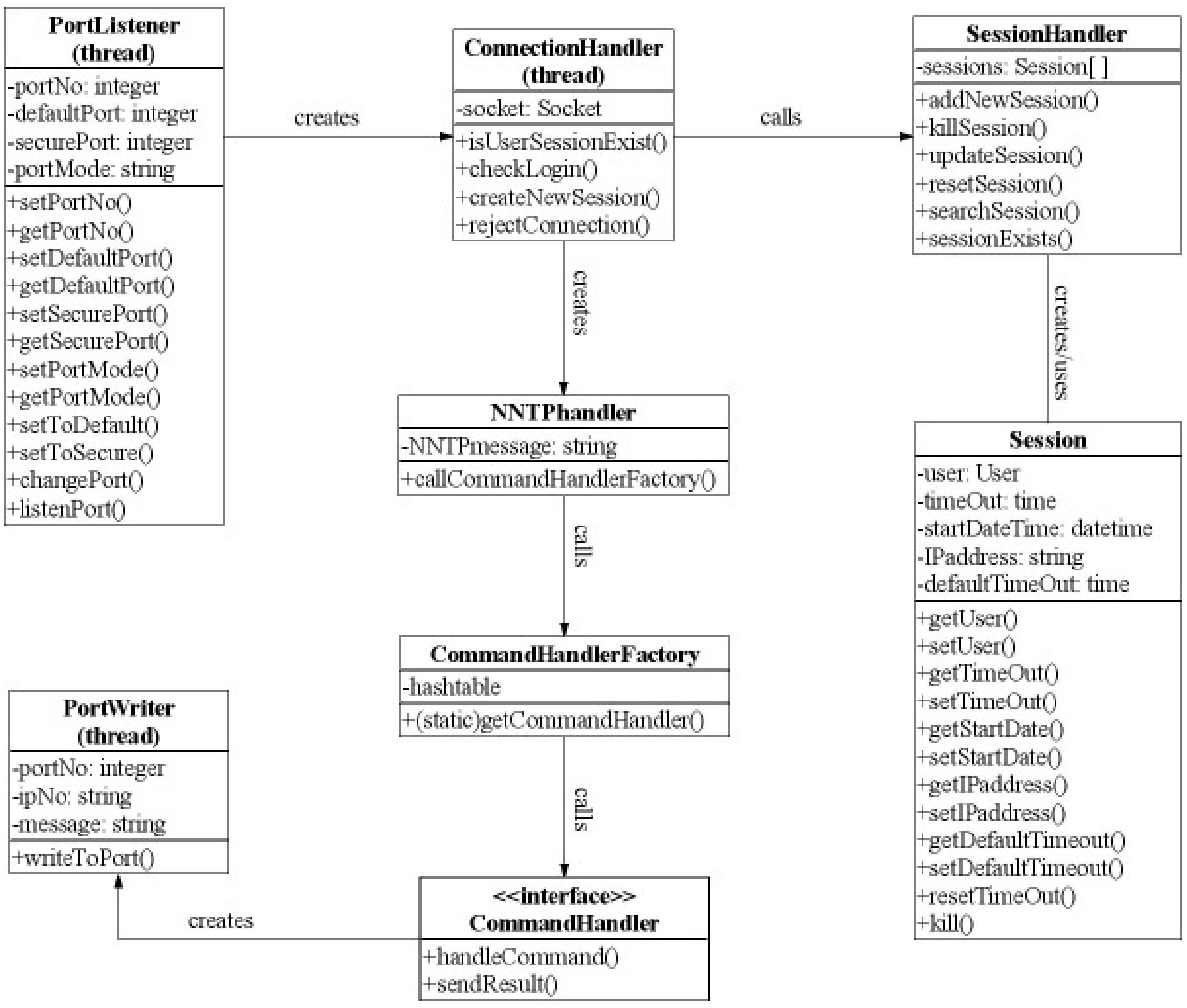


Web module classes are implemented in order to accomplish communication with the server via web. The main WebModule class includes user, request and response attributes. The user is an instance of User class, request is an HttpServletRequest and response is an HttpServlerResponse. According to the request, this class calls ControlLogin class, UserManagementForm class, HandleNewsGroups class, NewUserForm class or AdministrativeForm class.

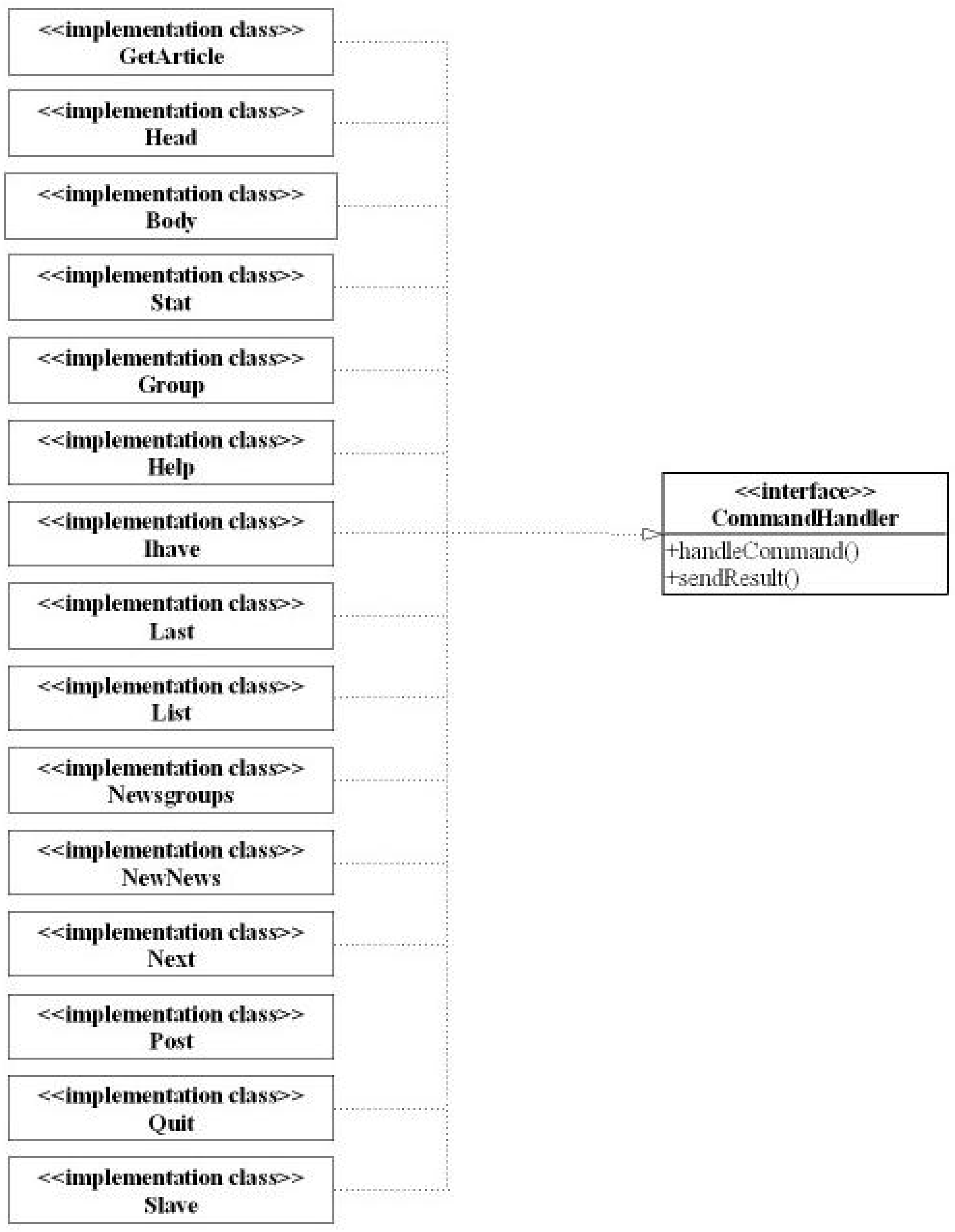
- **ControlLogin** class checks the login data (username and password) of the user from database through UserManagementWebService.
- **HandleNewsGroups** class handles requests related with newsgroups such as newsgroup listing, subscription and getting newsgroup articles. Listing and retrieving articles are handled by NgManagementWebService and subscription method calls SubscriptionForm. This class lists newsgroups than can be subscribed by that user, shows checkboxes stating whether subscribed or not, whether the user wants e-mail or not. If a user requests to subscribe, unsubscribe or set/reset mailing option, it handles these requests.
- **UserManagementForm** class includes methods that are related with the user's own modifications on his/her info. Change password is accomplished by UserManagementWebService, editing subscription info calls SubscriptionForm and editing user info uses UserInfoForm.
- **UserInfoForm** class displays user info, retrieves user's info after modifications and sends these info todatabase via UserManagementWebService.
- **NewUserForm** class is called when a new user wants to be added. It checks availability of the user to be added (e.g. e-mail conflict with another user or wrong e-mail), if it is available, user is added to the database and MailConfirmation is called.
- **MailConfirmation** class generates links and sends this link to the user via e-mail for confirmation. When user clicks the link from that e-mail, he/she will be authenticated and user rights, user group for that user will be set.
- **AdministrativeForm** class includes administrative actions which can be accomplished by admin type users. When an administrator modifies users, newsgroups or articles, that means UserManagementForm or NgManagementForm classes are called, actions realized by administrator are hold in an instance of the class AdministrativeLog class.

- **UserManagementForm** class includes methods related with the modifications on the users made by administrator. These modifications are retrieving users, retrieving and updating user info, adding and deleting users and editing user's preferences.
- **NgManagementForm** class includes methods related with the modifications on the newsgroups made by administrator. These modifications are retrieving newsgroup names, retrieving a specified newsgroup, adding and deleting newsgroups, archiving and articles. When administrator creates a newsgroup, he/she sends e-mail to all users and when a newsgroup is deleted, an e-mail is sent to the users who are subscribed to that newsgroup. Retrieving and updating user info methods use UserInfoForm class.

5.5 NNTP Commands Module



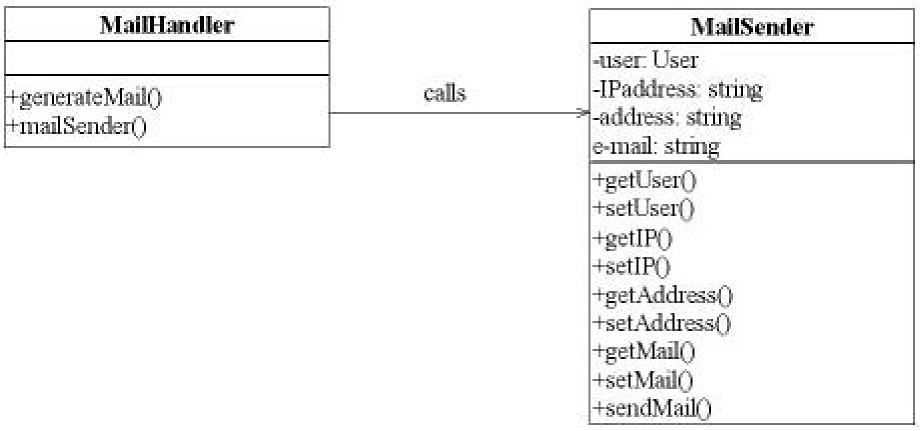
- **PortListener** class is a thread and listens the specified port continuously. When a new message arrives, an instance of **ConnectionHandler** class is created. The information about sender of the message also arrives when message is sent. With this information, user's session info is checked by calling SessionHandler class. If user's session exists, it is updated. If not, a new session is created after username and password check.
- **SessionHandler** class calls session related methods. If a new session is created, **Session** class is instantiated and returned to SessionHandler and added to the sessions array.
- **NNTPhandler** class is created by ConnectionHandler. ConnectionHandler passes the message it received from the socket to NNTPhandler. NNTPhandler calls **CommandHandlerFactory** which has a hashtable including available NNTP commands. According to the command, it calls related implementing class of interface **CommandHandler**. CommandHandler creates PortWriter after handling the command.
- **PortWriter** class receives the result of the command and writes it back to the port.



