Middle East Technical University

Department of Computer Engineering

Initial Design Report

by

1 Introduction

1.1 Purpose of This Document

The main purpose of this document is to initiate the design specifications of our project: GÜVERCİN. By this document, we intend to declare our solution for fulfilling the problem requirements. Throughout the report, we will use UML diagrams like use case, class, sequence diagrams in order to explain our approach to the functional requirements of the system

1.2 Design Considerations and Constraints

While designing the solution the team members have considered and emphasized these properties:

- The software must meet the functional requirements.
- The software should be such designed that it will be used by users with different levels-of knowledge and skills easily. It will provide a clear, easy to learn interface to the user.
- The software must be secure.
- The software should be made up of independent components with well-defined interfaces.

Time and the lack of experience of the team members are the two constraints for

the team. The project team has decided to make use of open-source free software components as much as possible. The reasons for these are:

- There exists open-source extendible software which can be used as components and integrated to the system.
- This software will sure be more reliable than a one implemented by the team members in a limited time. This software is reliable, continuosly used and developed by lots of people.
- It is very difficult for the team to implement an SMTP server or such components from scratch.

2 System Modules

Since NNTP and SMTP modules will be used as built-in servers, we do not need to explain them among our main modules. The 7 basic modules of the GÜVERCİN is explained as follows:

2.1 Login Module

Login is one of the most important modules of GÜVERCİN with Core Module and Administrator Module. The main function of this module is to authenticate the user for the whole system and creating a session for the user between the core module and both of SMTP and NNTP Modules. This module considers the HTTP users as they are NNTP users by converting the HTTP requests to NNTP requests.

2.2 HTTP Module

There are several duties of this module and they can be generalized as follows:

- Converting HTTP requests to NNTP requests
- Communicating with core for delivering users' and admin's NNTP requests
- Managing the RSS feed data storage
- Involving the configuration modules

2.3 User Configuration Module

The main function of this module is serving a user interface for the subscribed users for adjusting their membership preferences and configuring their mailing list options. The module is stored in HTTP Module and has a direct connection with our built-in mail server.

2.4 Core Module

Core Module is the most important module for the news and mail clients since it acts as a layer between the user and both of mail and news servers. Main function of this module is delegating the users' requests on behalf of the them and delivering the responses coming from these servers to the each mail and news clients. The module also plays a crucial role for the login process since the real session is created between the core and servers on behalf of the clients.

2.5 *Administrator Module*

Administrator module is the user interface for the admins for configuring the whole system. Admin may manage the newsgroups, users and configure the properties of the system by using this module. The module communicates with the main database by using the methods for doing user operations and some newsgroup properties. It also connects to the built-in news and mail servers by means of the core module for the basic news and mail group operations like adding, editing and deleting.

2.6 *DBLayer Module*

DBLayer module constitutes an interface for our database. In the analysis phase we choose MySQL as our DBMS but as the users of our product may grow, MySQL can not cope with the problem of traffic density. In order to solve this problem we decided to use a DBLayer module. The module basicly serves to the other modules for an easy connection to the database. By this way, efficiency and product independency of GÜVERCİN in terms of database systems is provided.

2.7 *RSS Module*

The main function of the RSS module is serving the RSS read and write comment requests. The module can be defined as a submodule of the HTTP module and it is stored in our HTTP server. Since the articles and mails that NNTP and SMTP clients sent may also be delivered as RSS feeds, the module is also communicates with the Core module.

3 Data Design

We have reviewed the entity relationship diagram drawn in our analysis report and altered our database design. In this section we will look at the revised data objects, entity relationship diagram, data dictionary and SQL queries of the data objects.

It is unnecessary to store the articles in our own database since the articles are stored in the transit servers of the built-in servers. Besides there is no need to store subscribed users' mail addresses since mailing list databases already store them.

The below attributes [1] are stored in the transit servers:

- *Message-ID* - a globally unique key
- *Newsgroups* - a list of one or more newsgroups where the article is intended to appear
- *Distribution* - (optional) a supplement to Newsgroups, used to restrict circulation of articles.
- *Date* - the time when the article was created
- *From*- the sender of the article
- *Subject*- the subject of the article

- *Path* - a list of the servers an article passed through on its way to the local server
- *Expires* - (optional) the time when it is requested that the article be deleted
- *Approved* - (optional) indicates an article that has been accepted for a moderated newsgroup

3.1 Data Objects

3.1.1 User

User entity stores data associated with the user. The core will be establishing a session between the NNTP Server and Mail Server on behalf of the user whom will be delegated. Therefore, we will have to store the user's information.

- user_id
- username
- common_surname
- common_name
- user_type_id

3.1.2 Password

Password entity stores the SMTP and NNTP passwords of the user. These passwords will allow the core to establish the secure connection and will be entered when the user logs into the system. Since the user may not prefer to choose the same NNTP and SMTP passwords, the two of them are both stored in the database. The user's password is converted to the MD5 hash to be secure.

- password

- SMTP_passwd
- NNTP_passwd

3.1.3 Admin

Admin entity is just a type of user, used to identify the administrators. Admin entity will store the user_id's of the privileged users. The users in this entity will have the administrative rights and they will be able to configure the system with their passwords.

- user_id

3.1.4 Banned User

BannedUser entity is again a type of user who has restricted rights, forbidding the user from sending messages to the system either by the NNTP or SMTP.

- user_id
- newsgroup_id
- ban_id
- ban_start_date

3.1.5 AllowedUser

AllowedUser entity is also a type of user who has additional rights from a naive user. AllowedUser entity is added to system in case of the restricted newsgroups, i.e. an administrator announcement newsgroup may allow the only the administrator to send messages and forbid the naive users sending messages. In this case, instead of banning all of the other users for the administrator

announcement newsgroup, the administrator's user id and the newsgroup's id will be added to the AllowedUser entity.

- user_id
- newsgroup_id

3.1.6 CoreConfiguration

CoreConfiguration entity will help us while making the configurations of the core. Since our project is based on making a unified news exchange server, we will use these ports. Admin will be able to configure whole system by using these entities.

- smtp_port
- nntp_port
- http_port

3.1.7 RSSFeedTable

RSSFeedTable will have a key attribute id that the system will give. The XML files will be stored in the directories, according to the RSS_id's of each newsgroup – mailing list. The RSS_path is associated with exactly one newsgroup.

- RSS_id
- directory_name
- RSS_path

3.1.8 BanPeriods

BanPeriods entity stores the ban period of a certain type of ban. The users are

banned according to rules defined previously and the ban periods are not variable; the ban period of a misuse is always constant. The administrator will have the capability of adding a new ban, removing or modifying an existing one. The system assigns each a unique id, which will be used as a primary key.

- ban_id
- ban_type
- ban_period

3.1.9 UserTypes

Although the main types are only administrator and logged user; in order to be able to add new features to the system, the administrator will be capable of adding new types of users to the system such as a student, instructor or assistant for a course newsgroup.

- user_type_id
- user_type

3.1.10 Newsgroup

Newsgroup entity assigns a unique id to each newsgroup identified with their addresses. The flag attribute of this entity will show us whether the newsgroup is moderated or not.

- newsgroup_id
- newsgroup_address
- flag

3.1.11 MailingList

Since there is no need for another attribute related to the mailing lists, MailingList entity will store the mailing list addresses.

- mailing_list_address

3.1.12 News2Mail

In Güvercin, there will be exactly one mailing list for each newsgroup and there may exist an RSS path for each of the newsgroup- mailing list pair. News2Mail relationship will associate newsgroup_id's with the mailing list addresses and store the RSS_id referencing the RSSFeedTable if there exists an RSS feed.

- newsgroup_id
- mailing_list_address
- RSS_id

3.2 Entity-Relationship Diagrams

3.2.1 Data Objects

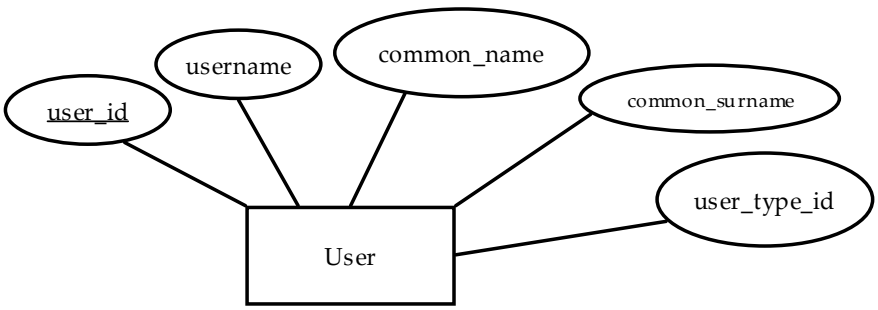


Figure 1: User Entity

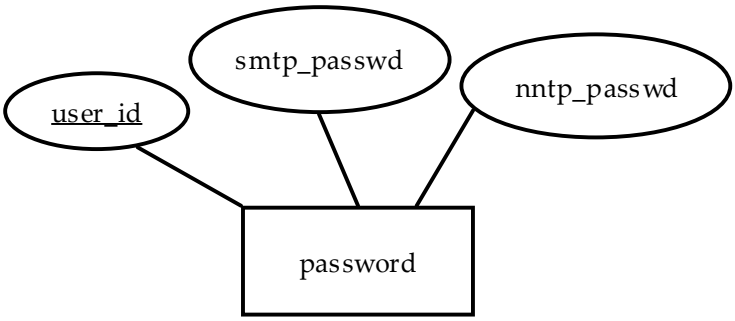


Figure 2: Password Entity

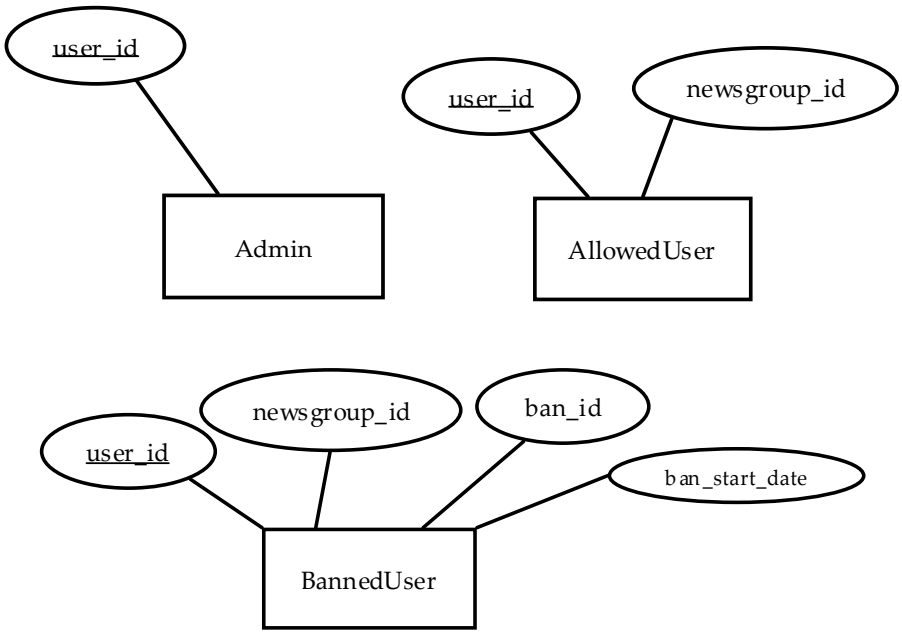


Figure 3: Users

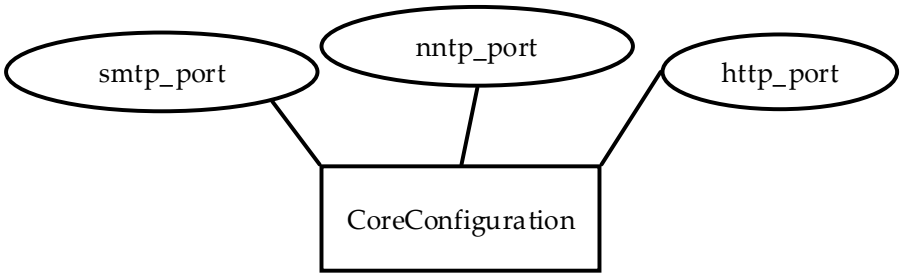


Figure 4: CoreConfiguration Entity

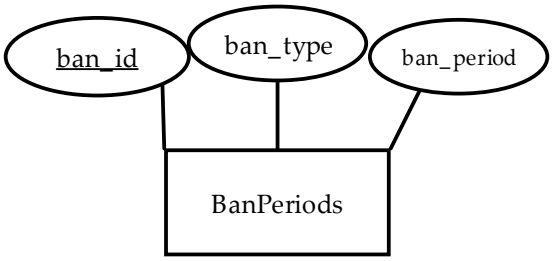


Figure 5: BanPeriods Entity

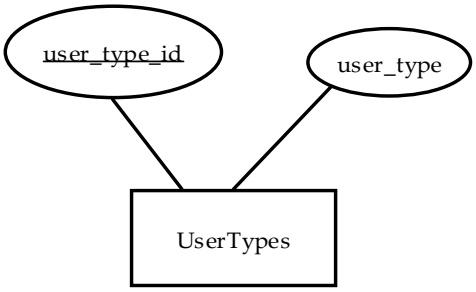


Figure 6: UserTypes Entity

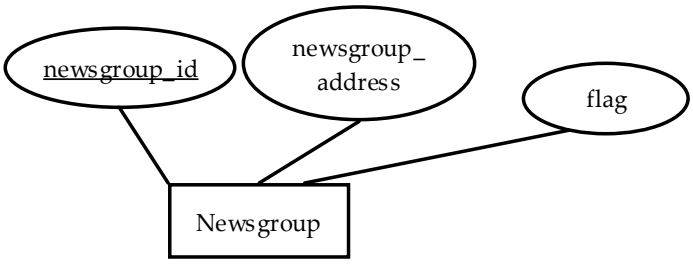
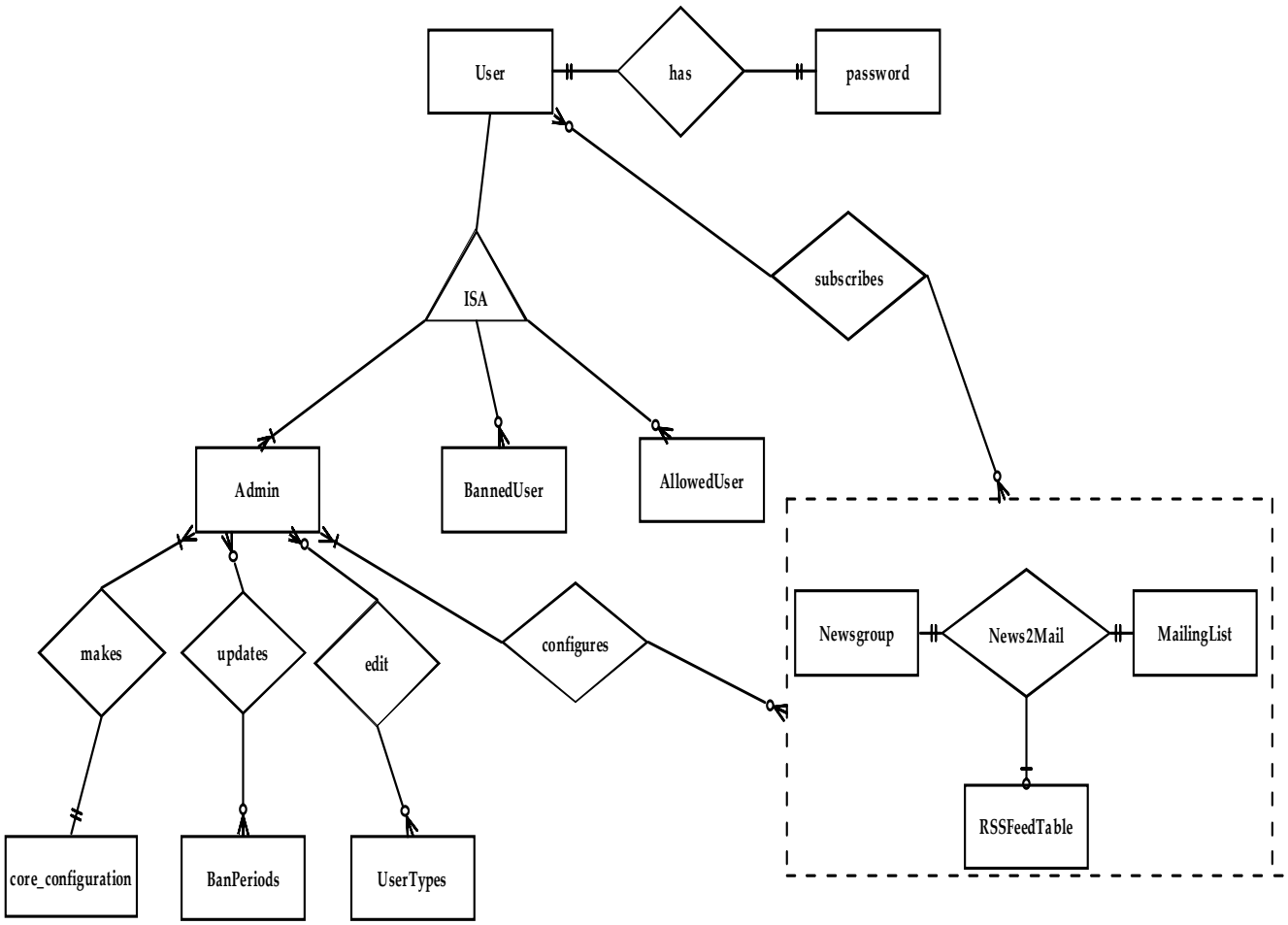


Figure 7: Newsgroup Entity

3.2.2 Entity-Relationship Diagram



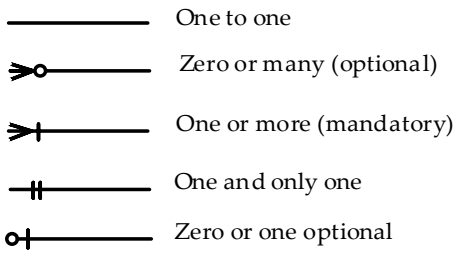


Figure 8: ER Diagram

3.3 Data Dictionary

3.3.1 User

User		
Attribute Name	**Attribute Type**	**Description**
user_id	INT (15)	This is the unique id assigned automatically to each user who has filled out the subscription page.
username	VARCHAR (15)	This is the username chosen by the user for the main displays the screen name, composed of at least 6 and at most 15 characters. In addition to the user_id's usernames will also be unique.
common_surname	VARCHAR (20)	This is the real surname of the user for displaying the names in the mails, newsgroups on the outgoing messages.
common_name	VARCHAR (25)	These are the other names of the user beyond his/her surname used for the same purposes with the common_surname.

user_type_id	VARCHAR(2)	This shows the type of the user referring to the user types defined in UserTypes entity

3.3.2 Password

Password		
Attribute Name	**Attribute Type**	**Description**
user_id	INT (15)	This is the unique id referring to the user_id in the user table.
SMTP_passwd	VARCHAR (32)	This is the password chosen by the user to access to the SMTP module of system.
NNTP_passwd	VARCHAR (32)	This is the password chosen by the user to access to the NNTP module of system.

3.3.3 BannedUser

BannedUser		
Attribute Name	**Attribute Type**	**Description**
user_id	INT (15)	This is the unique id referencing one of the user_id's in the user table.
newsgroup_id	INT (15)	This is the unique id referencing one of the newsgroup_id's in the newsgroup table.

ban_id	INT (15)	This is the unique id referencing one of the ban_id's in the BanPeriods table. This will giv e the ban peroid and help us to set the end date of ban.
ban_start_date	date	This is the current date and will be set by the system automatically.

3.3.4 AllowedUser

AllowedUser		
Attribute Name	**Attribute Type**	**Description**
user_id	INT (15)	This is the unique id referencing one of the user_id's in the user table.
newsgroup_id	INT (15)	This is the unique id referencing one of the newsgoup_id's in the newsgroup table.

3.3.5 CoreConfiguration

CoreConfiguration		
Attribute Name	**Attribute Type**	**Description**
smtp_port	INT(5)	This controls the port for which incoming mail connections.
nntp_port	INT(5)	This is the IP port to connect NNTP server.
http_port	INT(5)	The port number that HTTP server uses.

3.3.6 RSSFeedTable

RSSFeedTable		
Attribute Name	**Attribute Type**	**Description**
RSS_id	INT (15)	This is the unique RSS id assigned to each RSS path.
directory_name	VARCHAR(50)	The directory name that XML files are stored in.
RSS_path	VARCHAR(50)	The URL of the RSS of each newsgroup which have RSS feeds.

3.3.7 BanPeriods

BanPeriods		
Attribute Name	**Attribute Type**	**Description**
ban_id	INT (15)	The unique id numbers that the system will assign to each new type of ban.
ban_type	VARCHAR(20)	The ban types will be entered to the system with the corresponding ban periods by the administrator, e.g. newsgroup misuse 5 days.
ban_period	INT(2)	The ban periods stored in the database can be updated afterwards.

3.3.8 UserTypes

UserTypes		
Attribute Name	**Attribute Type**	**Description**
user_type_id	INT(15)	The primary key of each type of user assigned automatically by the system.

user_type	VARCHAR(20)	The predefined user types will be only administrator and naive user, after that the administrator will have the capability to define new types of users.

3.3.9 Newsgroup

Newsgroup		
Attribute Name	**Attribute Type**	**Description**
newsgroup_id	INT (15)	Each newsgroup will have an id in order to facilitate the connections in the relationships.
newsgroup_address	VARCHAR(30)	The address of the newsgroup.
flag	TINYINT(1)	The flag will store 1 for moderated groups and 0 otherwise.

3.3.10 MailingList

MailingList		
Attribute Name	**Attribute Type**	**Description**
mailing_list_address	VARCHAR(50)	The address of the mailing lists created.

3.3.11 News2Mail

News2Mail		
Attribute Name	**Attribute Type**	**Description**
newsgroup_id	INT (15)	The unique id of the newsgroup referencing the newsgroup_id attribute in newsgroup entity.
mailing_list_address	VARCHAR(50)	The address of the mailing list that will be associated to the newsgroup.
RSS_id	INT (15)	Relates each newsgroup with the RSS feed. However, each newsgroup may not give RSS feed, therefore this attribute may be null.

3.4 Database Tables' SQL Queries

```
create table User(
        user_id int(15) not null auto_increment,
        username varchar(15) not null unique,
        common_name varchar(20),
        common_surname varchar(25),
        user_type_id varchar(2) not null,
        constraint user_pk primary key(user_id)
);


create table Password(
        user_id int(15) not null auto_increment,
        SMTP_passwd varchar(32),
        NNTP_passwd varchar(32),
        constraint password_fk foreign key(user_id) references
                User(user_id)
                on delete cascade
);

create table BannedUser(
```

```
	user_id int(15) not null auto_increment,
	newsgroup_id int(15) not null auto_increment,
	ban_id int(15) not null auto_increment,
	ban_start_date date default CURRENT_DATE,
constraint banneduser_user_fk foreign key(user_id)
references User(user_id)
on delete cascade,
	constraint banneduser_ng_fk foreign key(newsgroup_id)
		references Newsgroup(newsgroup_id)
on delete cascade,
	constraint banneduser_ban_fk foreign key(ban_id)
		references BanPeriods(ban_id)
		on delete cascade
);



create table AllowedUser(
	user_id int(15) not null auto_increment,
	newsgroup_id int(15) not null auto_increment,
constraint alloweduser_user_fk foreign key(user_id)
references User(user_id)
on delete cascade,
	constraint banneduser_ng_fk foreign key(newsgroup_id)
		references Newsgroup(newsgroup_id)
		on delete cascade
);


create table CoreConfiguration(
	smtp_port int(5) default 0,
	http_port int(5) default 0,
	nntp_port int(5) default 0
);

create table RSSFeedTable(
	rss_id int(15) not null auto_increment,
	directory_name varchar(50) not null unique,
	RSS_path varchar(50) not null unique
);
```

```
create table BanPeriods(
        ban_id int(15) not null auto_increment,
        ban_type varchar(25) not null unique,
        ban period int(2)
        constraint banperiods_pk primary key(ban_id)
);

create table UserTypes(
        user_type_id int(15) not null auto_increment,
        user_type varchar(20) not null unique,
        constraint usertypes_pk primary key(user_type_id)
);




create table Newsgroup(
        newsgroup_id int(15) not null auto_increment,
        newsgroup_address varchar(30) not null,
        flag tinyint(1) default 0,
        constraint newsgroup_pk primary key (newsgroup_id)
);

create table MailingList(
        mailing_list_address varchar(50) not null unique
);

create table News2Mail(
        news_id int(15) not null,
        mail_list_add varchar(50) not null,
        rss_id int(15),
constraint news2mail_news_fk foreign key(news_id)
references Newsgroup(newsgroup_id)
on delete cascade,
constraint news2_mail_fk foreign key(mail_list_add)
references MailingList(mailing_list_address)
on delete cascade,
constraint news2mail_rss_fk foreign key(rss_id)
references RSSFeedTable(rss_id)
```

```
on delete cascade,
);
```