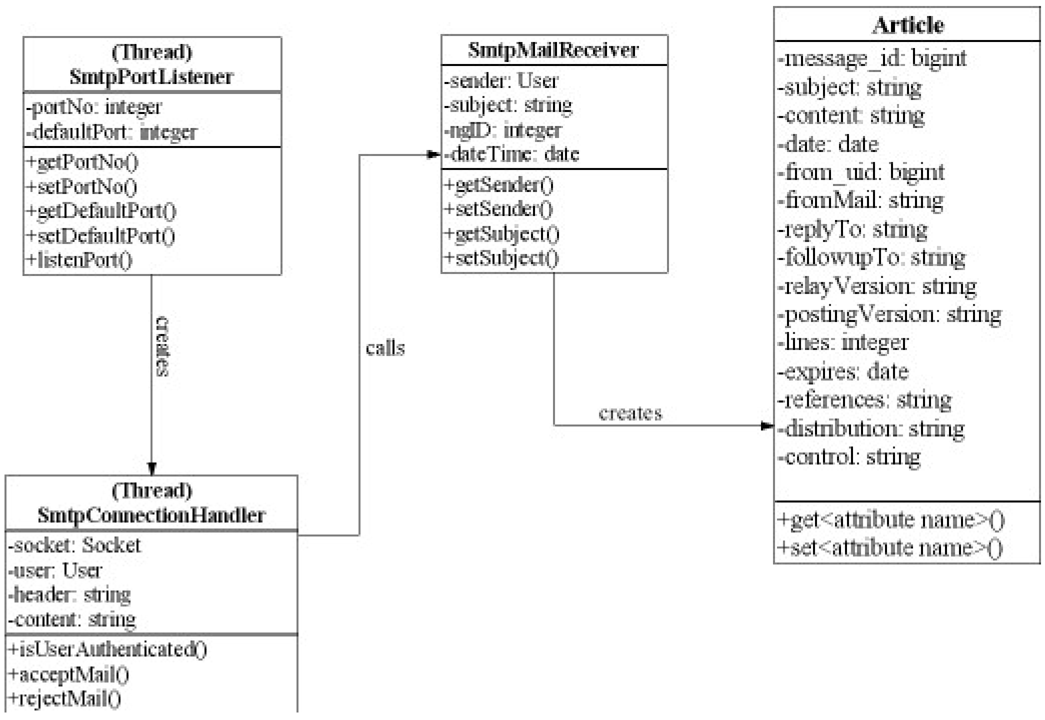
NNTP extension commands are not considered for initial design of NewsAgent, but the interface modularity of interface CommandHandler is very extensible to add new commands for command handling operations.

5.6 Mailing Module

Sending Mail



Receiving Mail



Since NewsAgent maintains the functionality to send e-mail to the users and receive e-mail from users, mailing module is examinde in 2 subparts. First part is mail sending; that means sending mail to the users who wanted to receive mail from the newsgroups that he/she is subscribed to. Second part is mail receiving; that means receiving the e-mails from users and inserting them into database as if they were posted from web or NNTP.

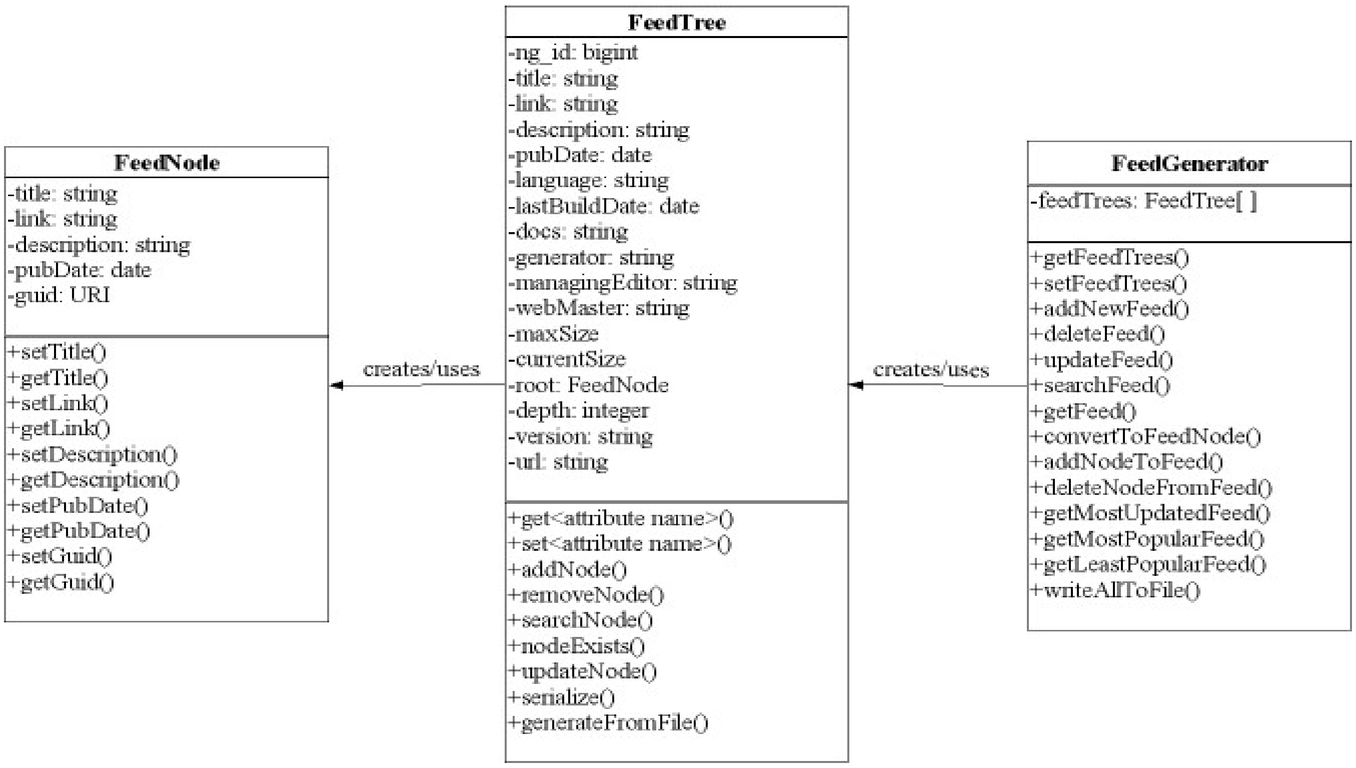
For the First part:

- **MailHandler** class is called when a new article is posted, inserted into database and a message is returned as it is inserted into database. It generates e-mail using the header, sender and body of article it received and creates an instance of MailSender.
- **MailSender** class maintains the information about the user and sends e-mail to the user via smtp.

For the Second Part:

- **SmtpPortListener** class is a thread. It listens the specified port and creates an instance of SmtpConnectionHandler when a message is received from that port.
- **SmtpConnectionHandler** class checks whether the user who sends the e-mail is authenticated or not. According to the result of this check, it accepts or rejects the user. After acception, it calls SmtpMailReceiver.
- **SmtpMailReceiver** class creates an instance of article class and creation of this article calls the related web service and then the article is inserted into database.

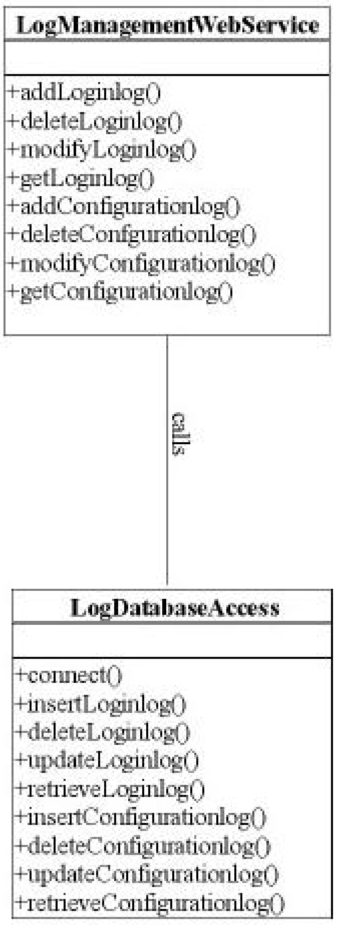
5.7 RSS Module



Users will be able to reach hot news from NewsAgent using their RSS readers. For this reason, we create an RSS feed including recently posted news. This module deals with RSS related jobs.

- **FeedGenerator** class is called when a new article is posted and inserted into the database. As shown in article management module, article management web service calls this class. FeedGenerator class has an array of feed trees which are instances of FeedTree class. Each newsgroup has its own feed tree, since a user may subscribe to any of them individually. For example, if the web service for inserting an article is invoked, it generates a request to the FeedGenerator after confirming the insertion of the article to the database. This request is to add a new entry for the specified newsgroup tree associated with the newly added article. FeedTree finds the corresponding feed tree and calls the method to add the article to the tree. Update and delete operations follows the same steps as in adding a new article.
- **FeedTree** class is a tree of feed nodes. It is a logical representation of the xml document. The listed methods above maintains the tree. Each tree has a maximum size. When the tree exceeds this size, the oldest entry of the tree is deleted to maintain the size. After each change operation to the tree, it serializes the tree to the file path specified by "url" attribute of the class. Now, any feed aggregators realize the changes when it checks out the feed for new news.
- **FeedNode** is a logical representation of the xml of a single article. It is appended to the related feed tree when a new post is inserted into database.

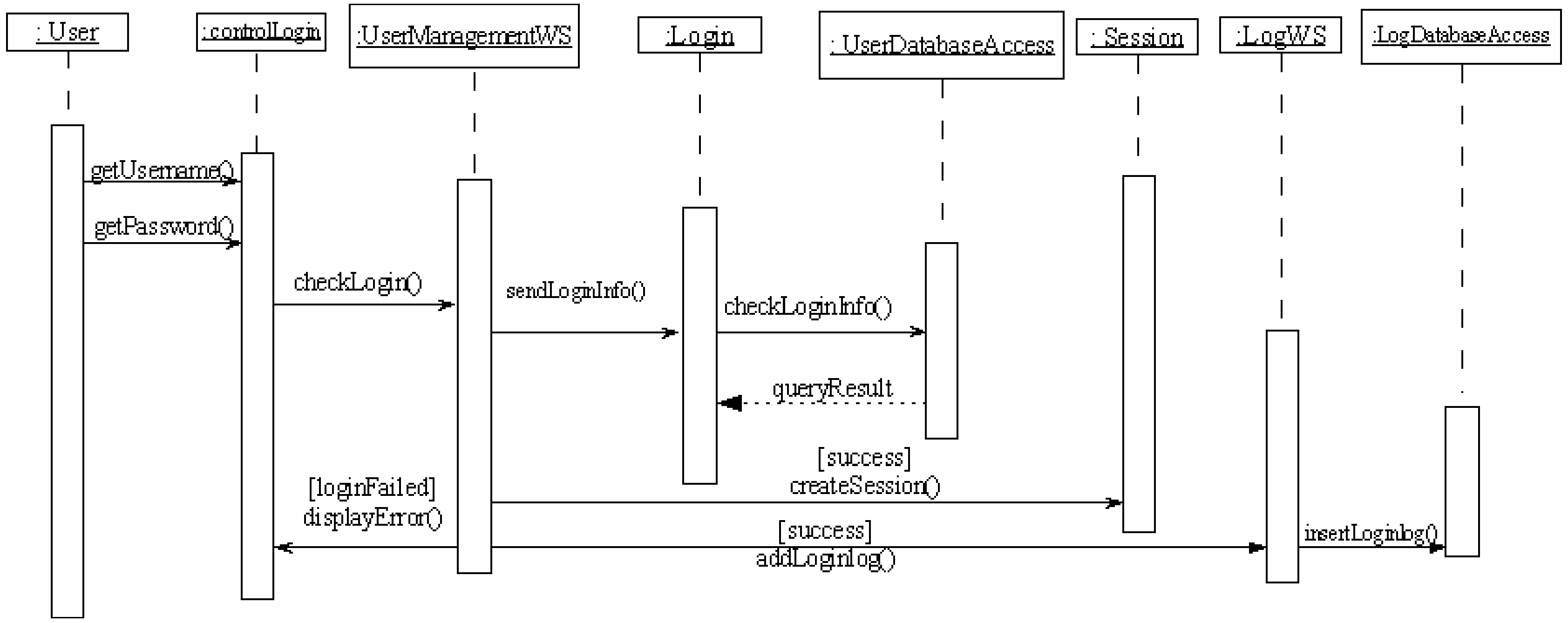
5.8 Log Module



- LogManagementWebService class is a web service that maintains required methods for login log and configuration log operations. This class calls LogDatabaseAccess class to reflect the modifications into the database.
- LogDatabaseAccess class establishes connection with the database and creates queries in order to retrieve data from database or insert and modify data into database. Its methods use these queries and do all the work related with logs.

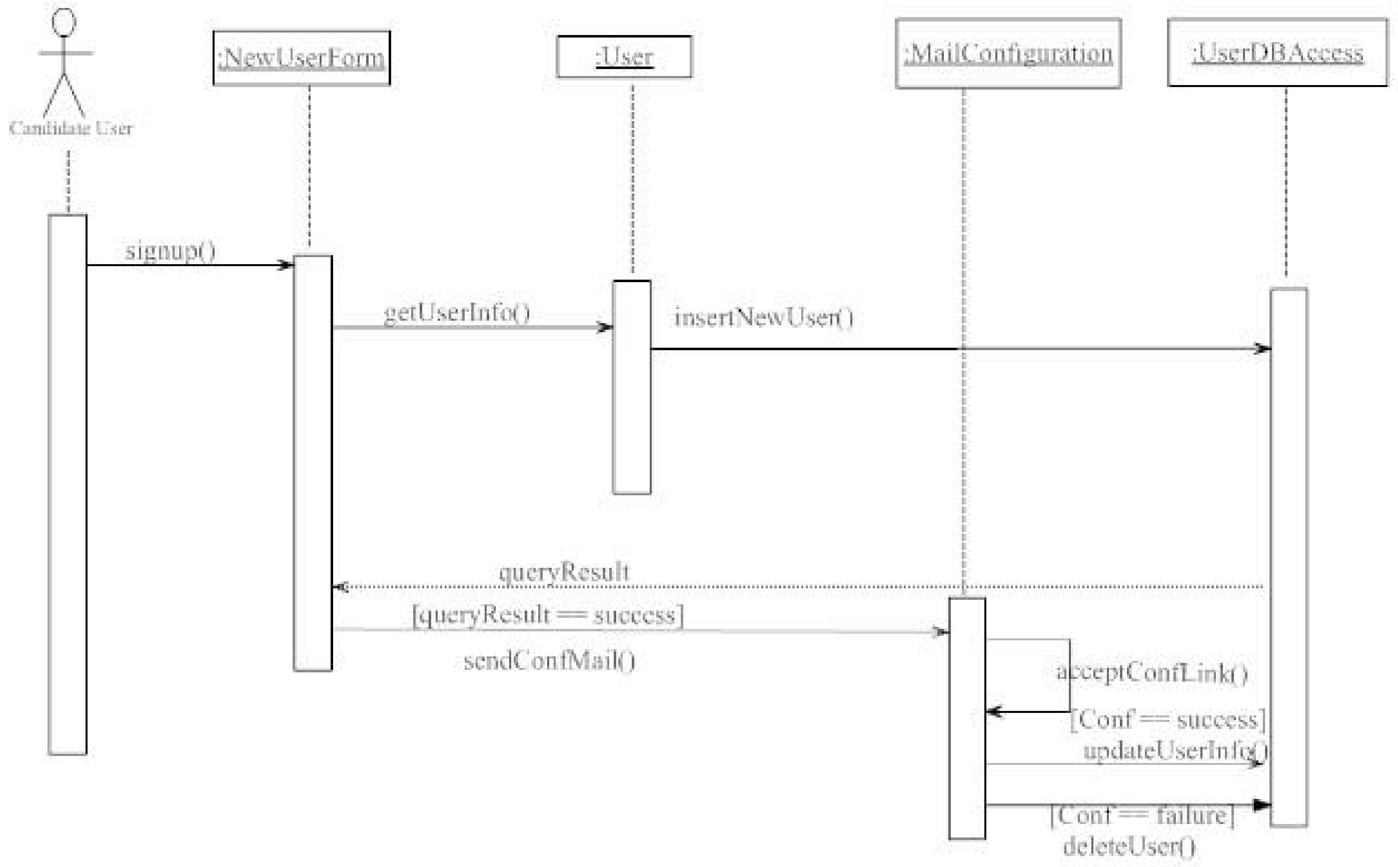
6 SEQUENCE DIAGRAMS

6.1 Login and Authentication

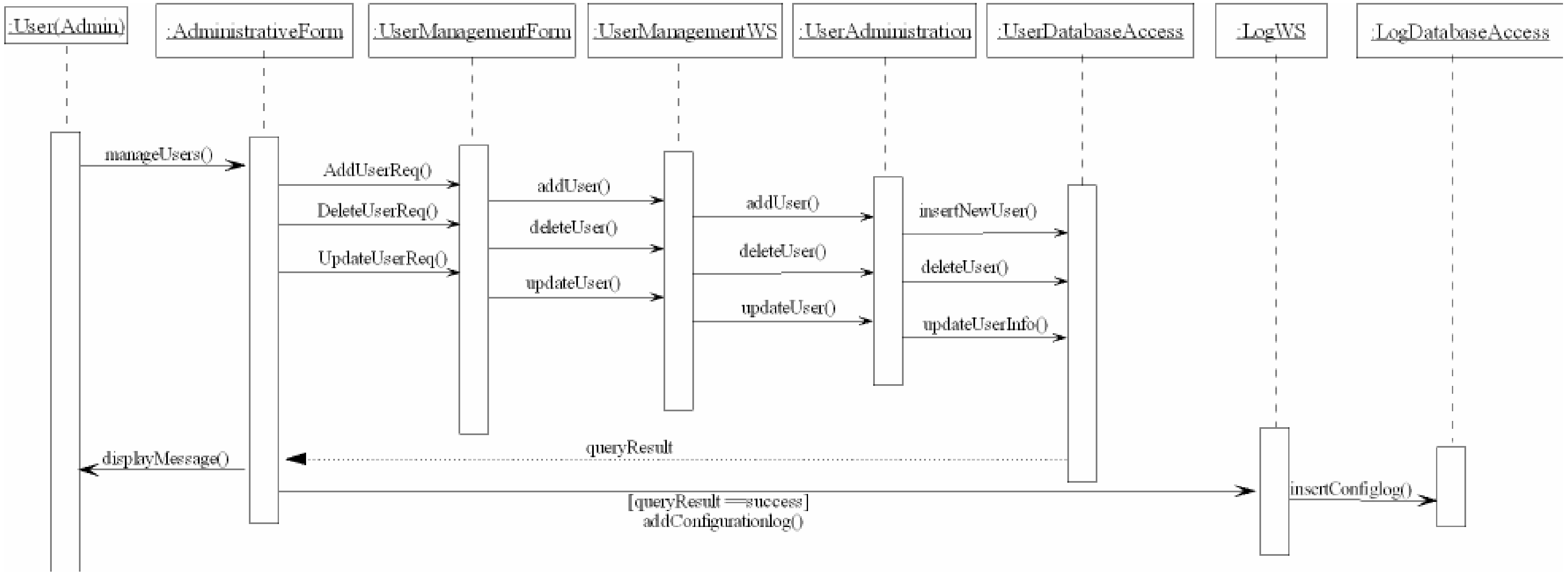


: User
:controlLogin
:UserManagementWS
:Login
: UserDatabaseAccess
: Session
:LogWS
:LogDatabaseAccess
getUsername()
getPassword()
checkLogin()
sendLoginInfo()
checkLoginInfo()
queryResult
[success]
createSession()
[loginFailed]
displayError()
[success]
addLoginlog()
insertLoginlog()