4 System Design

4.1 Use Case Diagrams

4.1.1 User

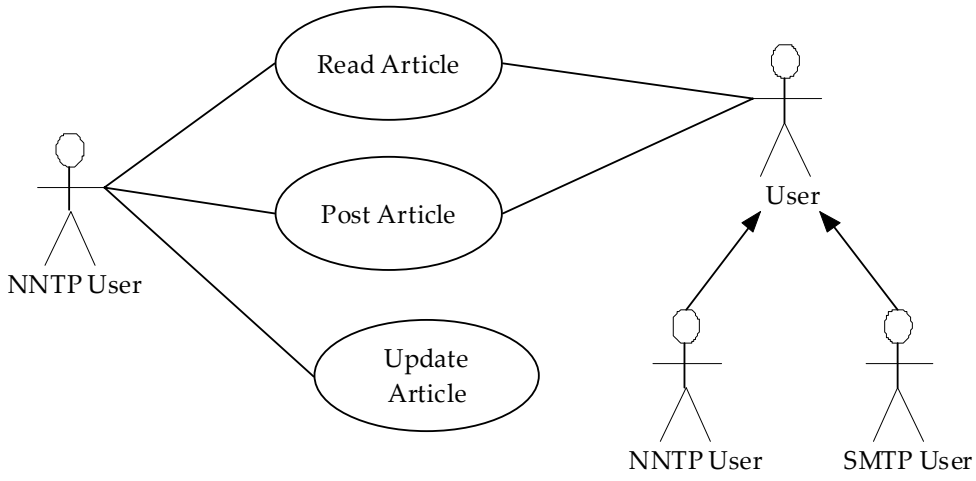


Figure 9: NNTP User

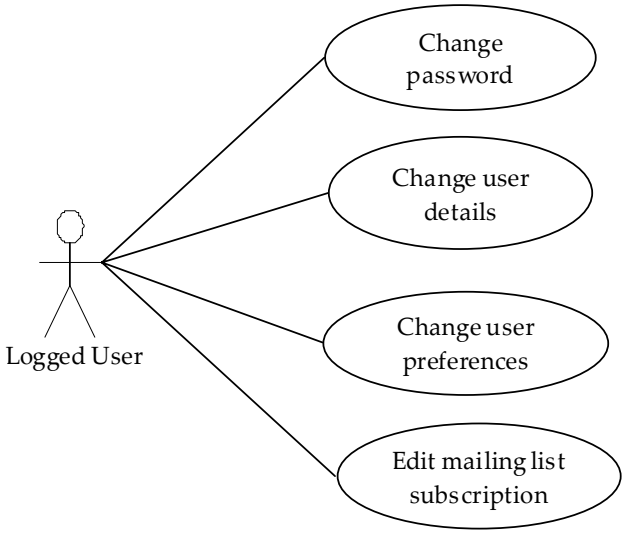


Figure 10: Logged User

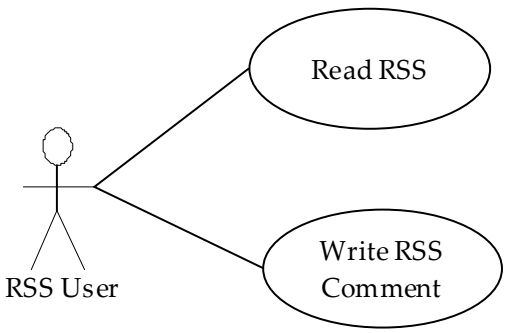


Figure 11: RSS User

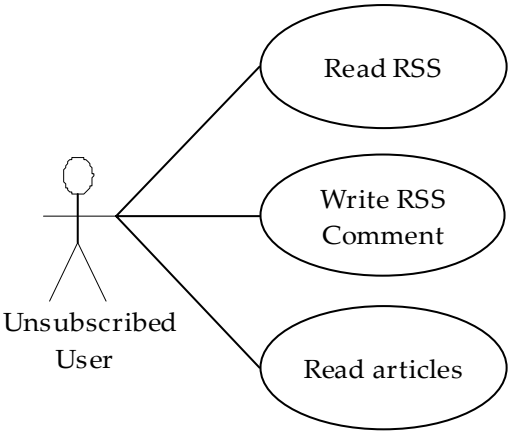


Figure 12: Unsubscribed User

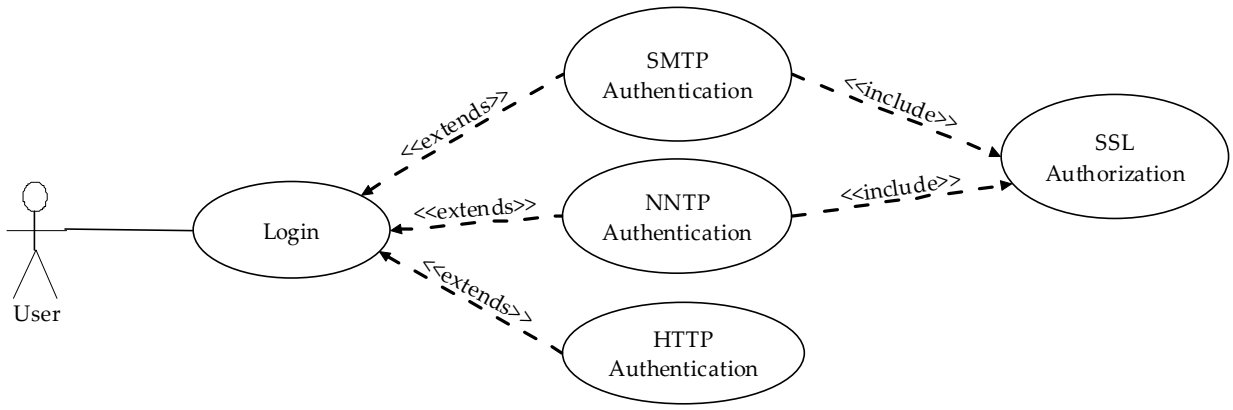


Figure 13: User

4.1.2 Admin

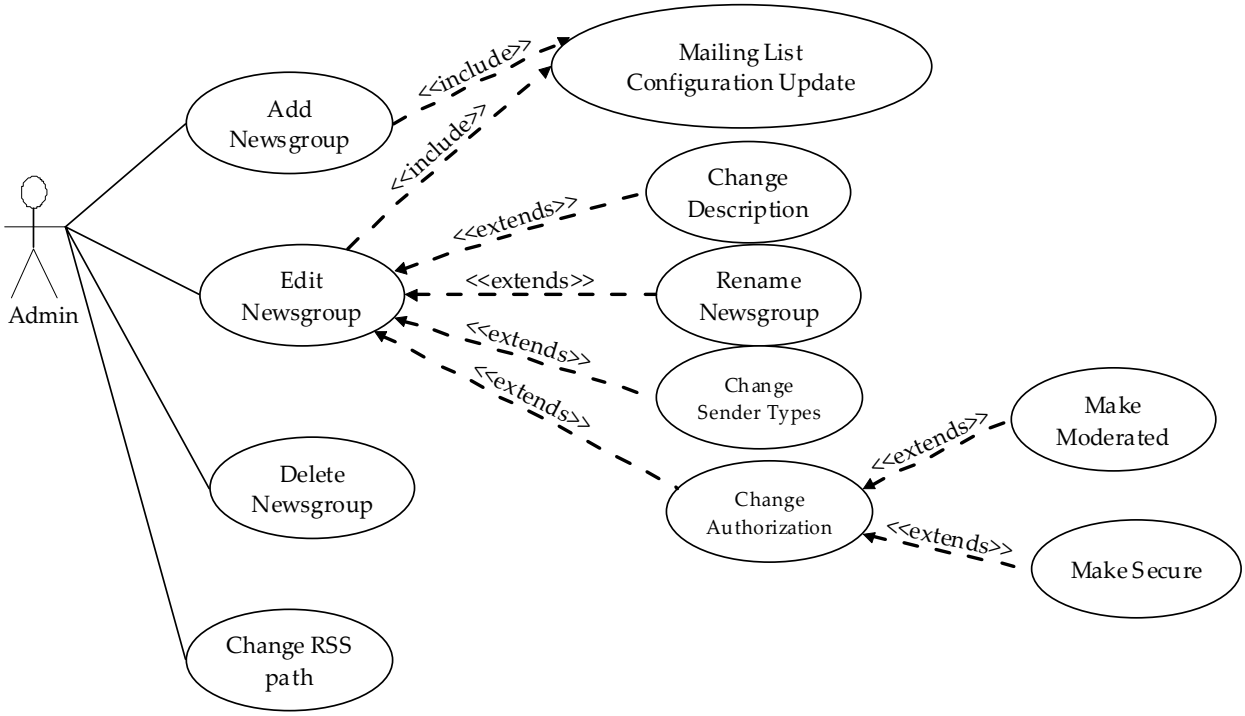


Figure 14: Newsgroup Configuration

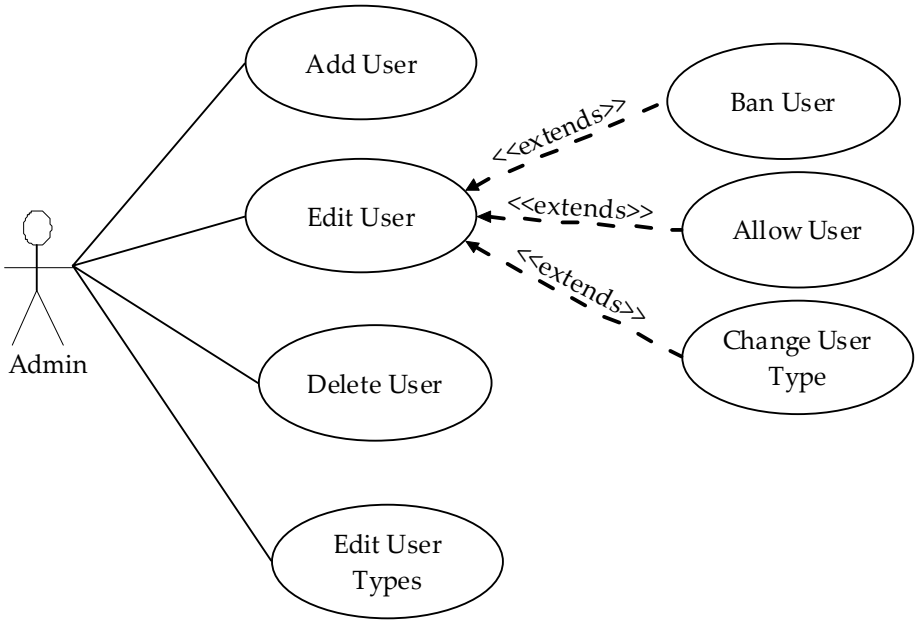


Figure 15: Users Configuration

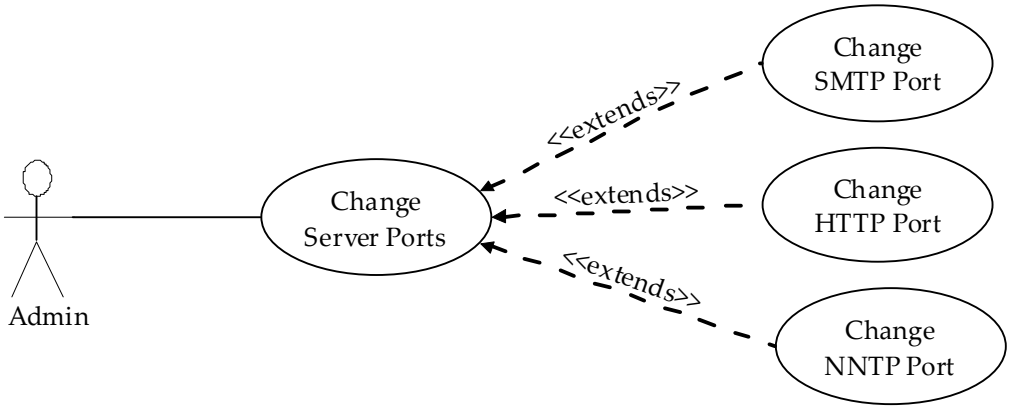


Figure 16: System Configuration

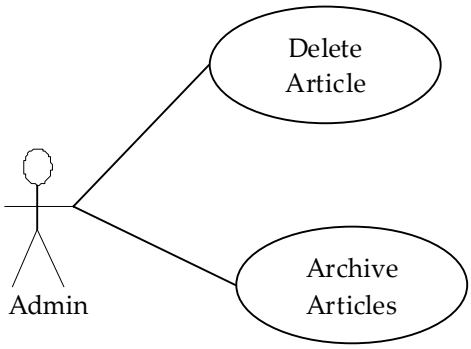


Figure 17: Article Management

4.1.3 Core

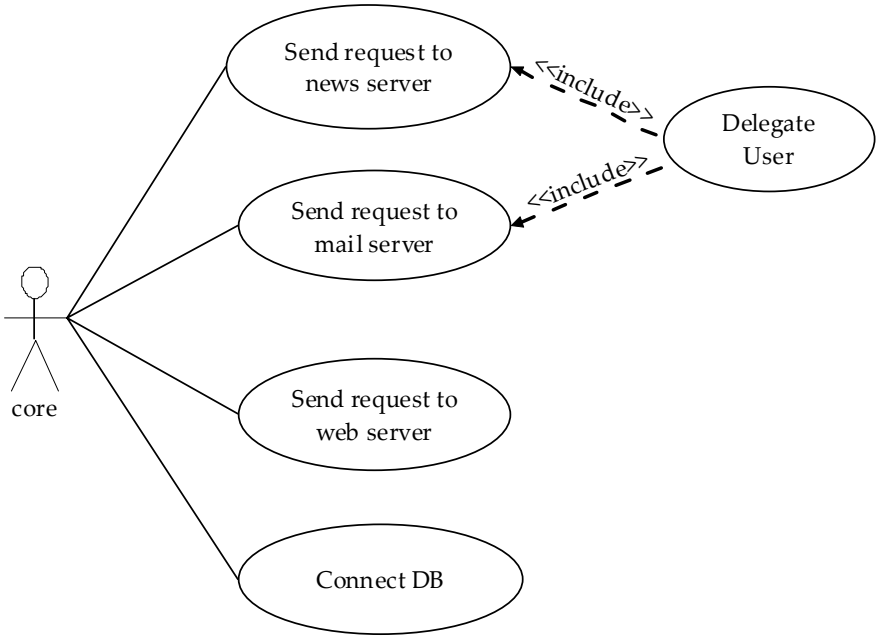
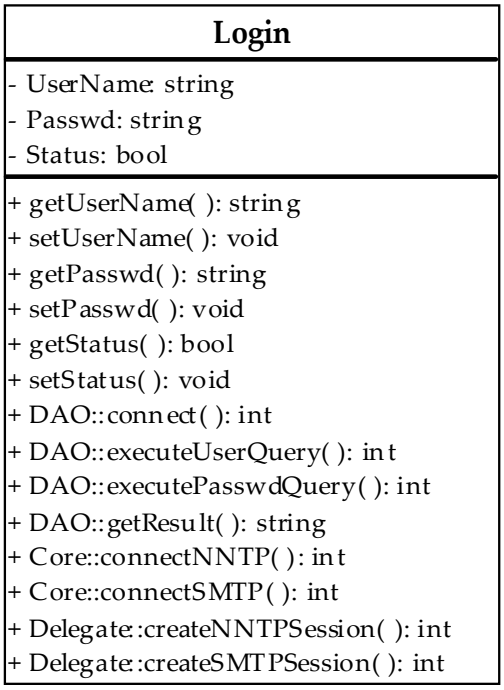


Figure 18: Core

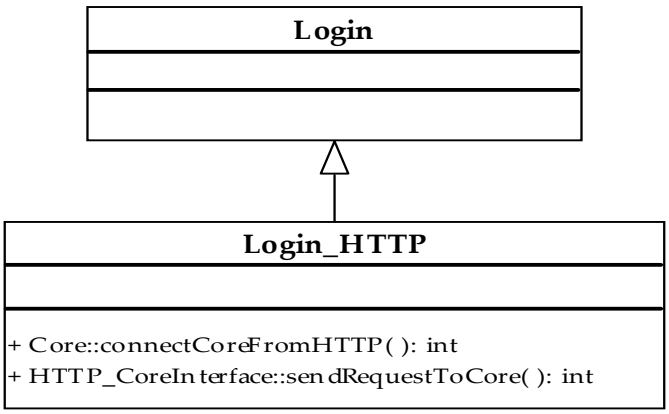
4.2 Class Diagrams

4.2.1 Login Module

4.2.1.1 Login

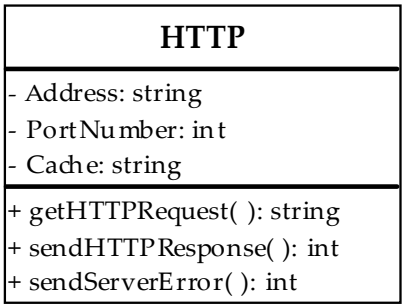


4.2.1.2 Login_HTTP

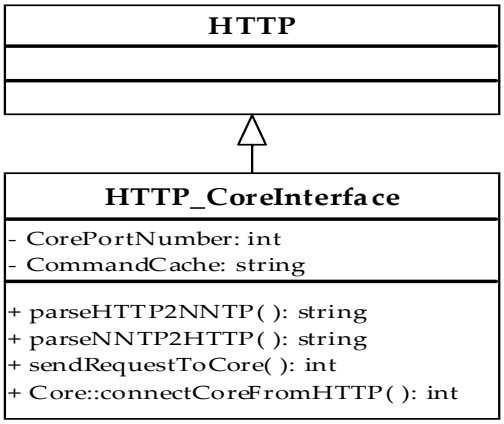


4.2.2 HTTP Module

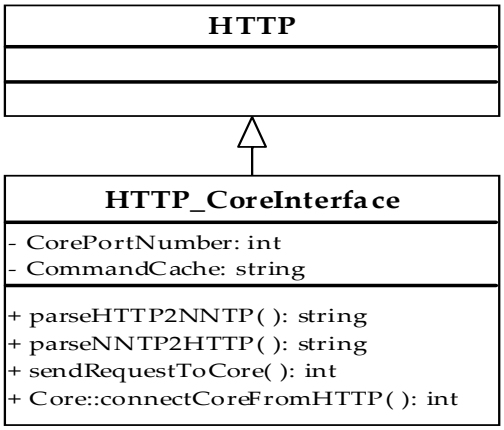
4.2.2.1 HTTP



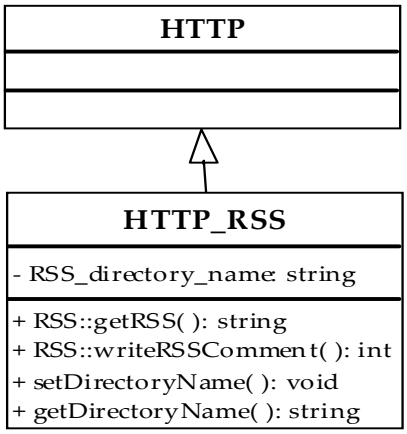
4.2.2.2 HTTP_CoreInterface



4.2.2.3 HTTP_DB

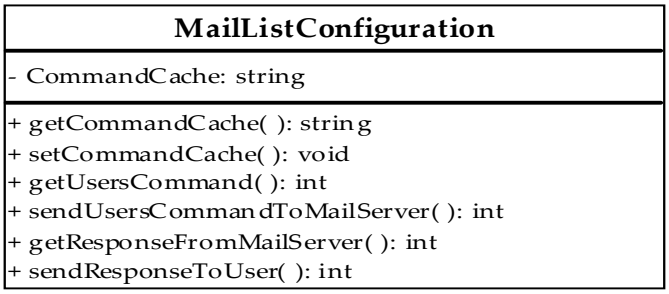


4.2.2.4 HTTP_RSS

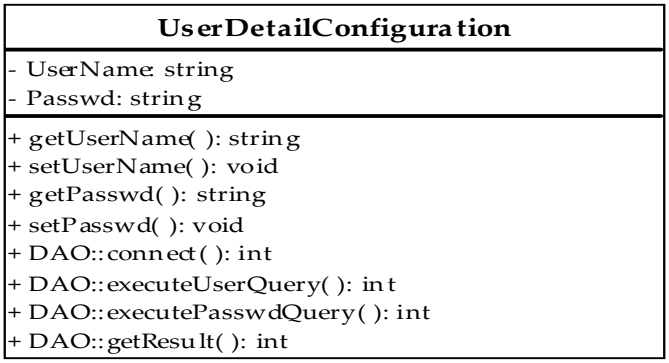


4.2.3 User Configuration Module

4.2.3.1 MailListConfiguration



4.2.3.2 UserDetailConfiguration

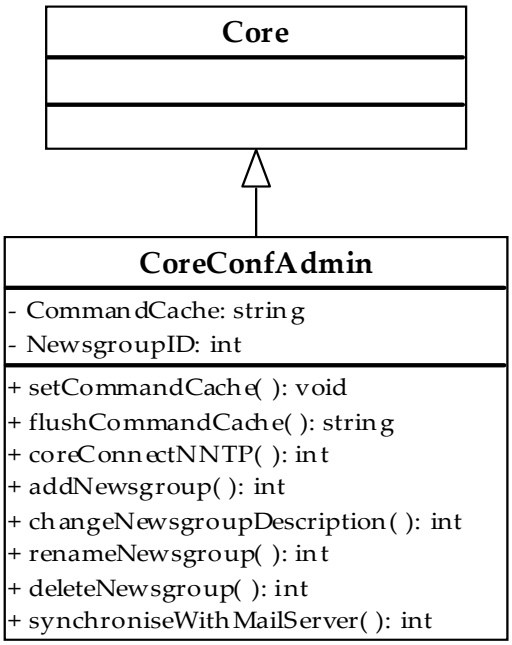


4.2.4 Core Module

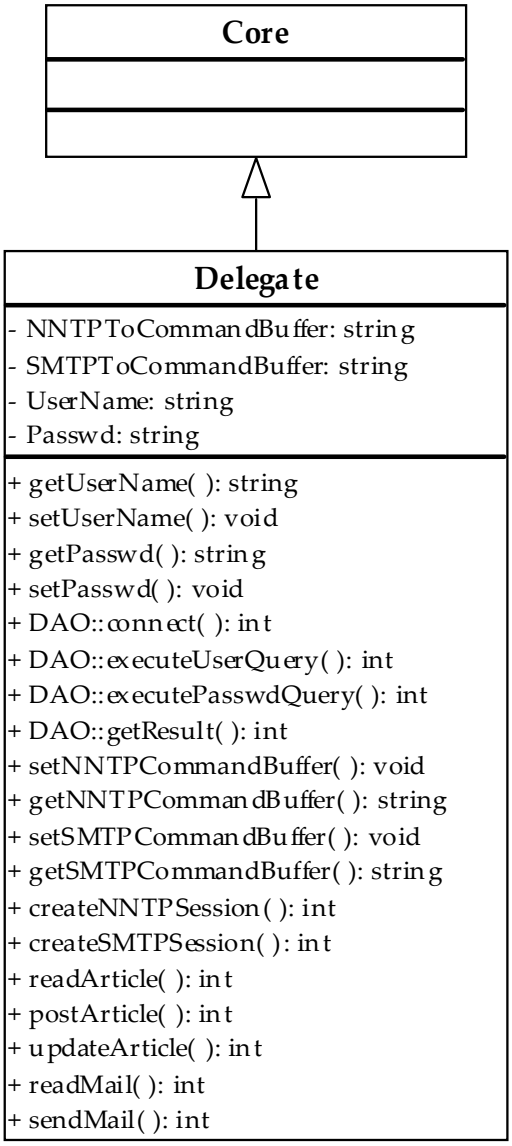
4.2.4.1 Core

Core
- NNTPAddress: string - SMTPAddress: string - PortNumber: int
+ ConnectCoreFromHTTP( ): int + ConnectHTTP( ): int + ConnectSMTP( ): int + getNNTPAddress( ): string + getSMTPAddress( ): string + getPortNumber( ): int + NNTP2SMTP( ): int + SMTP2NNTP( ): int + sendCommandToNNTP( ): int + sendCommandToSMTP( ): int + sendResponseToHTTP( ): int + writeToRSS( ): int + getNNTPRequest( ): int + sendNNTPResponse( ): int

4.2.4.2 CoreConfAdmin

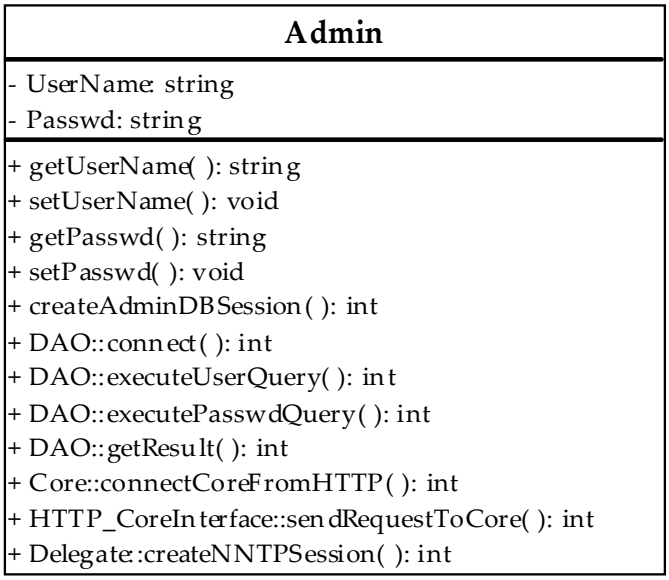


4.2.4.3 Delegate

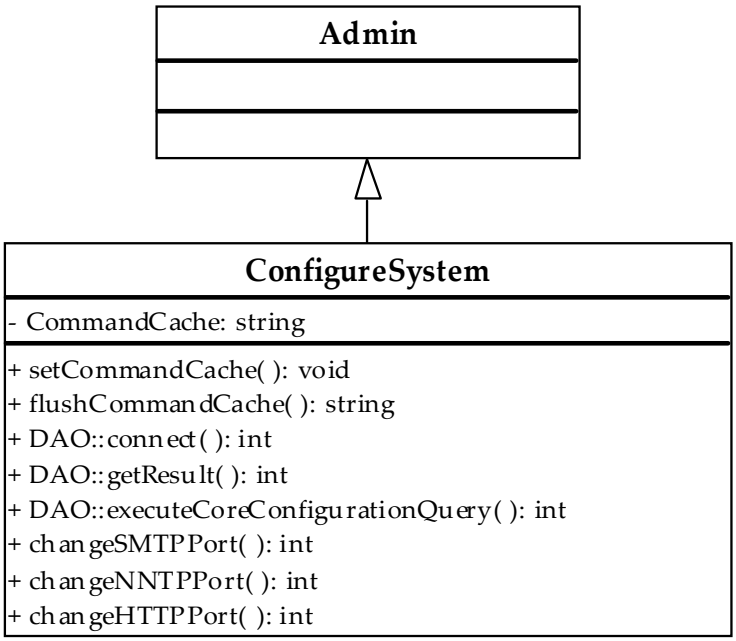


4.2.5 Admin Module

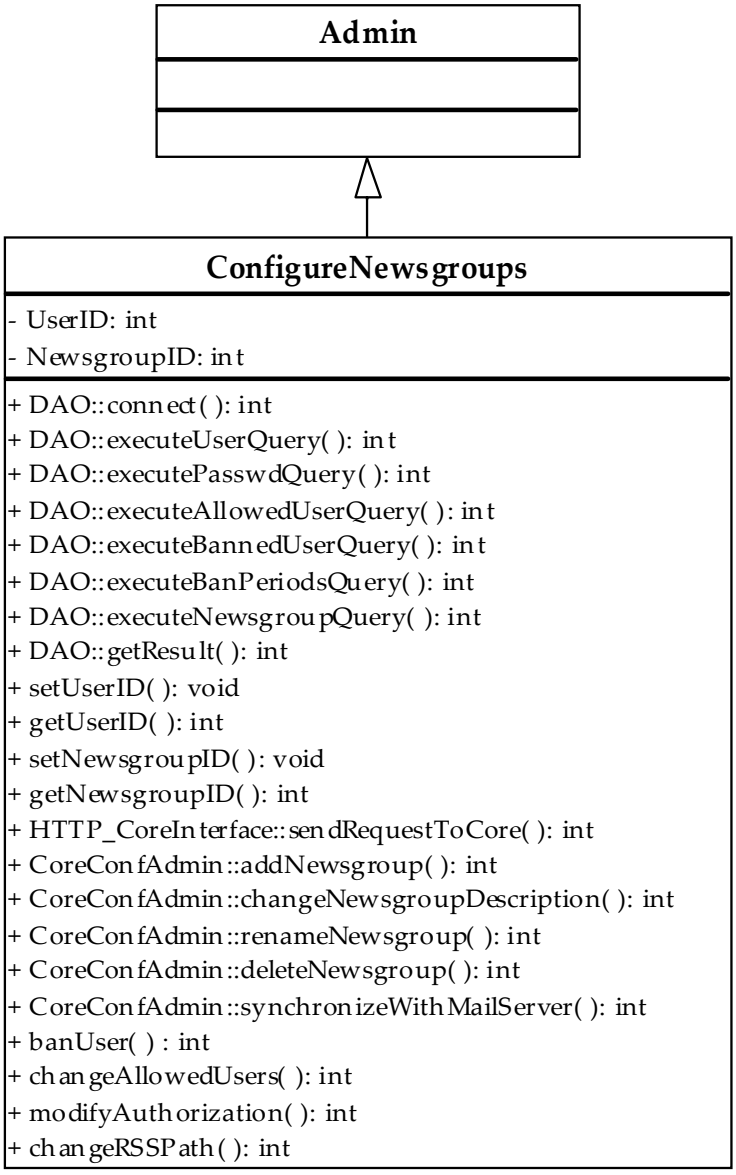
4.2.5.1 Admin



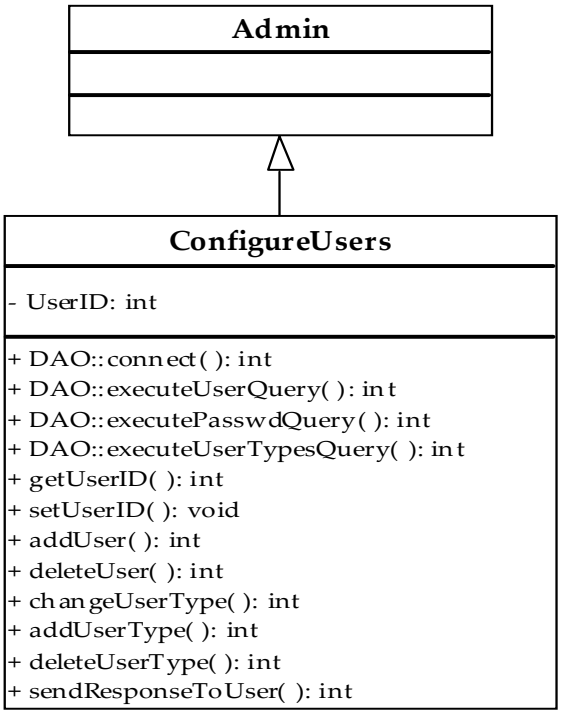
4.2.5.2 ConfigureSystem



4.2.5.3 ConfigureNewsgroups

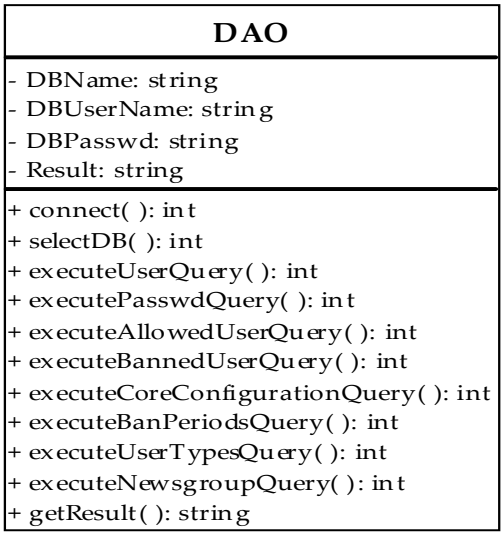


4.2.5.4 ConfigureUsers



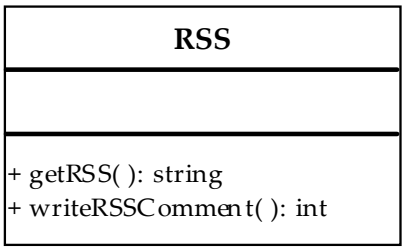
4.2.6 DBLayer Module

4.2.6.1 DAO



4.2.7 RSS Module

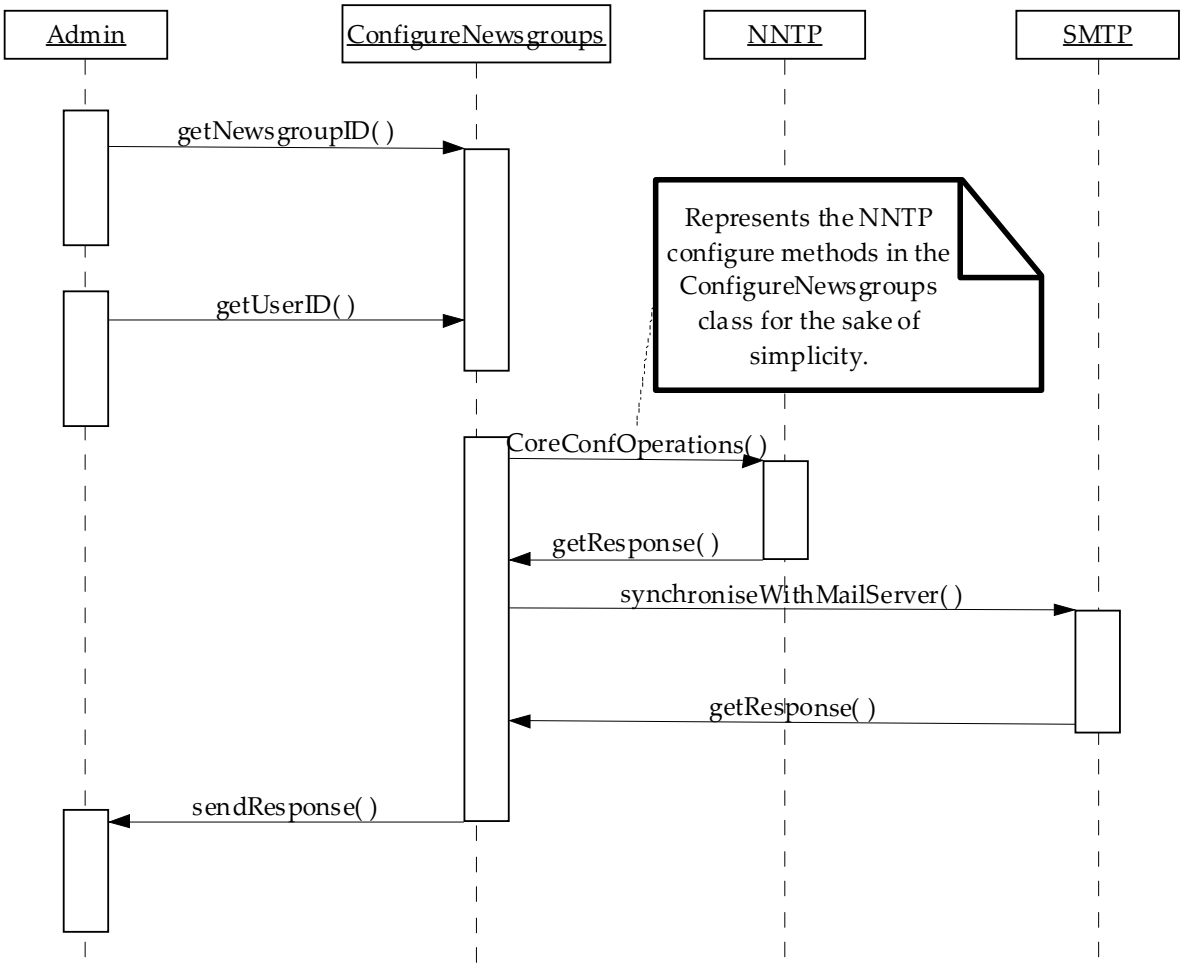
4.2.7.1 RSS



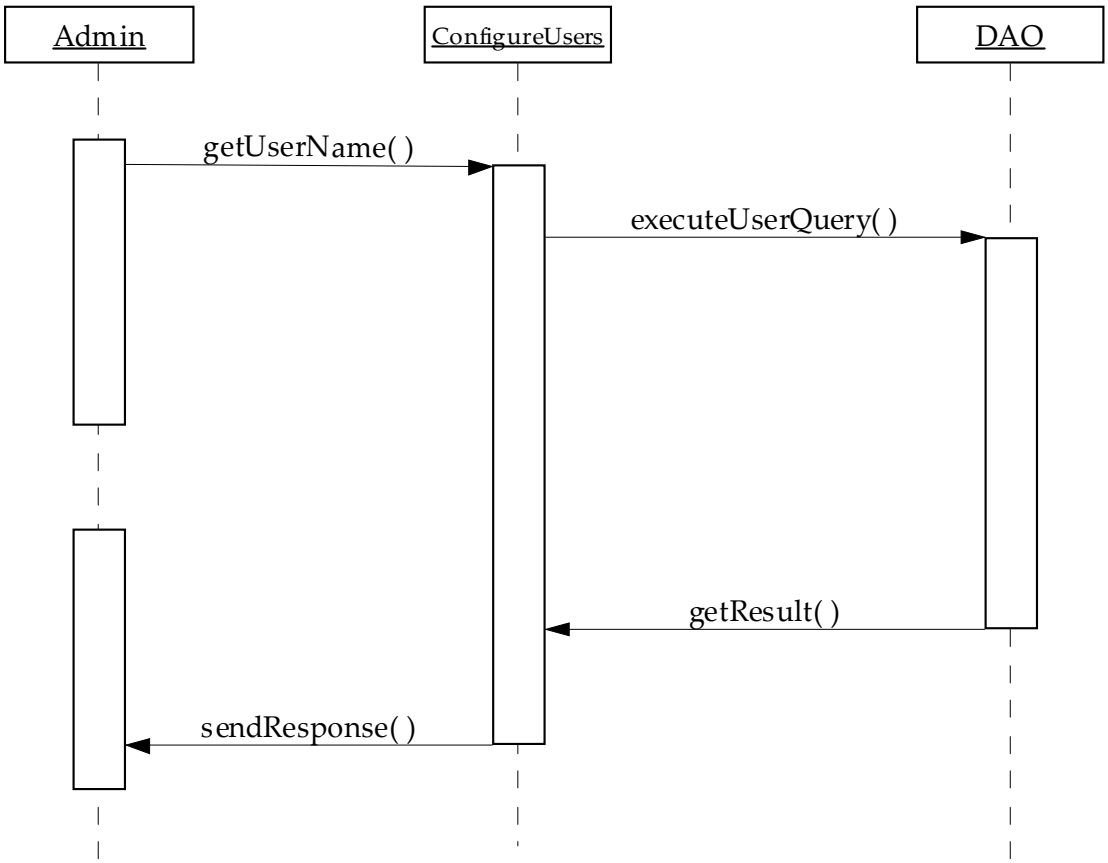
4.3 Sequence Diagrams

4.3.1 Admin