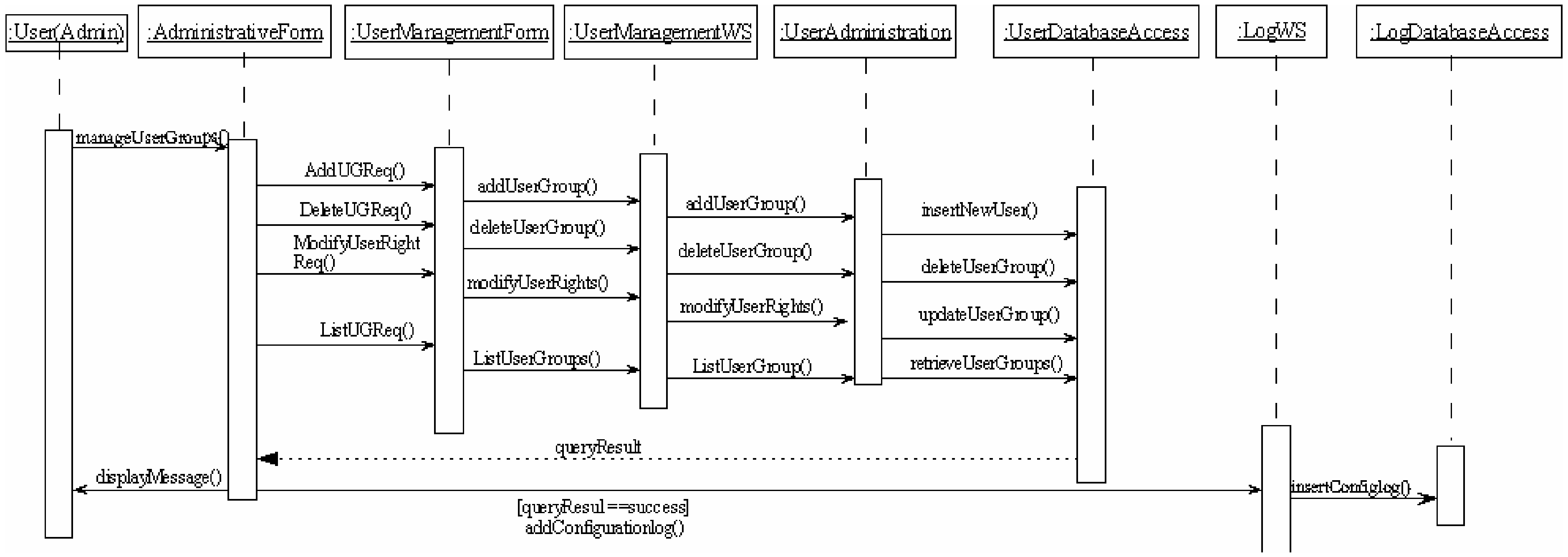
6.2 Signup


Candidate User
:NewUserForm
:User
:MailConfiguration
:UserDBAccess
signup()
getUserInfo()
insertNewUser()
queryResult
[queryResult == success]
sendConfMail()
acceptConfLink()
[Conf == success]
updateUserInfo()
[Conf == failure]
deleteUser()