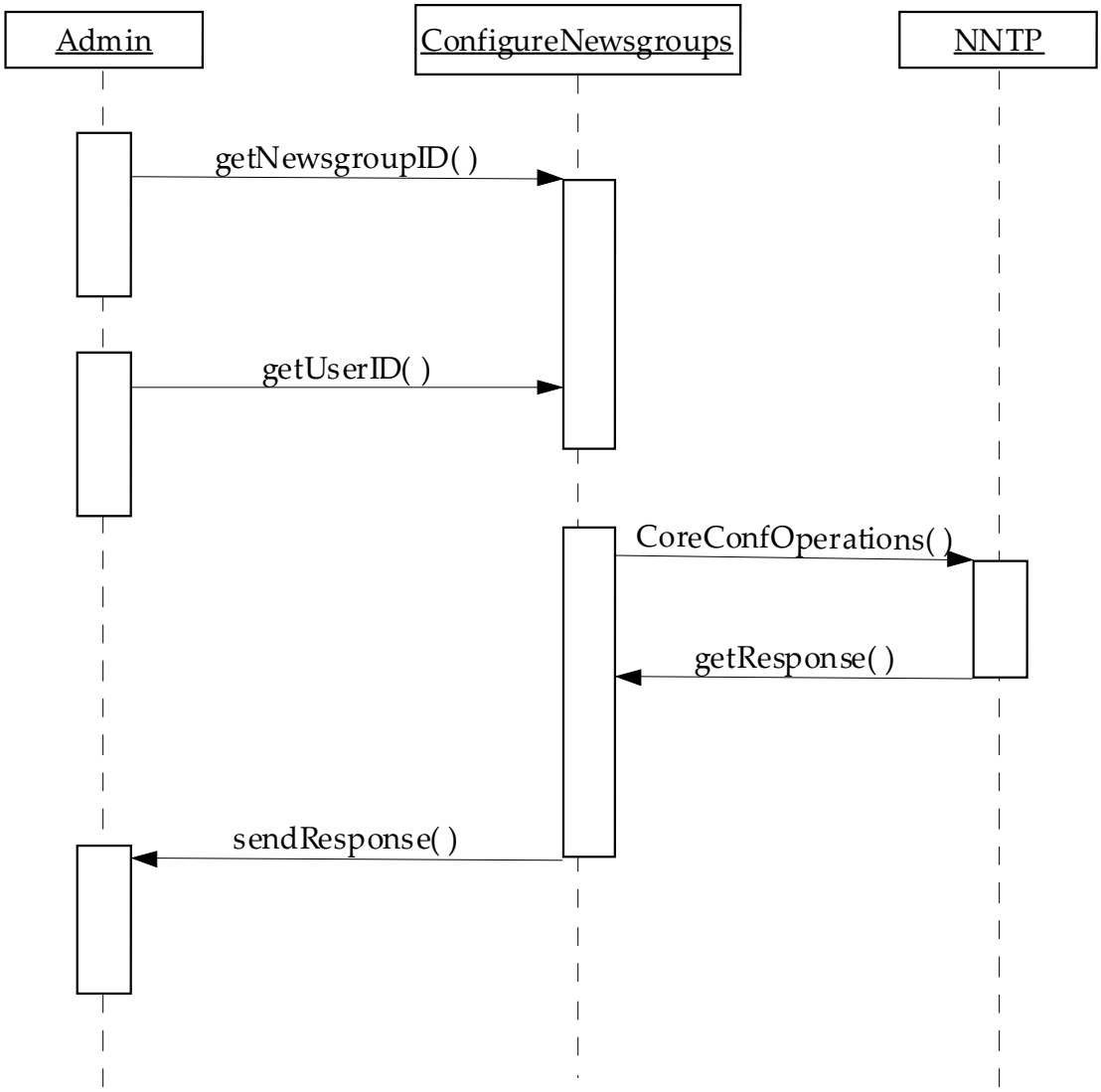
4.3.1.1 Newsgroup Configuration Process



4.3.1.2 User Management Process

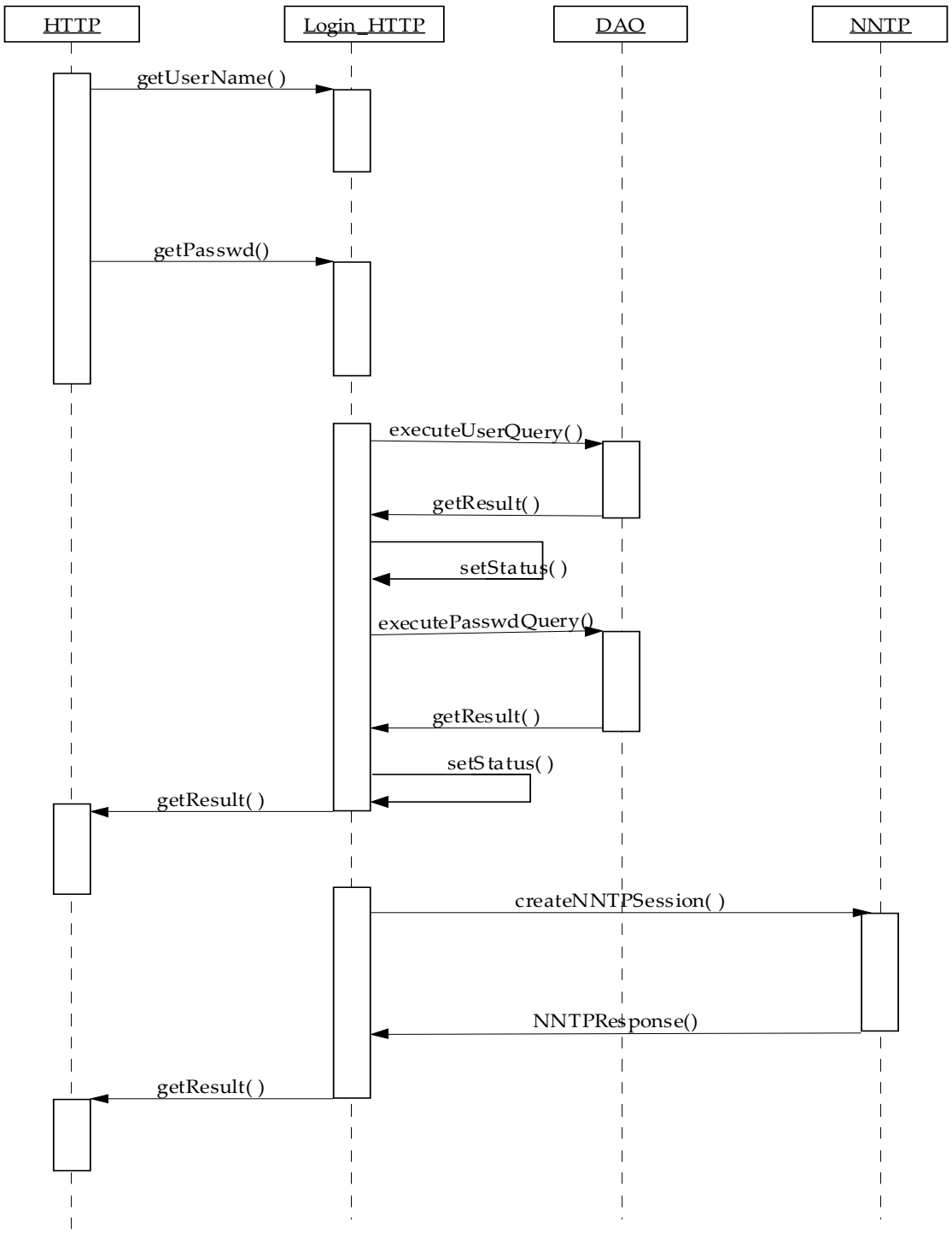


4.3.1.3 Newsgroup – User Configuration Process

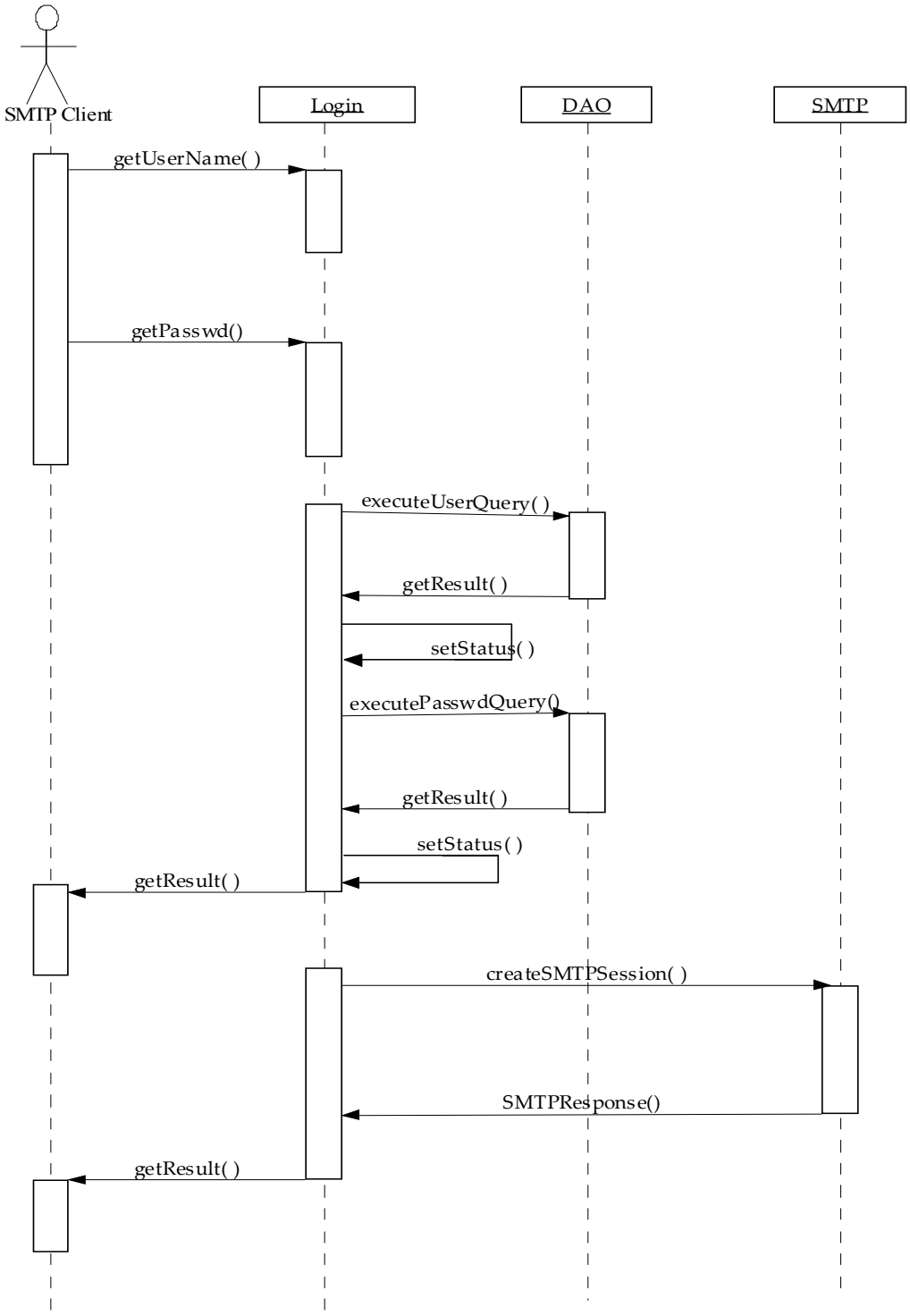


4.3.2 Login Process

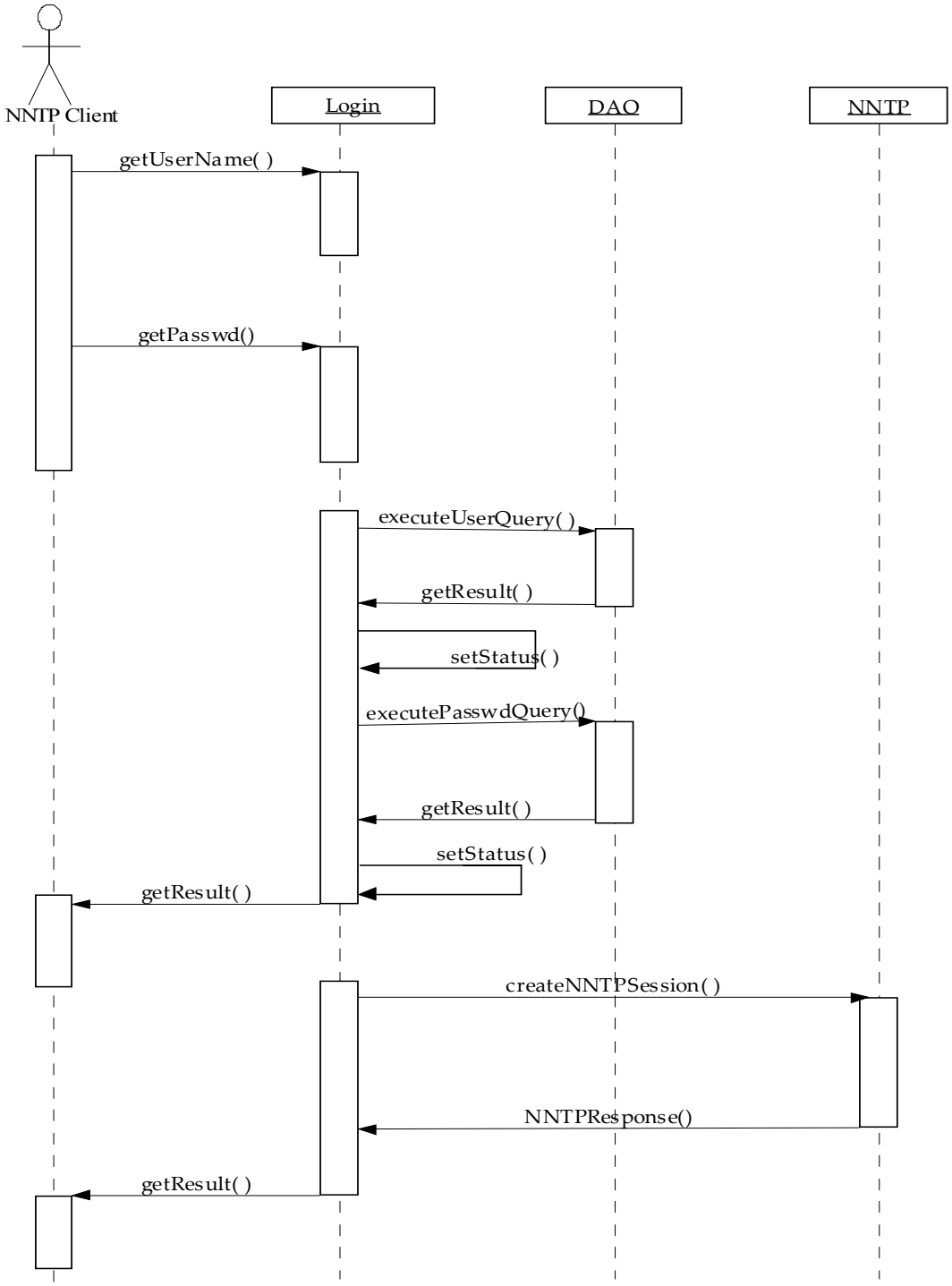
4.3.2.1 HTTP Login Process



4.3.2.2 SMTP Login Process

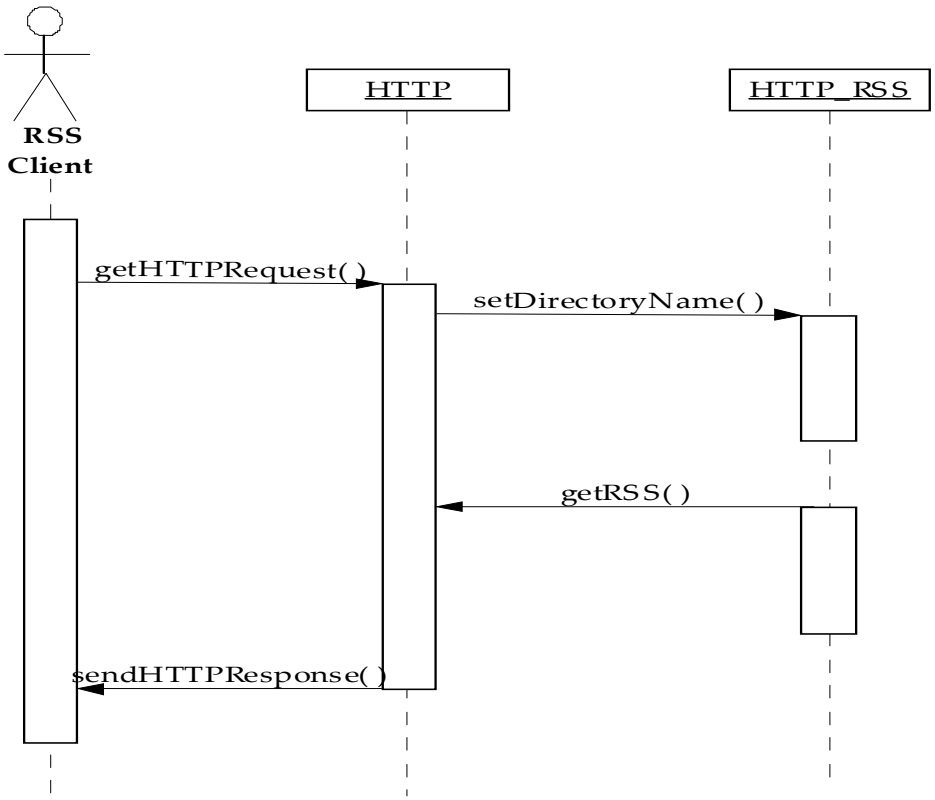

SMTP Client
Login
DAO
SMTP
getUserName( )
getPasswd()
executeUserQuery( )
getResult( )
setStatus( )
executePasswdQuery()
getResult( )
setStatus( )
getResult( )
createSMTPSession( )
SMTPResponse()
getResult( )