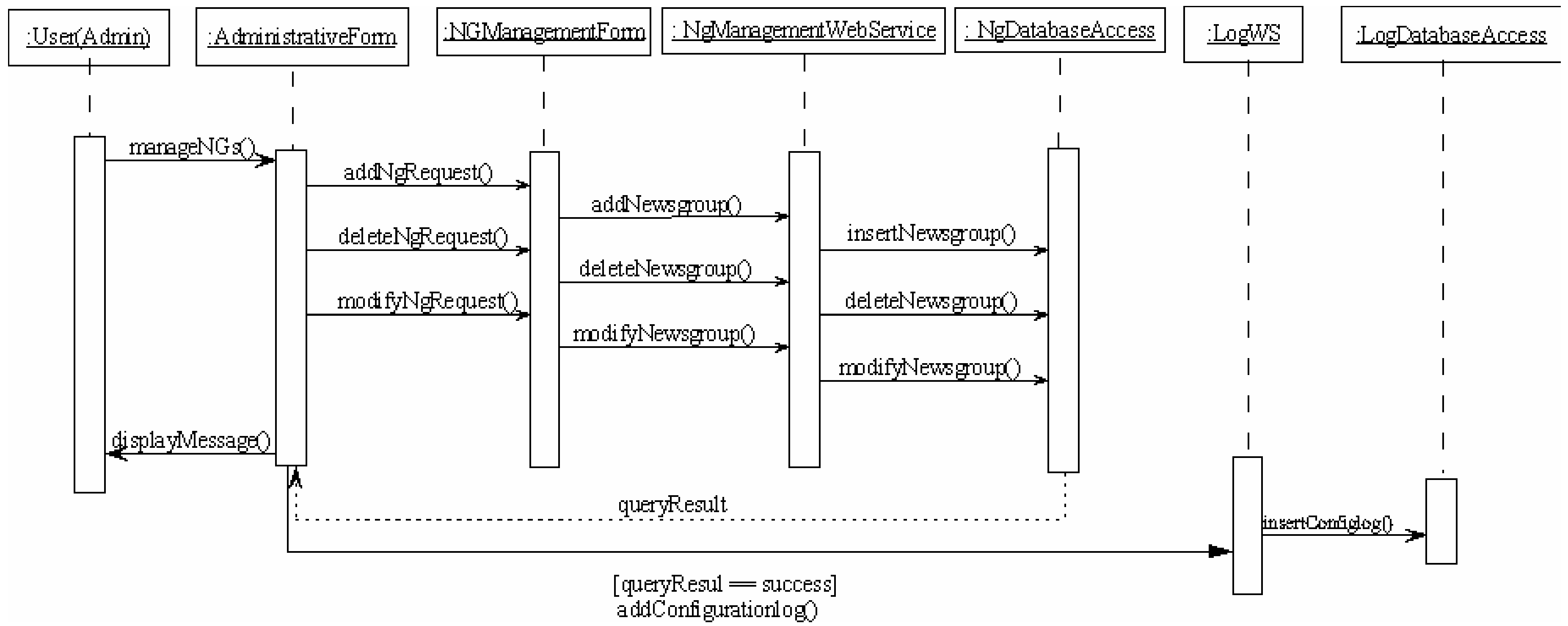
6.3 User Management



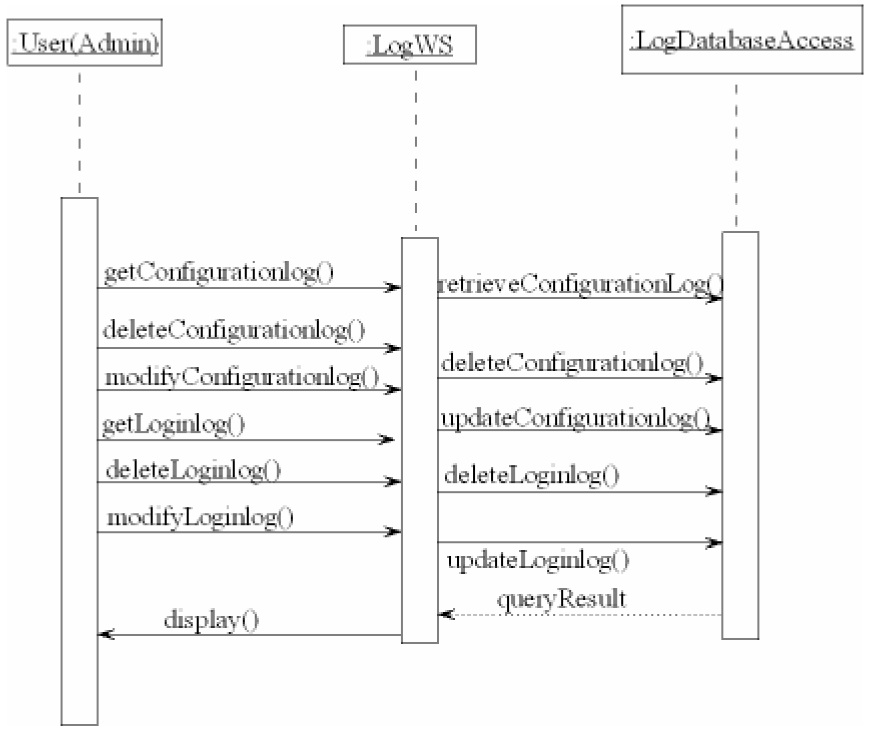
6.4 Usergroup Management



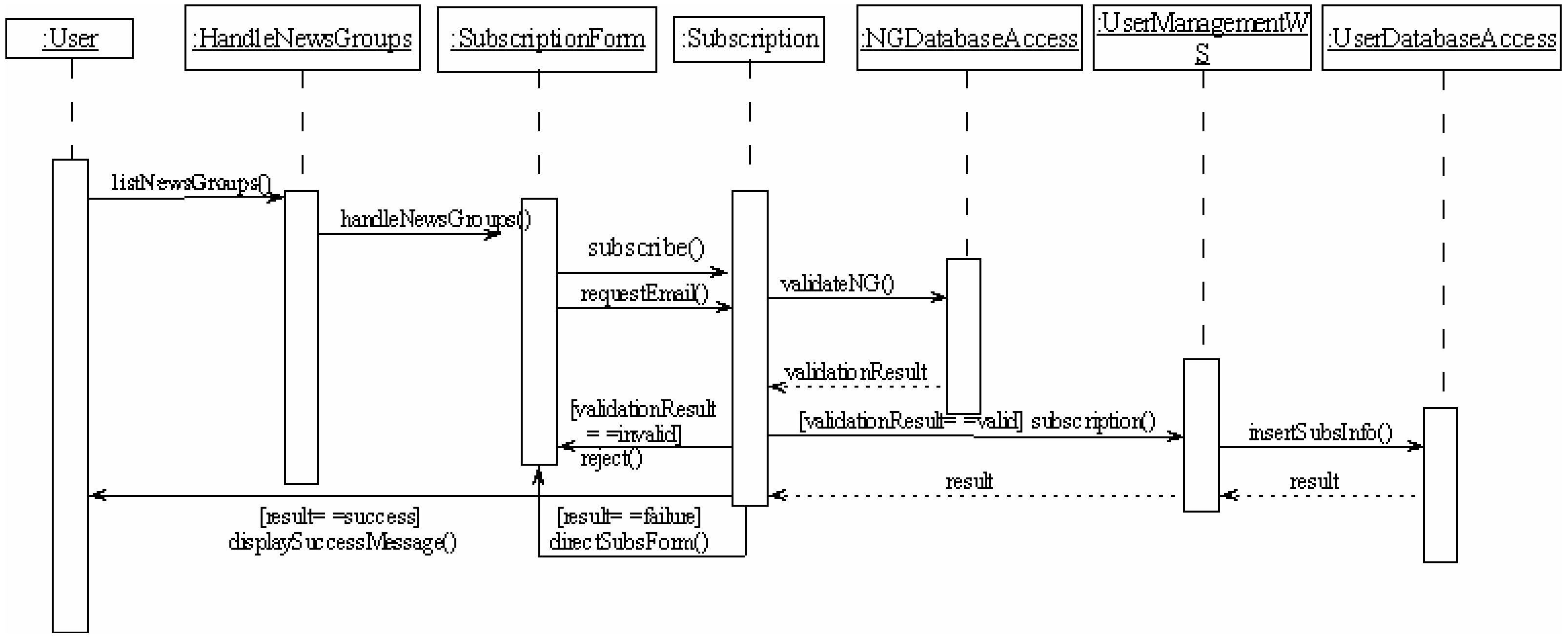
6.5 Newsgroup Management



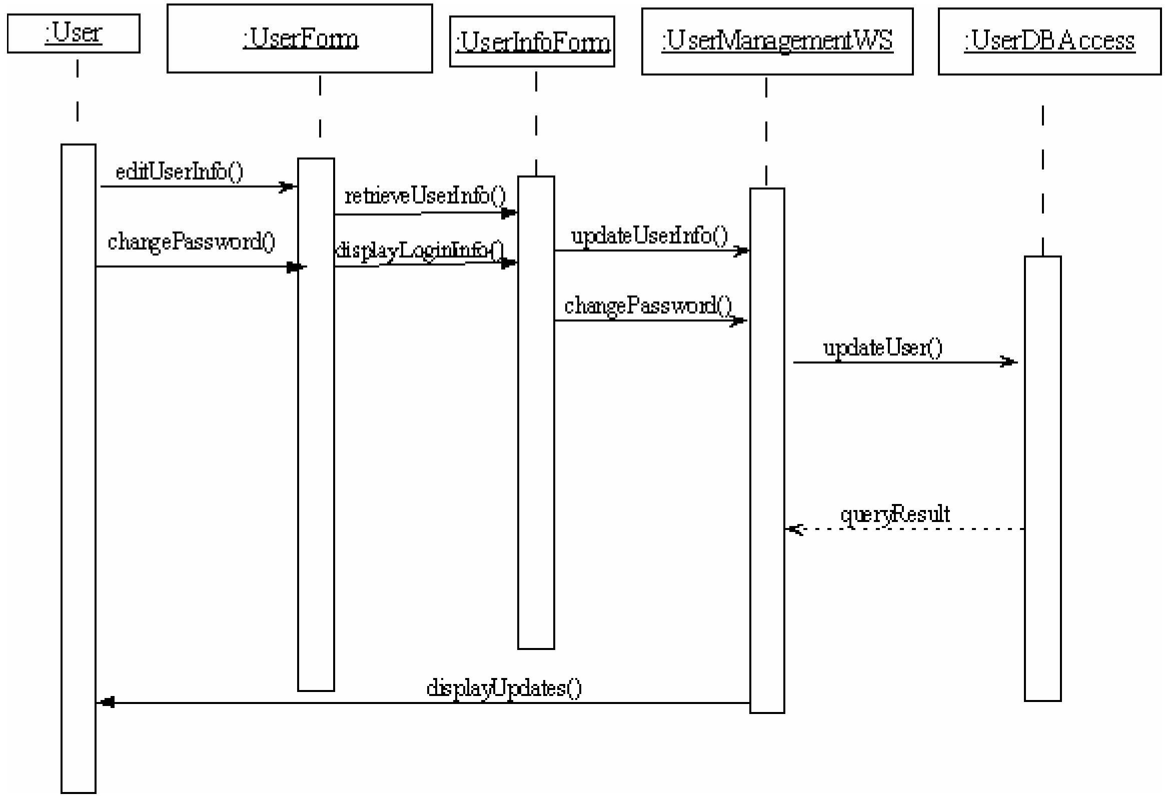
6.6 Admin Log Control



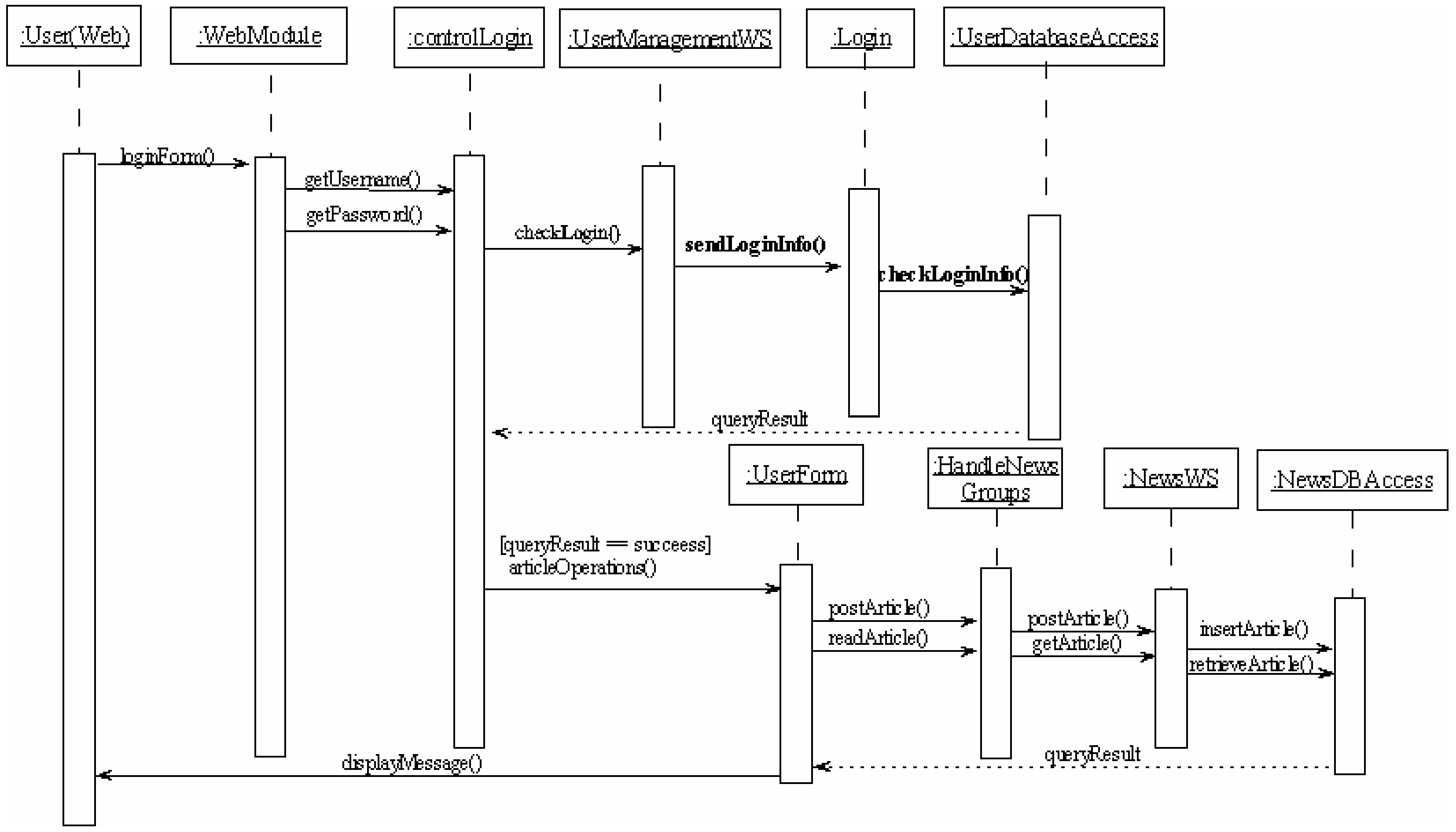
6.7 Subscription



6.8 Update User Info

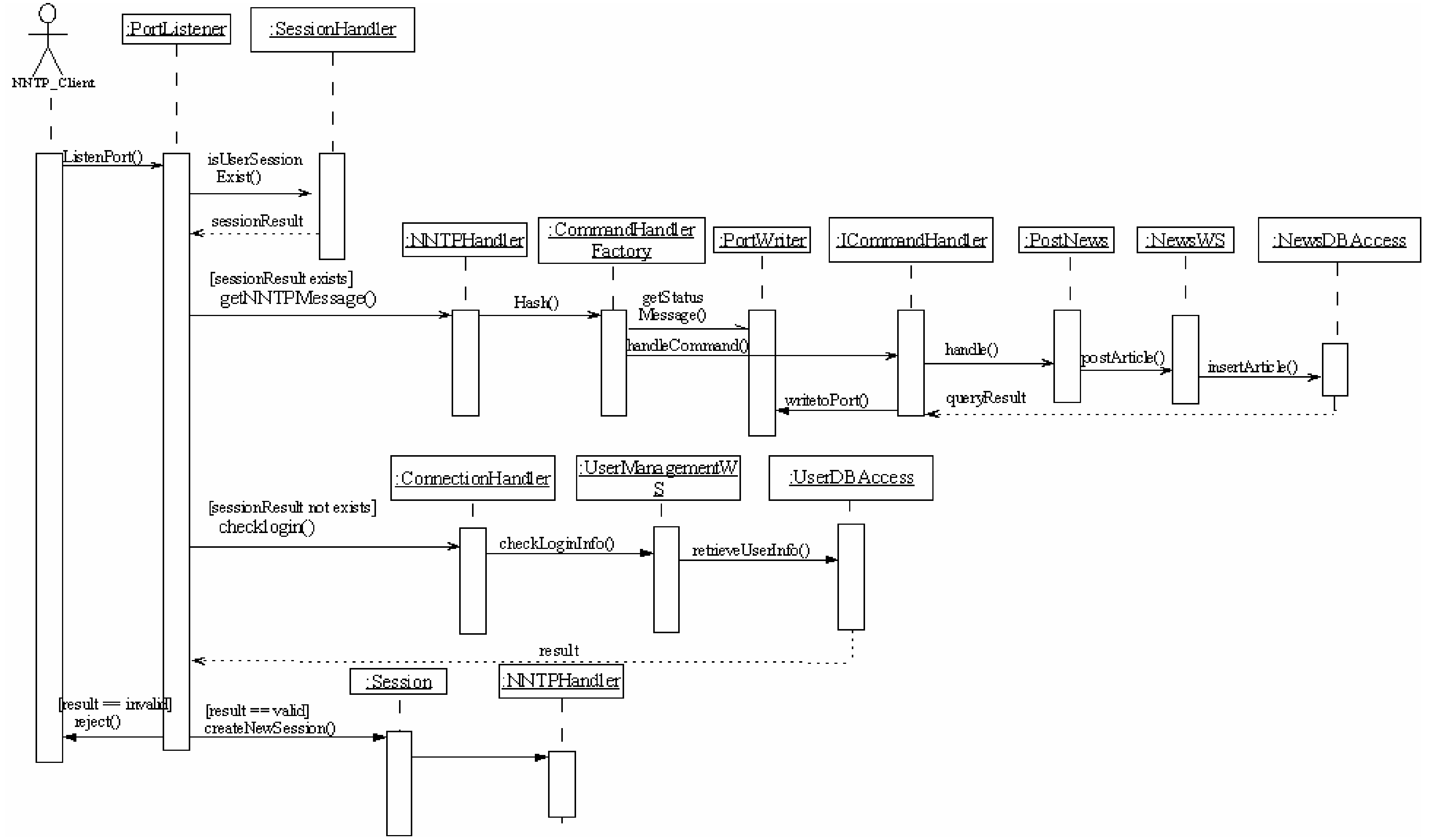


6.9 Web User Operations

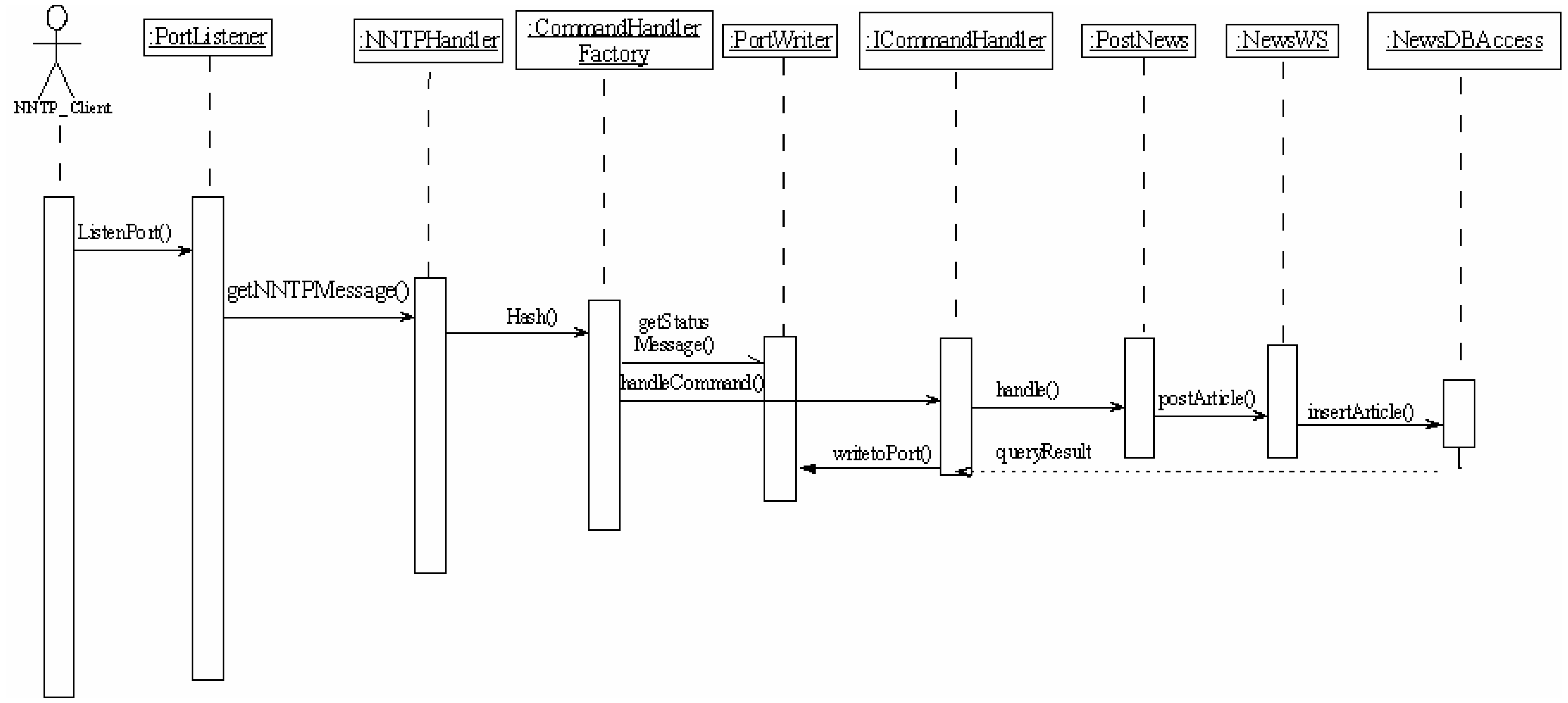


For unauthenticated web users, the user does not login to the system and can request only a small set of article operations.

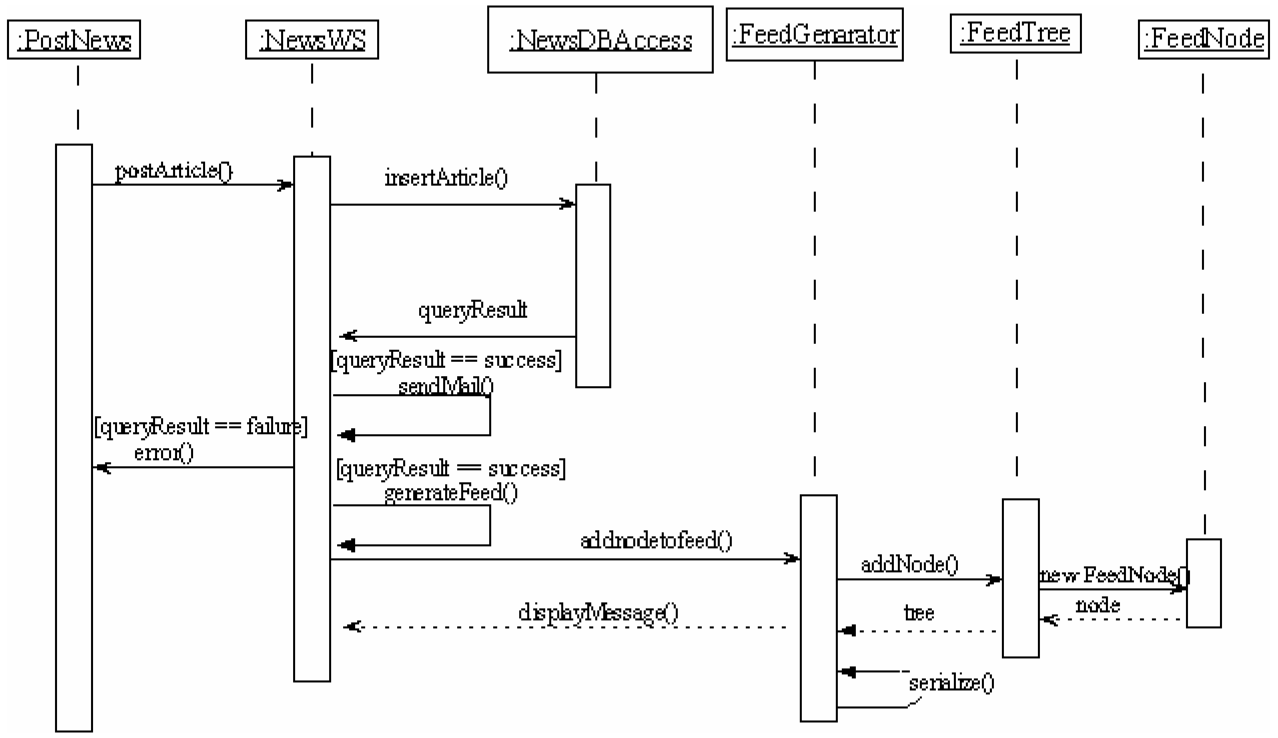
6.10 Authenticated NNTP User Operations



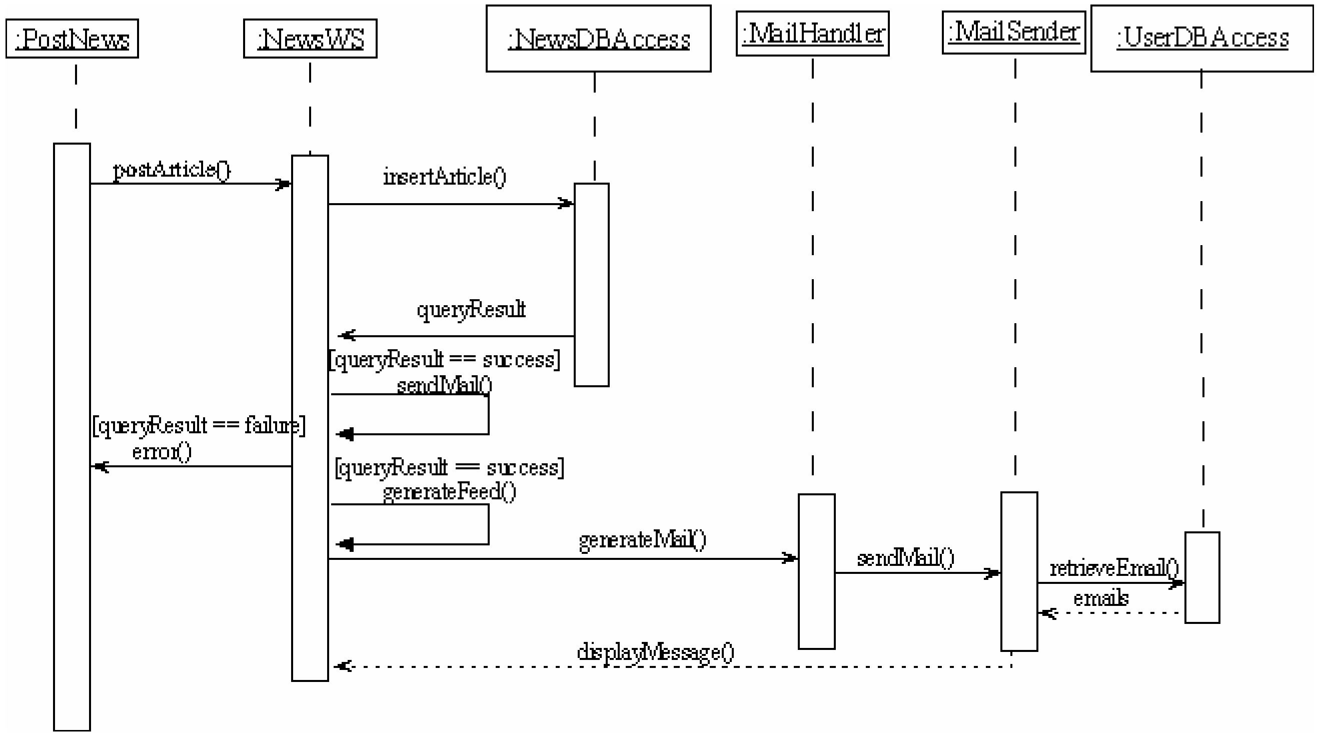
6.11 Unauthenticated NNTP User Operations