4.3.2.3 NNTP Login Process

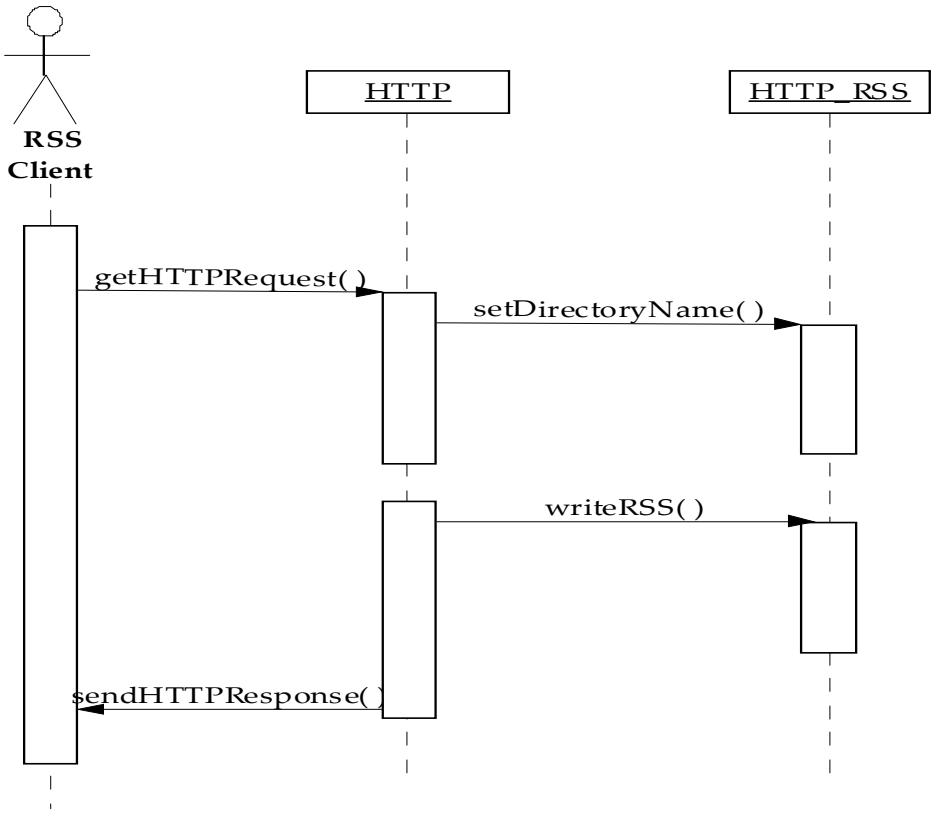


4.3.3 RSS

4.3.3.1 RSS Read Process

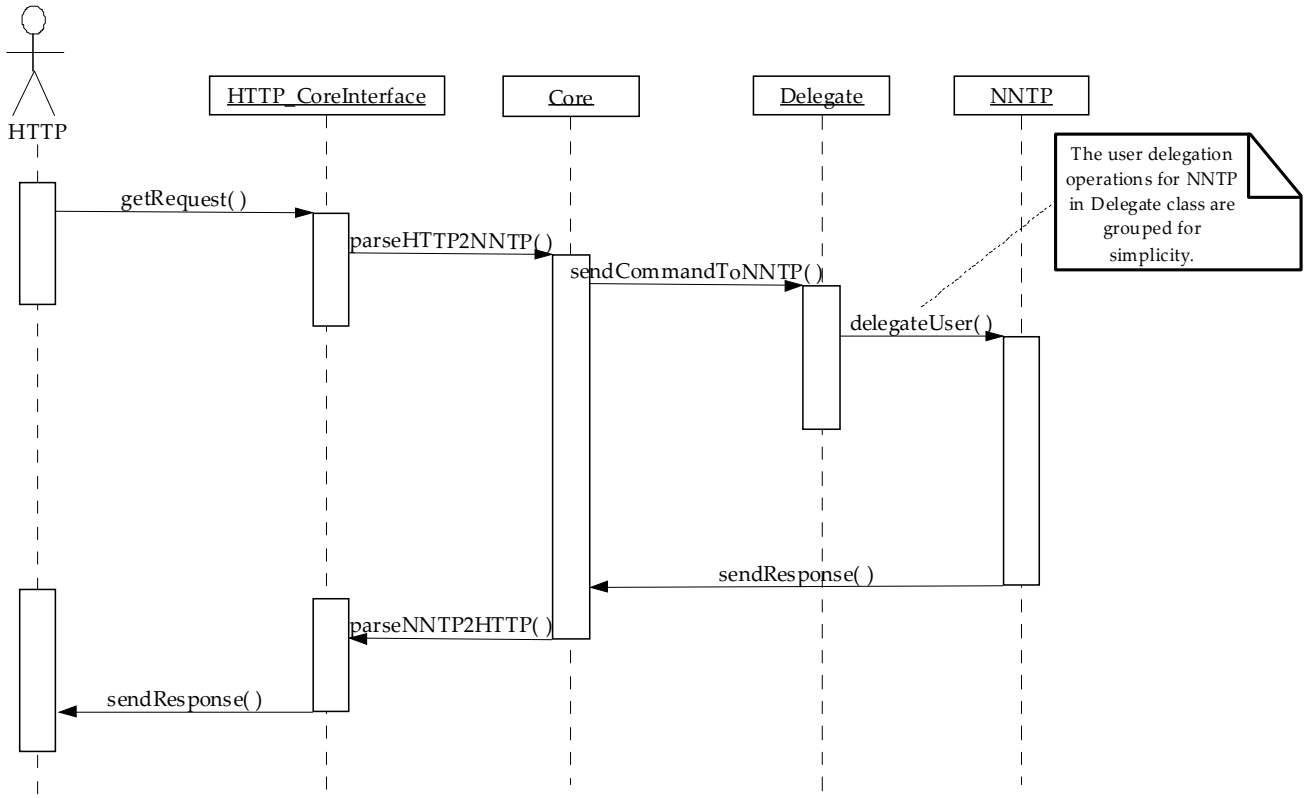


4.3.3.2 RSS Write Process

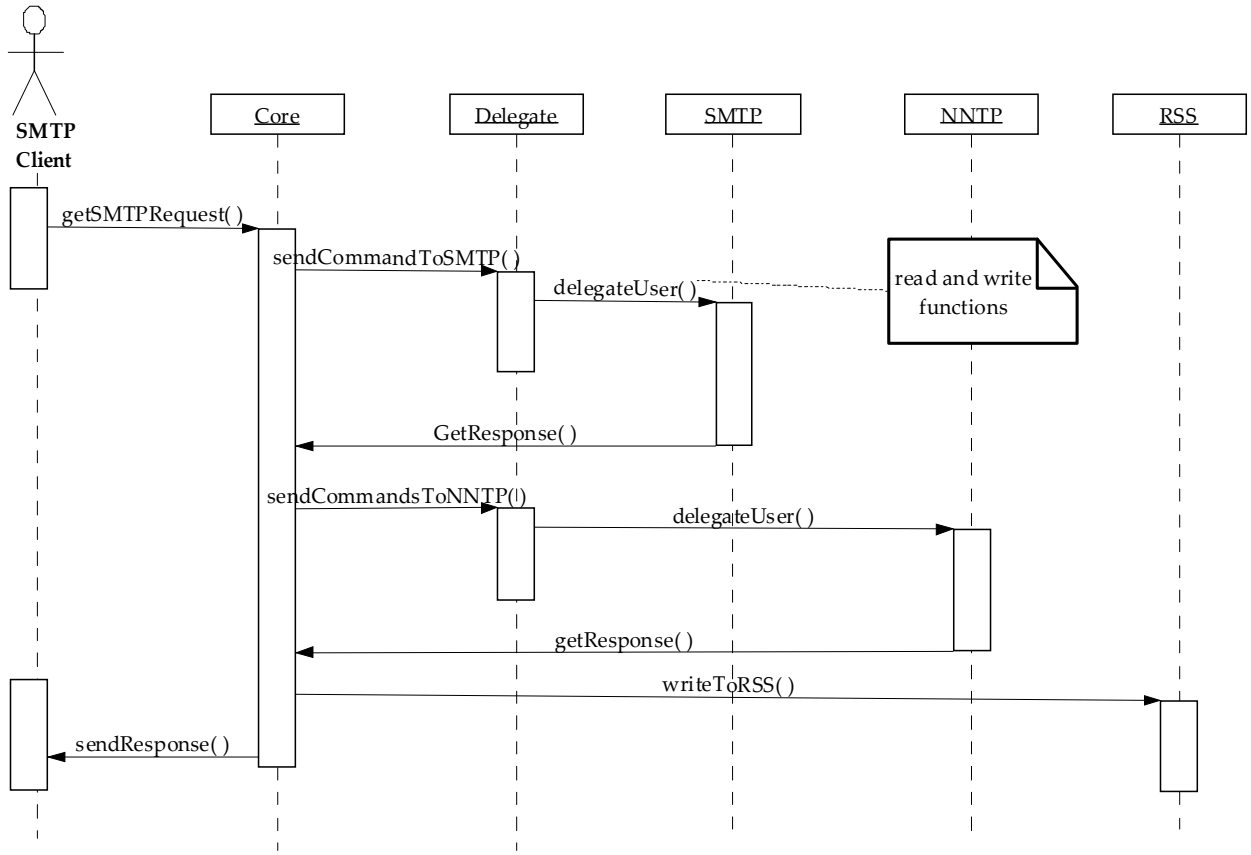


4.3.4 Logged User Access

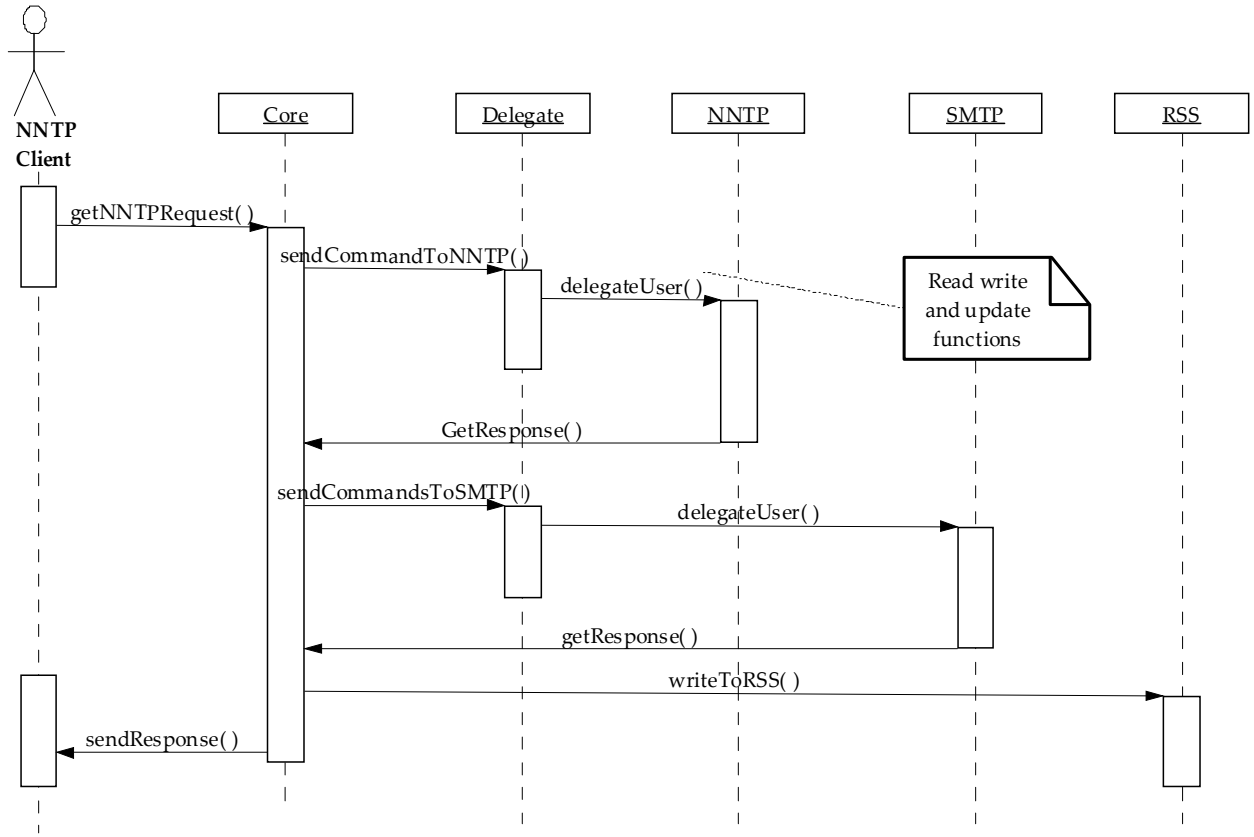
4.3.4.1 HTTP Access Process



4.3.4.2 SMTP Access Process



4.3.4.3 NNTP Access Process



5 Requirements

5.1 Functional Requirements

Our project's main concern is establishing communication between mail clients or news clients and built-in servers. These mentioned clients can be of four types, which are:

- NNTP Client

- SMTP Client
- HTTP Client
- RSS Client

And the servers can be of three types, which are:

- NNTP Server
- SMTP Server
- HTTP Server

Among these clients RSS Client and HTTP Client uses HTTP Server in order to communicate with news server. Since an HTTP to NNTP converter operates between HTTP Server and the core that we are going to build, core perceives an HTTP request as an NNTP request. Hence core's role in this communication process is handling NNTP and SMTP requests.

Serving both NNTP and SMTP clients on the same server sometimes requires having one type of request changed to another (e.g. NNTP to SMTP) and delivered to both servers. That is the servers need to be synchronized. In this project three main levels of synchronization need to be considered, which are:

Write Level Synchronization

When a write request comes from an NNTP or SMTP client, the request must be transmitted (in our case, delegated) to all three servers. Thus NNTP2SMTP or SMTP2NNTP conversion is made and the incoming message is delivered to all three servers. If any one of the servers fails to write the new message, then the

operation fails, all changes must be undone and a “failure” message must be sent to the client. Otherwise related RSS file must be updated and a “success” message must be sent to the client.

Read Level Synchronization

When a read request comes from an NNTP or SMTP client, since it only requires a read operation on the related server, no conversion is needed. Read request is delegated to the related server and the answer is sent to the client.

RSS clients’ read request is fulfilled on the HTTP server via an .xml file, which contains the RSS data.

Update Level Synchronization

Update requests originate only from NNTP clients. Such a request is delegated to NNTP server however, since a mail can not be updated, the needed action is to delegate the updated message to SMTP server as a write request. If the updated message is on a newsgroup which is used as RSS feed, related .xml file must also be updated. If any one of these attempts fail, all changes must be undone and a “failure” message must be sent to the client.

Another main requirement of our project design is user delegation, which can be described as the core acting like a server to the clients and like a client to the servers. This is needed for maintaining security and sending requests to both NNTP and SMTP servers. In order to maintain security using SSL option is given to the users and using SSL needs establishing a session between client and server. We can not directly transmit a request coming from the client to the server, because we have to convert it by SMTP2NNTP or NNTP2SMTP functions

and send the same request to both of the servers. Thus that session is needed to be established between the core and the servers instead of the clients and the servers and this is what we call user delegation.

User capabilities should also be mentioned as functional requirements. Our design allows system administrators to

- create, edit or delete newsgroups,
- create, edit or delete mail groups,
- add, delete users,
- edit type of users
- edit user types,
- archive or delete articles,
- change RSS path of newsgroups,
- change server ports;

Logged users to

- read, write articles,
- update articles (for NNTP users),
- change password,

- edit information,
- change preferences,
- edit mailing list subscriptions.

Additionally, if admin allow newsgroups to be open to everyone, then even unsubscribed people can read articles and write comments by using newsreaders or RSS.

5.2 Non-Functional Requirements

The final product should meet these nonfunctional requirements:

Platform Independence

GÜVERCİN should be able to operate on both Windows and Linux servers.

Secure Data Transmission

GÜVERCİN should provide its users an option to use SSL which provides secure communications on the internet.

Reliability

GÜVERCİN should be as bug free as possible. Coherence of built-in servers with each other should be sustained.

Modularity

GÜVERCİN should be built on interrelated modules. This will provide easier

testing and maintenance.

Developer-friendly

GÜVERCİN should be implemented as modular as possible in order to understand the code and implement future plug-in's easier.

User-friendly admin configuration

GÜVERCİN should offer a user-friendly admin panel in order to provide the admin an easier access to the system.

5.3 Minimal Software and Hardware Requirements

These two requirements are as same as they are stated in the analysis phase.

6 User Interface Design

6.1 Design Principles

The user interface has been designed and improved in accordance with the principles listed below.

- The user interface should be simple and clear and so that the user will learn how to use it easily

- The interface should provide the possible shortest way for performing an action.
- The user should be able to easily switch to other interface parts which are irrelevant to his intended path.
- The user should never be forced to memorize.
- The user interface components should be consistent. Colors, buttons and graphical icons should be standardized in all the components.
- The interface should represent entities of the same group with different properties with different colors or symbols.
- Grouping of relevant for similar interfaces which provides interaction with different components is important.

6.2 User Interface Description

6.2.1 Logged-User Interaction

In this section how the two types of the users "admin" and "logged-user" interact GÜVERCİN will be described. "The logged user" refers to the user who reads the news in any of the four ways in context of this graphical user interface design section.

The logged-user interacts with GÜVERCİN in the following ways :

- The user will send to or receive mail from an e-mail list if he is subscribed.
- The user reads and post news to newsgroups via the forum interface.
- The contents of your newsgroups can now be accessed in two different formats used by news aggregators:To access the newsgroups in RSS format, the RSS reader should access the *<servername>/rss/read?newsgroup=<newsgroup>* provided by the server

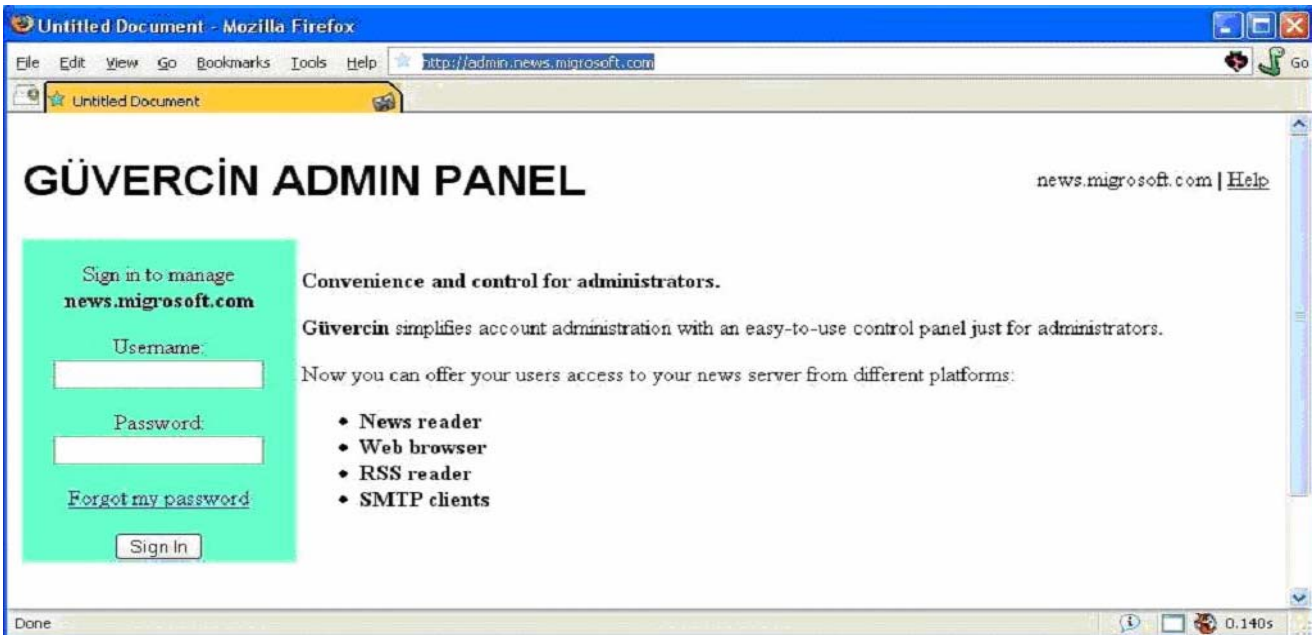
6.2.2 Admin Panel

After describing the interaction of the end-user, the interaction of admin will be described. Administrator can manage the server settings, users and groups, namely the whole system by the Admin Panel.

Below are the screen shots and explanations of the GÜVERCİN Panel. This is an initial design so there may be many improvements in the next phases both in terms of organization and presentation. The snapshots will be described in a walkthrough manner.

To access the GÜVERCİN Admin Panel the admin writes *htttp://admin.news.migrosoft.com*

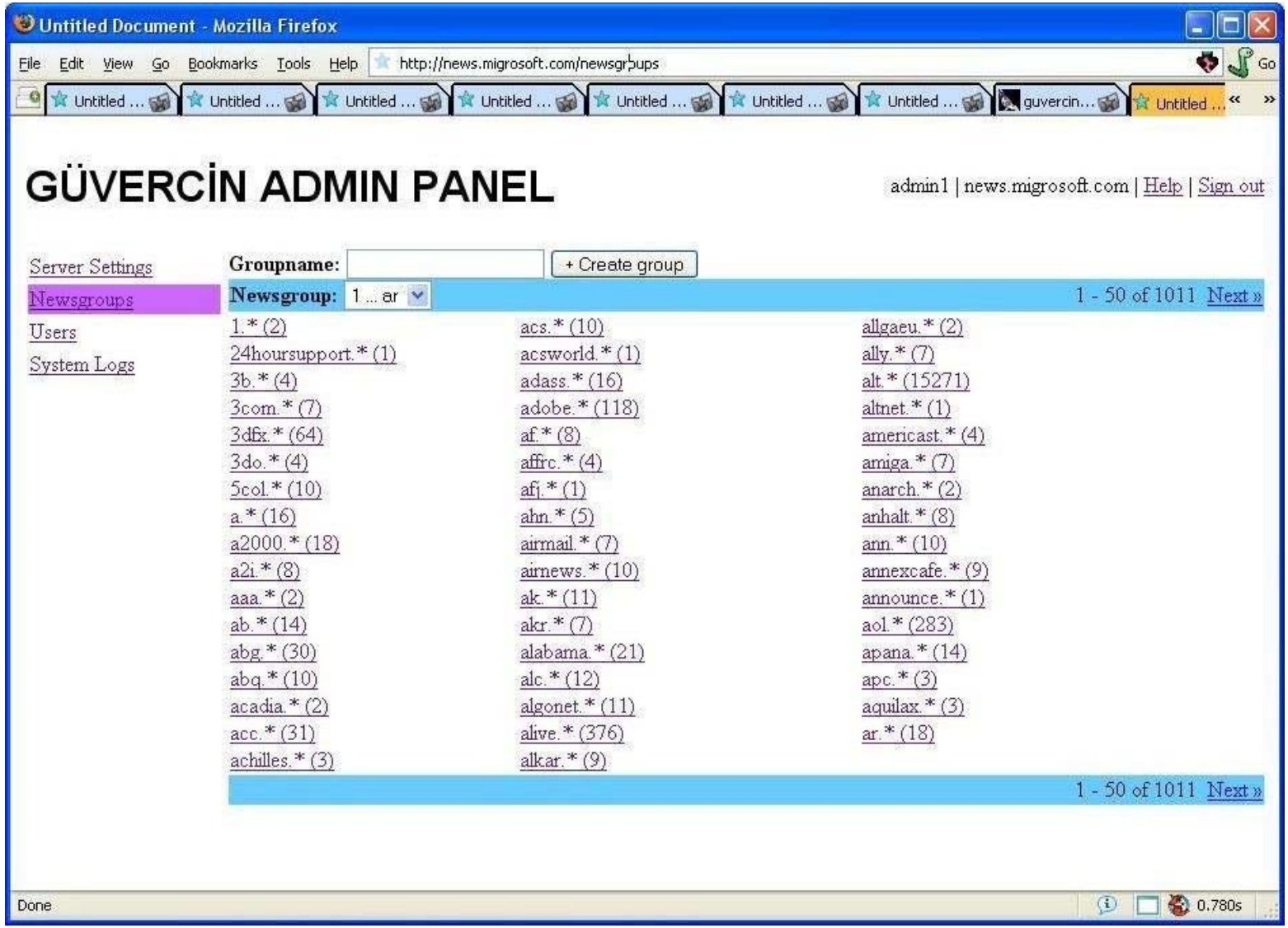
The path here is in form *htttp://admin.<server-name>*. The admin faces this interface:



The admin is asked to log-in. If the admin logins successfully, he will face this page.

As you can see on the right top corner the name of the server and the username of the admin is displayed. By default all the newsgroups on the server will be listed. Using the link on the left side the user will be directed to the different settings pages.

6.2.2.1 Newsgroup Management


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://news.migrosoft.com/newsgroups
Go
GÜVERCİN ADMIN PANEL
admin1 | news.migrosoft.com | Help | Sign out
Server Settings
Newsgroups
Users
System Logs
Groupname:
+ Create group
Newsgroup: 1 ... ar
1 - 50 of 1011 Next »
1.* (2)
24hoursupport.* (1)
3b.* (4)
3com.* (7)
3dfx.* (64)
3do.* (4)
5col.* (10)
a.* (16)
a2000.* (18)
a2i.* (8)
aaa.* (2)
ab.* (14)
abg.* (30)
abq.* (10)
acadia.* (2)
acc.* (31)
achilles.* (3)
acs.* (10)
acsworld.* (1)
adass.* (16)
adobe.* (118)
af.* (8)
affrc.* (4)
afj.* (1)
ahn.* (5)
airmail.* (7)
airnews.* (10)
ak.* (11)
akr.* (7)
alabama.* (21)
alc.* (12)
algonet.* (11)
alive.* (376)
alkar.* (9)
allgaeu.* (2)
ally.* (7)
alt.* (15271)
altnet.* (1)
americast.* (4)
amiga.* (7)
anarch.* (2)
anhalt.* (8)
ann.* (10)
annexcafe.* (9)
announce.* (1)
aol.* (283)
apana.* (14)
apc.* (3)
aquilax.* (3)
ar.* (18)
1 - 50 of 1011 Next »
Done
0.780s

The user can add a new group by the “+create group” button. When the user enters the group name and clicks the button, a newsgroup is created and a settings page will be immediately opened so that the admin can set properties of the new group. Every newsgroup can be managed through such as explained below.

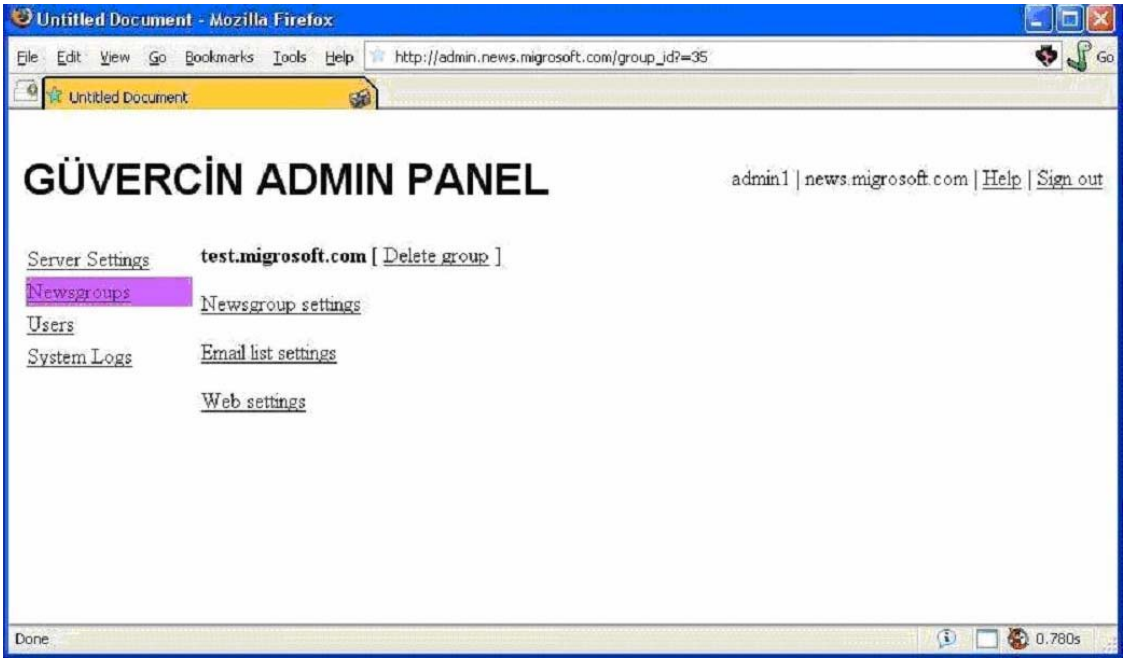
The user can see the news groups in an interval according to alphabetic order by the combo menu. The newsgroups are listed here in the group hierarchy. To illustrate the link “announce.*” refers to the group of groups

announce.design

announce.design.moderated.

When the user clicks the “announce.*” link, links to this groups will be displayed.

When a group link is clicked, the management page for that group will be opened. Snapshot of this page is given below:

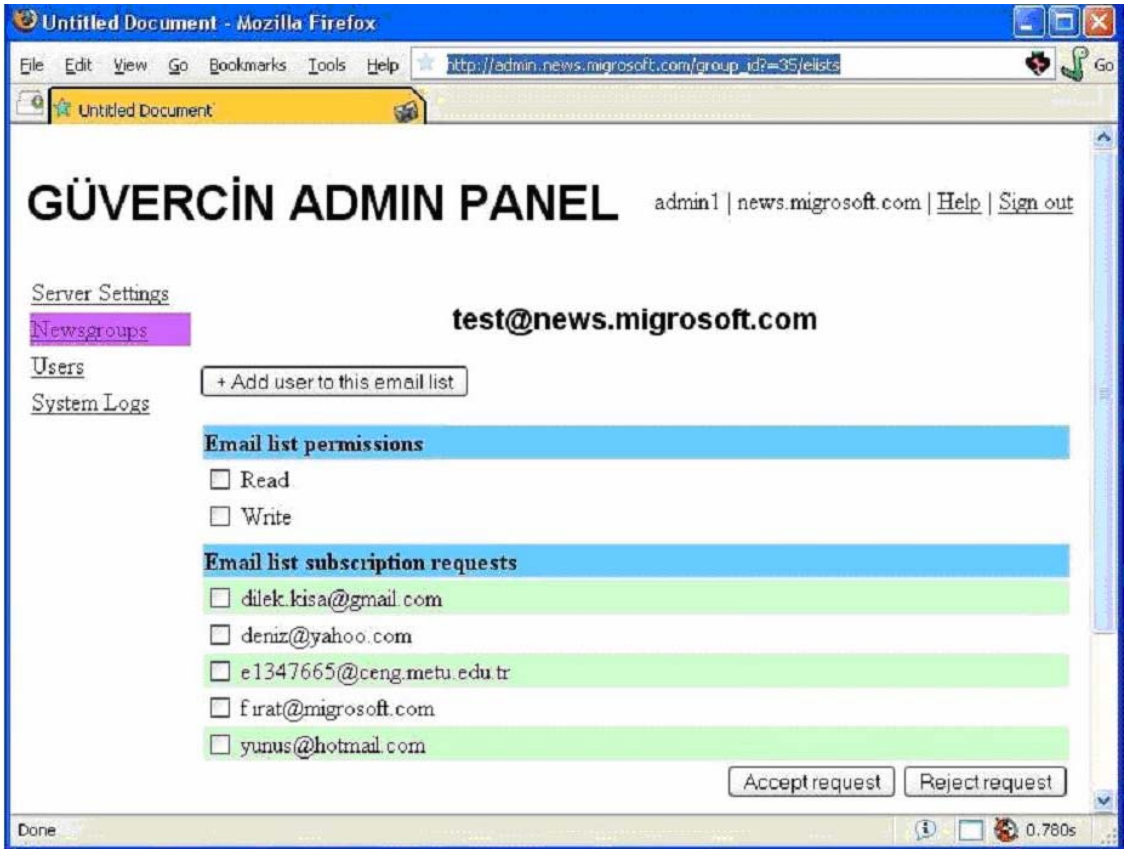


From this page the user can set the settings for the e-mail list and the forum settings associated with this group. When one of these links are clicked one of the three pages below are opened.

In the email list page the mail addresses waiting to be approved for subscription are listed if the subscription requires approval. The admin can select the users by the checkboxes and accept or reject them by clicking the reject/submit button. The permission checkboxes "Read "and "Write" have the following meanings:

- *Read*: The users can read messages written by other people to the e-mail list.
- *Write:* The users can post new messages of their own to the e-mail list.

These permissions are valid for all the users subscribed to the mail list.

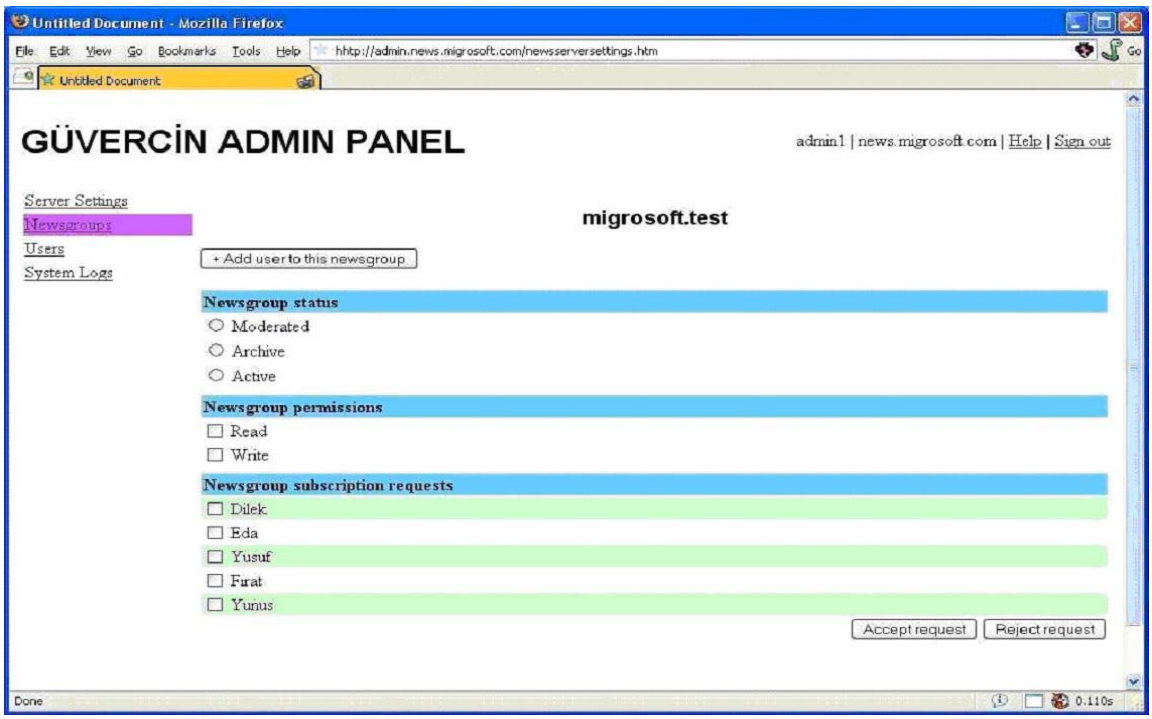


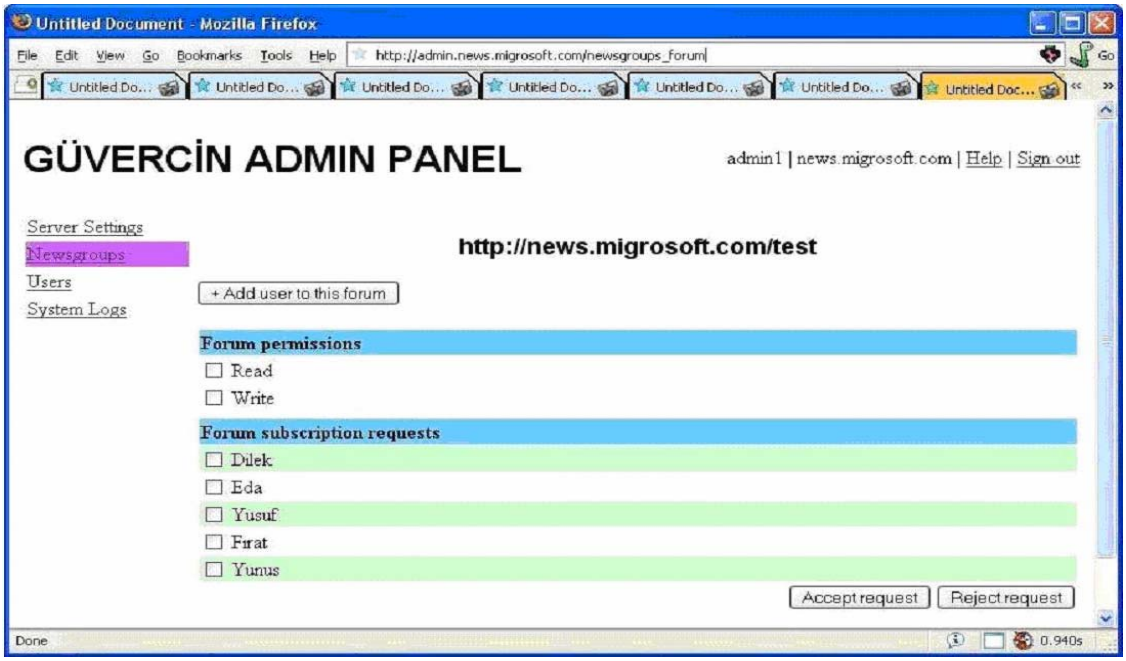
In the newsgroup settings page for a newsgroup the status is set one of the followings:

This indicates whether new messages will be allowed in the newsgroup. The available options are:

- *Active*. This is the default value, and indicates that new messages are allowed to be posted to the newsgroup.
- *Moderated*. This indicates that new messages are allowed to be posted to the newsgroup, but they must be approved by a moderator before they will be visible to others.
- *Archived*. This indicates that no new messages are allowed to be posted to the newsgroup. This is typically used to allow the messages from an old newsgroup to be read, while moving new messages to a different newsgroup.