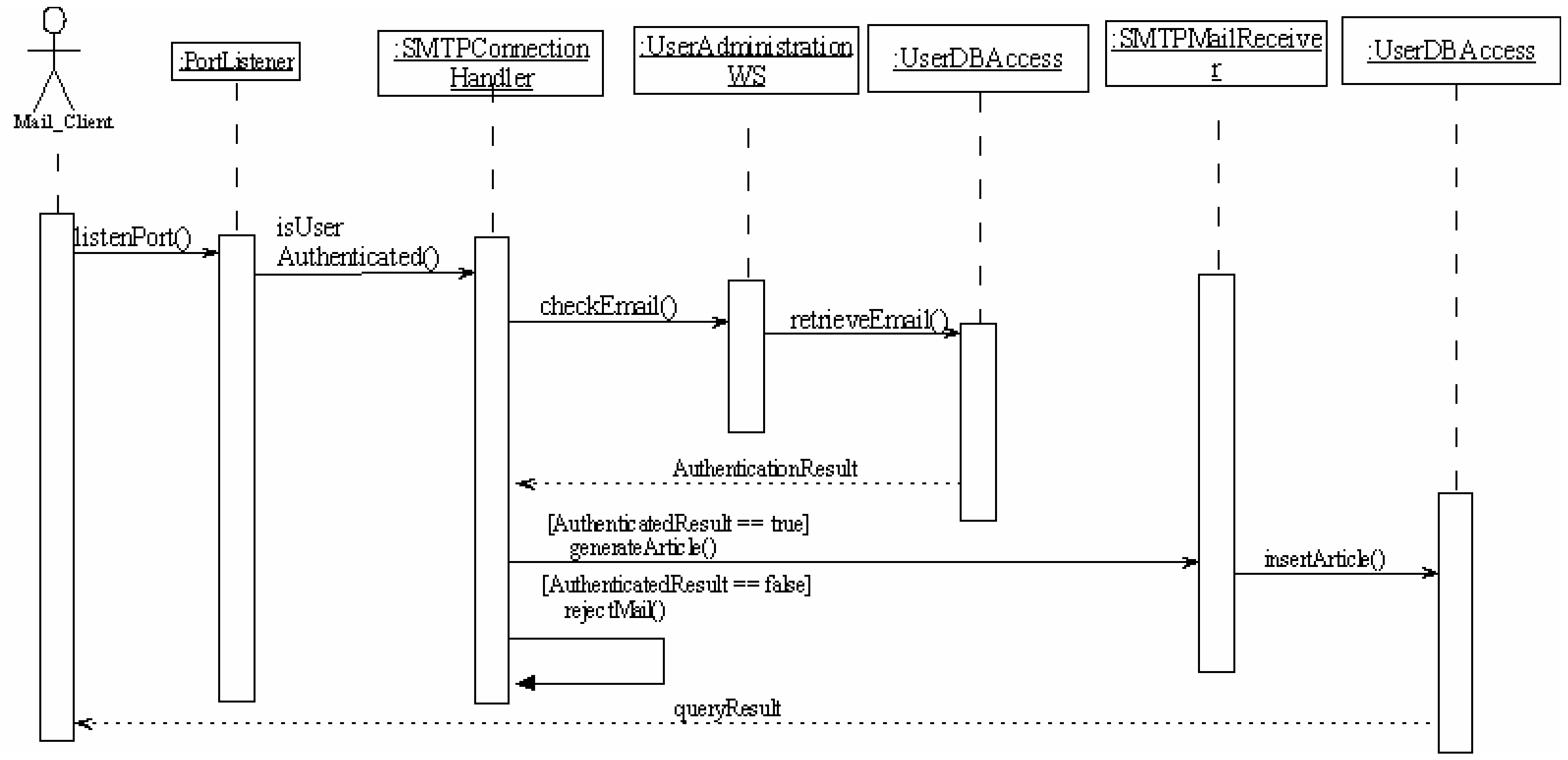
6.12 Feed Generation



6.13 Sending E-mails



6.14 Receiving E-mails



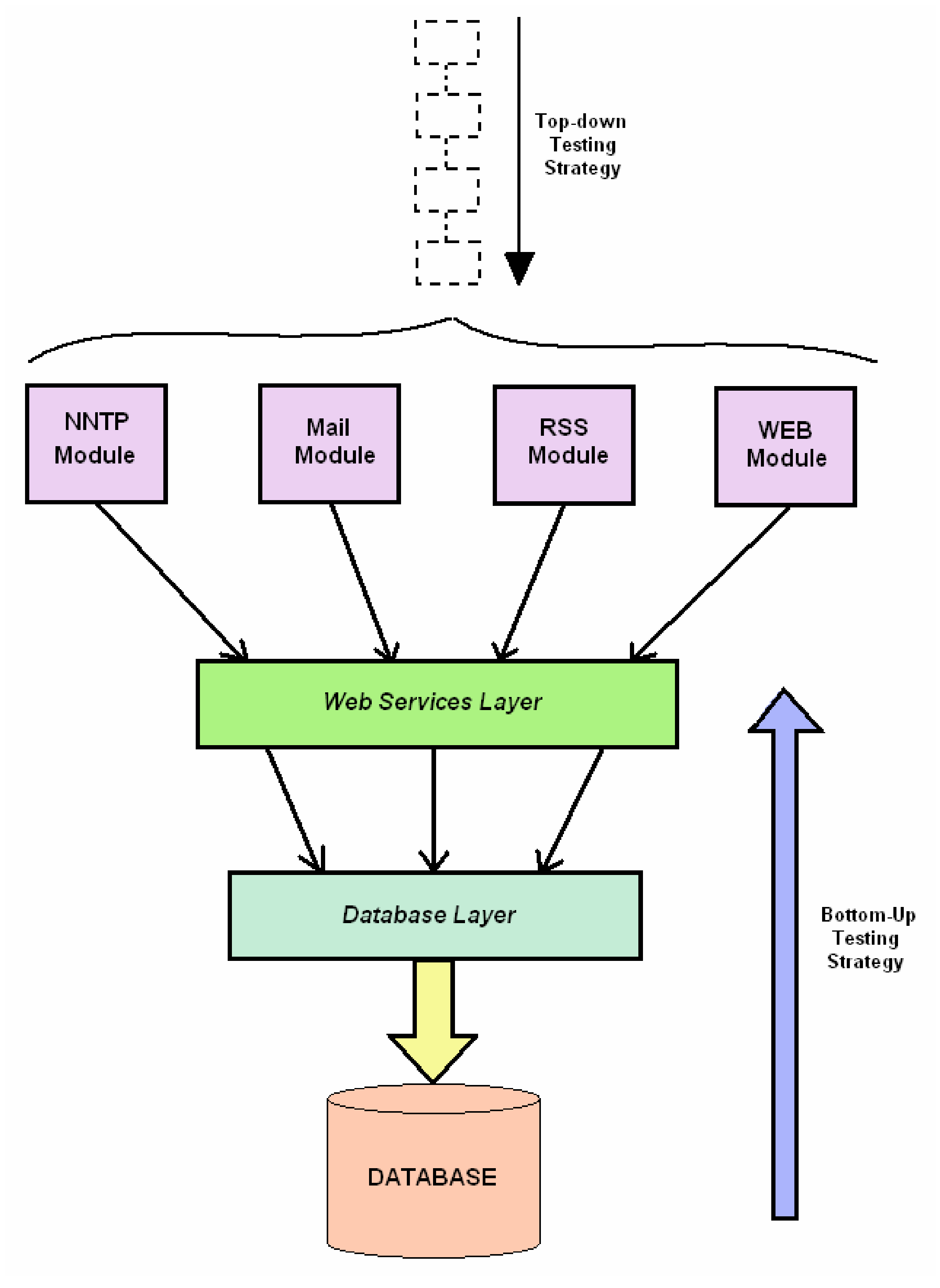
7 TESTING PLAN AND PROCEDURES

7.1 Testing Plan

Our aim is to find errors and make a good test that has a high probability of finding an error. We also want to make sure that there are no defects in the product.
After we have generated the source code, we are going to test our program to identify the errors and remove them before delivery to the customer. Our goal is to correct as many errors as possible early in our software development cycle. In order to acquire this we have to design a series of test cases that have a high likelihood of finding errors.

7.2 Testing Strategy

Since NewsAgent has different layers and modules, testing strategy will differ for each subpart of the product. We present a testing schema below, which will briefly explain our testing strategy.
In general, we will follow a bottom-up strategy for testing. Therefore, we will start from database layer as shown in the schema. For this layer, we will apply unit tests in order to check performance and correctness of our database queries. We will test our retrievals, insertions and modifications. Testing of this part is very important since each web service and its methods use the data returned from database layer and insert data into database through this layer. Any mistaken coding error in this layer can cause many problems in above layers.
After testing database layer, we will pass to web services layer. Any operation in NewsAgent will be handled by web services. So testing this part is another important issue in testing the product. For testing our web services, we will deploy each of them separately and invoke related methods. We will check whether each web service works correctly.
Then we will test our modules; NNTP Module, Mail Module, RSS Module and Web Module. While testing these modules, we will follow a different strategy which is top-down testing strategy.



Testing Strategy of NewsAgent

7.3 Testing Procedure

7.3.1 Unit Testing

In the unit test case we will be testing the separate modules of the software. White box testing will be used where each module or component of the software is tested individually. By this type of testing we have advantages as mentioned below.

i) As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
ii) The other advantage of white box testing is that it helps in optimizing the code
iii) It helps in removing the extra lines of code, which can bring in hidden defects.

We will be carrying out unit testing in order to check if the particular module or unit of code is working fine. The Unit Testing comes at the very basic level as it is carried out as and when the unit of the code is developed or a particular functionality is built.

We will be looking for entry and exit conditions of the data. We will make sure that all the components work without any troubles. The test primarily is carried out by the programmer who designed and implemented the module. Lead tester will than carry out test on the modules to finalize the testing.

7.3.2 Integration Testing

In this testing period we will be looking for any signs of the collision between our software components and those of the clients. We want to make sure there is no confusion among the application on the network when they are running simultaneously.
As we know, integration testing is testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications, client and server applications on a network, etc. And this type of testing is especially relevant to client/server and distributed systems. We will be carefully looking for any sort of collision between several different applications.

7.3.3 Security Testing

Testing the security of a news server is really a key point and also testing is an inevitable feature of NewsAgent. Since NewsAgent may be used in workplaces or foundations where security of data is the most important issue, security should be handled carefully. NewsAgent will use SSL for handling security issues. SSL provides data encryption which will be used in transmission of passwords. Also, newsgroups and articles should not be accessed by users who have not right to access them. Security testing will be done by controlling the flow of data in different modules of NewsAgent and will be useful for finding out any security holes.

8 SYNTAX SPECIFICATION

Coding standards occupy large amounts for big projects which have multiple developers and coders. These standards are so important that some big companies, military services and governmental services only rely on the products which have been produced through a very strictly specified line. This line is determined by the rules. Every developer included in the project must obey these rules.
Not being a big company, even not a company, we can also benefit some rules to simplify the understandability and readability of the codes. As a team we will develop the system together, but most of the time we will work on the code at different time slots. So, with the help of the CVS and a predefined specification rules will prevent us the get in conflicts and doing wrong things.

We have agreed on some coding conventions to benefit the syntax specification.

8.1 Naming the Classes and Files

All classes will have names beginning with a capital letter. The classes with more than one word will have a capital letter at the beginning of each word. For instance, “ConnectionHandler” is a suitable class name in NewsAgent.

For the files of the Java classes, Java has a restriction that the file name must be same as the class name inside. Every file can only include one class. But that class can contain multiple classes.

8.2 Naming Functions

Function names start with lower-case letters and continue until a new word starts. New word stars with capital letter and continues with lower-case letters. For example "checkLoginInfo()" is a suitable function name in NewsAgent.

8.3 Naming Variables

Variable names start with a letter indicating the scope of that variable.

- "m" --> attribute of a class. Indicating that member variable of a class.
- "v" --> parameter of a function. Indicating that scope of the variable is the function that it is passed.
- "l" --> local variable. Indicating that the variable is defined locally.

After the initial letter, variable name continues with a letter sequence indicating the type of it.

- "int" --> indicating that the variable is an integer variable.
- "float" --> indicating that the variable is a float variable.
- "double" --> indicating that the variable is a double variable.
- "str" --> indicating that the variable is a string variable.
- "obj" --> indicating that the variable is an object.

After these conventions are applied, the usual naming conventions mentioned above are applied to the variables. Suitable variable examples are as follows;

- "mstrUsername"
- "mintPortNo"
- "mobjConnectionHandler"

8.4 Comment Conventions

Commenting is also a critical issue to increase the understandability of the code. Since each java class is defined in separate files we have decided to have detailed information at the beginning of each file as described follows:

```
/* ************************************************************
/* File name:
/* Created by:
/* Created at: ( Date:DD.MM.YY – Time: HH:MM:SS)
/* Modified by:
/* Modified at: ( Date:DD.MM.YY – Time: HH:MM:SS)
/* Description:
************************************************************/
```

9 GANTT CHART

Gantt chart of NewsAgent is presented in APPENDIX.

10 CONCLUSION

In the content of this report, design of NewsAgent is specified in a detailed manner. Each part is described using different diagrams and discussions on them. These diagrams and discussions on different aspects of NewsAgent provide it to be handled by using different techniques which will be useful for observing different modules of NewsAgent from different point of views.

To sum up all design issues in this report precisely, it will be useful to have some discussion apart from diagrams presented before. For this reason, in this part, it will be reasonable to discuss each module from a generic point of view.

Web Service:

Web service is the core of NewsAgent, since all database accesses will be controlled by it. Web service methods are consistent with NNTP commands specified by RFC 977 standards and each command in this standard will be mapped to a web service method using a handler which is used to specify the meaning of NNTP command. Meaning of each NNTP command is determined by using a hash table which will also preserve time to find which web service method to be called. In the hash table, each NNTP command will be mapped to a Web service method, so after that point control of the application will be passed to web service of NewsAgent.

<u>RSS/Atom</u>:

RSS/Atom module will provide users to access articles without reading accessing newsgroups manually. By using an RSS reader, user will be able to access feeds of NewsAgent, to use RSS module of NewsAgent, user should subscribe to feeds of newsgroups that he/she wants to access. A feed tree will be used to handle RSS module. For instance, after an article is inserted into the database, feed of the newsgroup that the article is posted and feed tree will be updated, accordingly. After that time, RSS reader used by user will be able to access the updated feed. Of course, users will be able to access of the feeds of newsgroups corresponding to his/her access level.

<u>SMTP</u>:

SMTP module will provide users to receive mails from news server and send mails to newsgroups in a specified format. By using this functionality, users will be able to receive articles via e-mail by not being forced to log in to the system, then listing newsgroups and articles. SMTP server in NewsAgent will also provide NewsAgent to receive mails from clients and users will be able to post article to newsgroups via e-mail. Of course, users will again be able to post and receive articles from/to newsgroups corresponding to his/her access level.

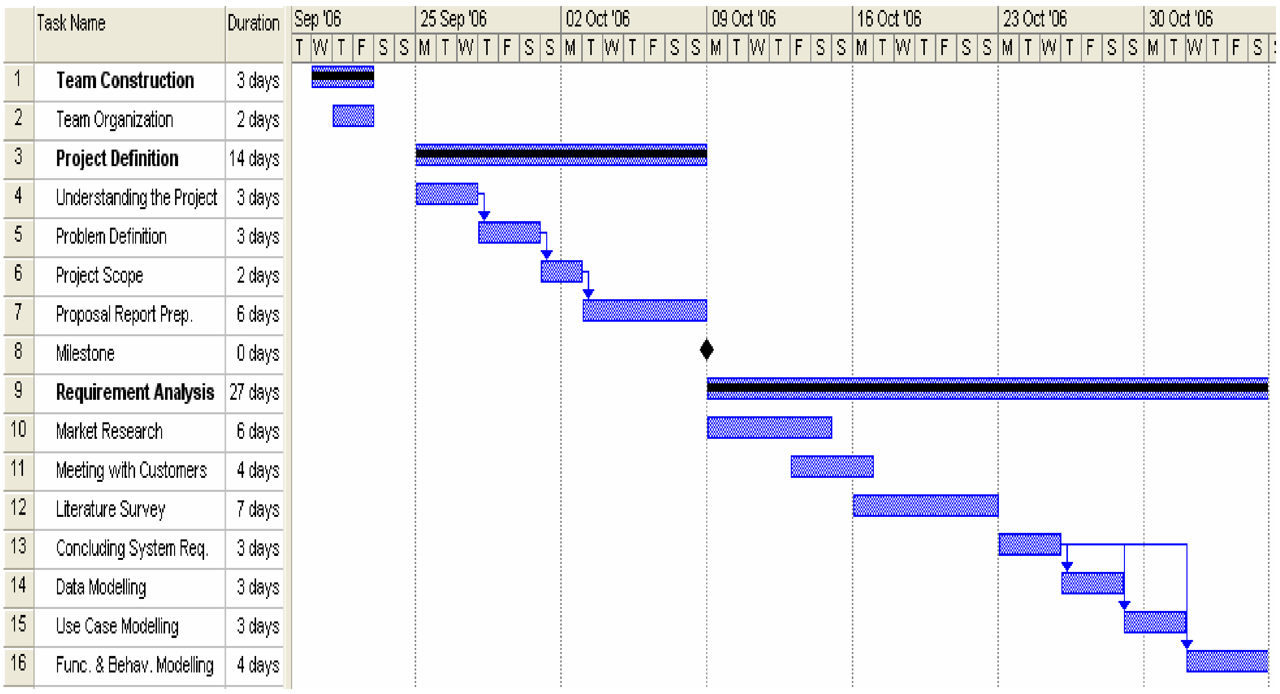
<u>SSL</u>: By using SSL features, NewsAgent will be a secure newsgroup environment and users will access newsgroups without considering any insecure applications. Authentication and SSL encryption will provide secure applications when user passwords and authentication is taken into account. Also newsgroups, articles and user information will not be accessible by any unauthorized user or a user who should not be able to access.

In conclusion, providing such conveniences to users will be so beneficial that users will be able to post articles to newsgroups by using different features of NewsAgent. By this way, even who will not directly connect to news server, will be able to post and read articles. In addition to these, security will be one of the key points that NewsAgent will provide users.

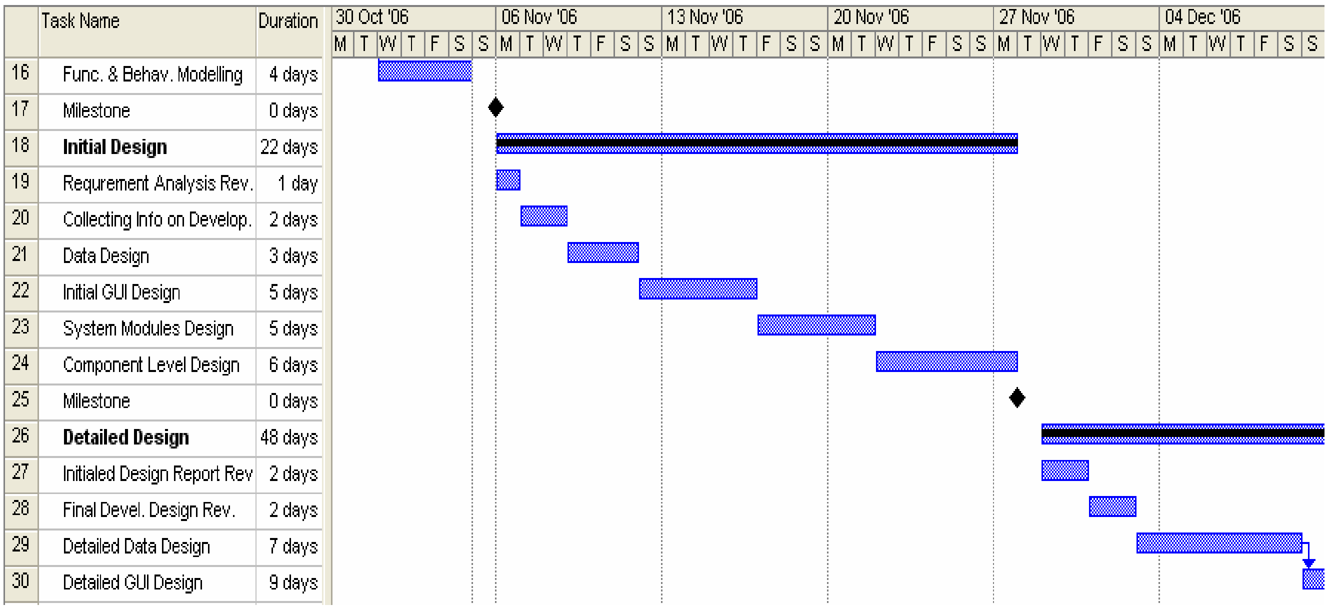
11 REFERENCES

1. http://www.tcpipguide.com
2. http://en.wikipedia.org/wiki/MD5
3. http://www.ietf.org/rfc/rfc0850.txt
4. www.ietf.org/rfc/rfc977.txt

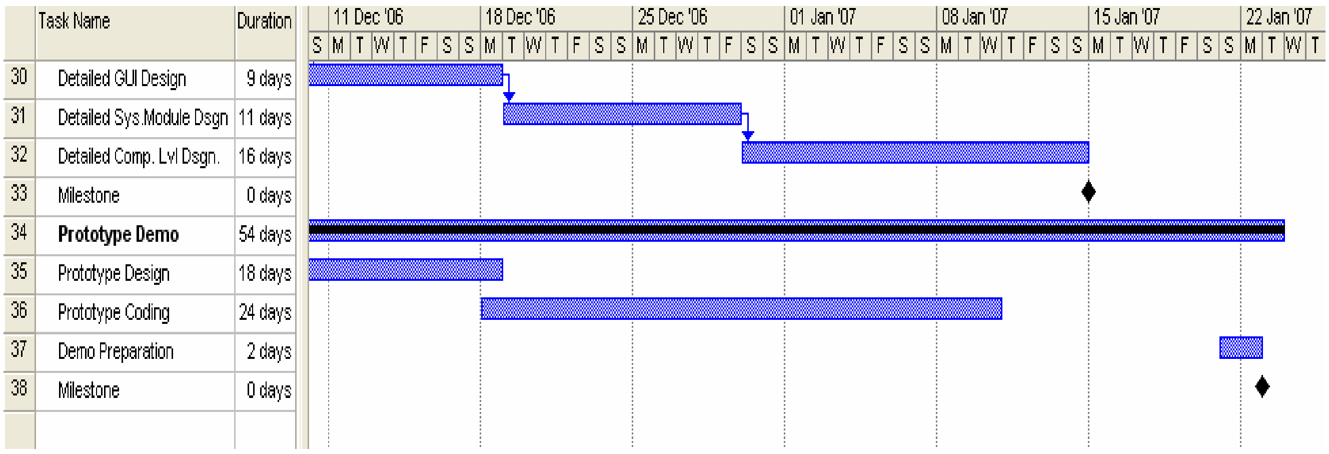
12 APPENDIX



	Task Name	Duration
1	**Team Construction**	3 days
2	Team Organization	2 days
3	**Project Definition**	14 days
4	Understanding the Project	3 days
5	Problem Definition	3 days
6	Project Scope	2 days
7	Proposal Report Prep.	6 days
8	Milestone	0 days
9	**Requirement Analysis**	27 days
10	Market Research	6 days
11	Meeting with Customers	4 days
12	Literature Survey	7 days
13	Concluding System Req.	3 days
14	Data Modelling	3 days
15	Use Case Modelling	3 days
16	Func. & Behav. Modelling	4 days



	Task Name	Duration
16	Func. & Behav. Modelling	4 days
17	Milestone	0 days
18	**Initial Design**	22 days
19	Requrement Analysis Rev.	1 day
20	Collecting Info on Develop.	2 days
21	Data Design	3 days
22	Initial GUI Design	5 days
23	System Modules Design	5 days
24	Component Level Design	6 days
25	Milestone	0 days
26	**Detailed Design**	48 days
27	Initialed Design Report Rev	2 days
28	Final Devel. Design Rev.	2 days
29	Detailed Data Design	7 days
30	Detailed GUI Design	9 days



	Task Name	Duration
30	Detailed GUI Design	9 days
31	Detailed Sys.Module Dsgn	11 days
32	Detailed Comp. Lvl Dsgn.	16 days
33	Milestone	0 days
34	**Prototype Demo**	54 days
35	Prototype Design	18 days
36	Prototype Coding	24 days
37	Demo Preparation	2 days
38	Milestone	0 days