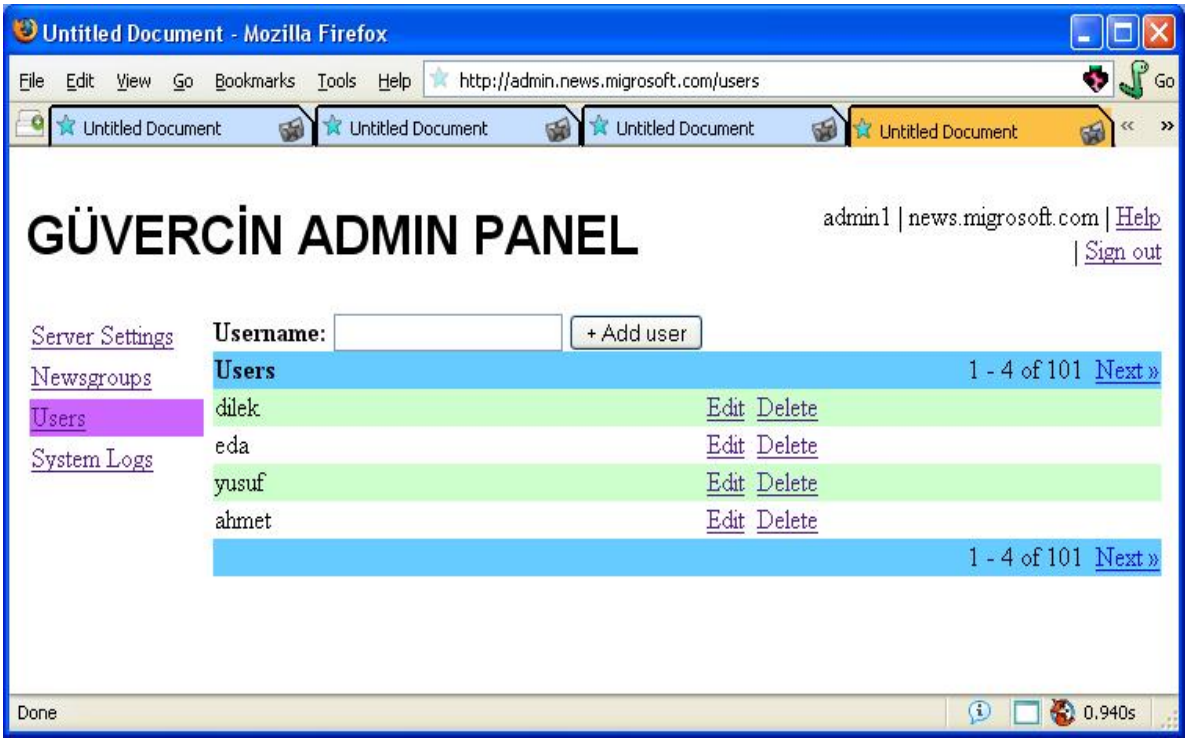
All the Subscription Requests part in these three settings windows corresponding to a single group will not occur if no authentication is needed.





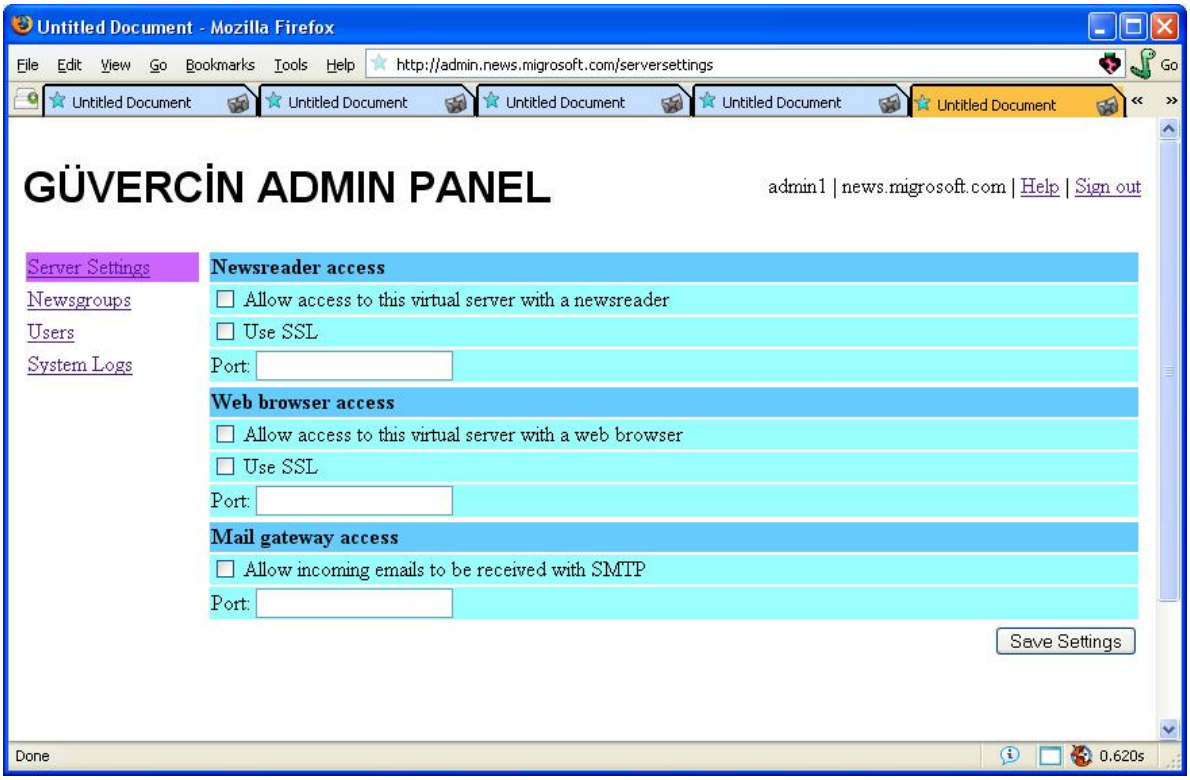
6.2.2.2 User Management

By clicking main page the admin can get to the interface to manage users. The users of the server are listed. The user information can be updated from this site. Moreover new users can be created.



6.2.2.3 Server Settings

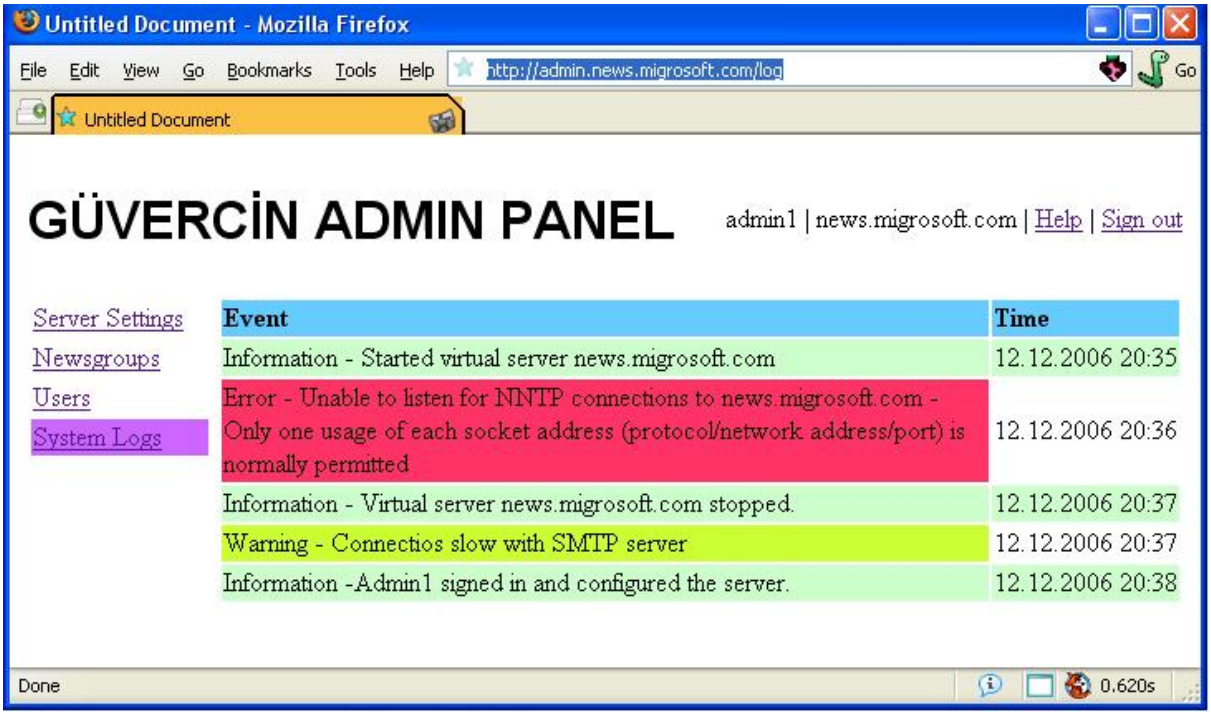
This part of the interface is for the configuration of the communication of the server with clients and other servers (SMTP servers for the mail business).



Moreover on the security side, the SSL configuration will be performed on this interface. Only a check box is provided for now but it will be added in the next design phases. Moreover there will be settings for the POP3 server and all the settings in this page will be much more detailed.

6.2.2.4 System Logs Page

Events of the system are presented in tabular form. Time info is also provided. Different colors are used for information entries, warning and error entries. The entries here are not logical, but text is added to represent the format better. Moreover instead of writing "Information"," Error" or "Warning" simple symbols will be used in the real implementation. This will be introduced in the next phases of design.



7 Testing Strategies

This initial design plan will be improve in next design phases.

7.1 Unit Testing

Modules will be tested for functionality and performance before integration.

- **Login module Tests:**
  - Can the administrator log in to the system securely?
  - Can session info be retrieved and written successfully?

- **HTTP Module**
  - Can the module communicate with a browser ?
  - Can the module communicate with an ordinary NNTP server?
  - Can the module send messages to a newsgroup?

- **RSS Module**
  - Can the xml file for the RSS feed be accessed?
  - Is the XML file updated regularly?

- **DBLayerModule**
  - Can the database be accessed continuously?
  - Do database operations cause any unwanted changes or corruption in the database?

- **Administrator Module**
  - Are the administrator requests correctly translated into actions? Do these actions succeed?
  - Does the interface work correctly?

- **Core module**
  - Can the core module correctly translate SMTP requests to NNTP requests and the other direction?

7.2 Integration Testing

Integration testing is the most important part of this project. Since the team intends to use open-source extendible software as components, the integration of these requires effort. The integration testing mainly will focus on whether synchronization can be achieved between the four user access ways the e-mail lists, the forum, RSS and the newsreaders.

7.3 GUI Testing

For the functional testing the team aims to use an automated GUI tester, the Eclipse Test and Performance Platforms Automated GUI Recorder. Moreover the TPTP will be used for stress testing and profiling.

8 Project Schedule

8.1 Project Milestones

The updated work packages are the following:

Work Package 1 – Project Management

- Project Proposal
- System Requirements Specification and Analysis Report
- Initial Design Report
- Final Design Report

Work Package 2 – Core

- Database Implementation
- User Authentication Implementation
- NNTP2SMTP Module Prototype Implementation
- Security Implementation

- SMTP2NNTP Module Prototype Implementation
- Prototype Integration and Testing
- Prototype Release
- NNTP2SMTP Module Complete Implementation
- SMTP2NNTP Module Complete Implementation
- RSS Module Implementation
- Core System Integration
- Alpha Testing
- Beta Testing and Debugging
- Core System Release

Work Package 3 – Admin Panel

- GUI Design
- GUI Implementation
- Panel Implementation
- Alpha Testing
- Beta Testing and Debugging

Work Package 4 – Finalizing Activities

8.2 Gantt Chart

Our updated Gantt chart can be seen in the Appendix.

9 Appendix

Figure 19: First Part of the Gantt Chart

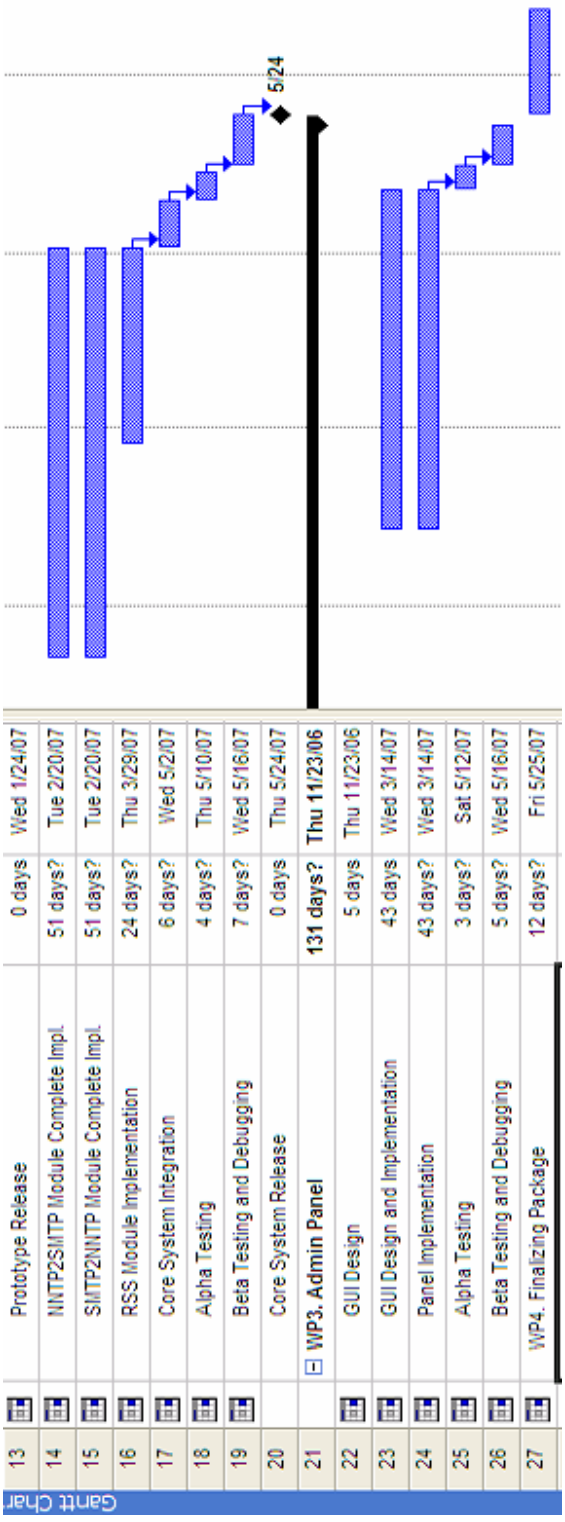


Figure 20: Second Part of the Gantt Chart

References

[1] http://en.wikipedia.org/wiki/News_server