Middle East Technical University

Department of Computer Engineering

Final Design Report

by

Migrosoft®

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 Problem Definition

As the technology evolves, the way that people get across had changed compared to as it was in ten years ago. Internet is intensively being used as a communication environment since 1995[1]. Today most of the people is using web to retrieve and send information since Internet is the largest and fastest way of this. Newsgroups and mailing lists have been the most popular approaches to this demand till now. After RSS 2.0 released in 2003, it has also become one of the most popular news feeding system around the Internet users. Besides there are still considerable amount of users using newsgroups and mailing lists. Since these methodologies use different conventions and nowadays news servers can not combine all of these conventions on their constitution, there's a serious need for an all-in-one news server that's synchronizing RSS, newsgroups and mail-groups. Users can read or wirte news via newsreaders, mail, RSS or a simple web-page. Our aim in making this project is making a unified news server that has the abilities of receiving and delivering messages by using the methods mentioned above.

1.3 *Design Considerations and Constraints*

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

- The software must meet the functional and non-functional requirements.
- 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, continuously 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 Requirements

2.1 *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 converted 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 fails, 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,
- edit mailing list preferences.

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.

2.2 *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.

Modularity

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

2.3 Minimal Software Requirements

- Güvercin can perform on both Linux and Windows systems, one of these operating systems are required.
- ***Güvercin*** will be implemented in ***Python***. Although the team has initially decided on PERL, the last decision of the team is to change the implementation language from PERL to PYTHON. Both languages are script languages but PYTHON is designed as an object-oriented language and has a simpler syntax. The project design is object-oriented; to map classes from Güvercin design to PYTHON will be easier. Nowadays many Linux distributions include a recent version of Python; also some Windows machines come with Python already installed.
- ***Java EE SDK*** is needed for Apache James, which we will use as built in mail and news server, to work properly.

2.4 *Minimal Hardware Requirements*

Although news servers best perform on server systems, our product Güvercin can perform quite well on a system with Pentium 4 main processor, 512 MB of RAM, modem or network card.

2.5 *Development Requirements*

Güvercin will need the following technical requirements in order to be developed:

- **Apache Web Server**: Apache[2] will be used as our HTTP Server because it is compatible with the other tools we will use.

- **Apache James:** Our unified news server needs built-in NNTP and SMTP servers as well as a mailing list interface. We will use Apache James as it offers all together.

- **Java EE SDK:** Apache James needs Java EE SDK to function properly.

- **Python:** We will use python programming language in order to implement the core of Güvercin. Python is the interpreter, which is named after the language.

- **Eclipse with plug-in PyDev:** Eclipse is an open source program which can be used for various purposes. We will use it as a python development environment.

- **Macromedia Dreamweaver MX 2004**: As we will implement admin

GUI by using PHP, Macromedia Dreamweaver is the best choice for designing a comprehensive admin tool.

3 Architectural Design

Modularity is an important point for all software products in some aspects like development, efficiency and easy-interferebility. The logic of modularity is more likely to be seen in server systems like our product. After deeply analyzing the overall capabilities of GÜVERCİN, we decided to divide whole system into seven modules. The first one is the login module which basically creates a session between the core module and each of mail and news server modules. HTTP module provides the web interface of our system. It also provides the web-based news accessing. For configuration utility for administrator and users, the module interacts with the user configuration module and administration module. The module is also in interaction with the core and mail server modules. DBLayer module just forms an interface for our database system. The existence of this module also provides flexibility, that is we can change our database without a detailed modification on our system. RSS Module is interaction with the Core and HTTP modules for reading RSS feeds and writing RSS data. According to this separation, the interaction between these modules are as follows:

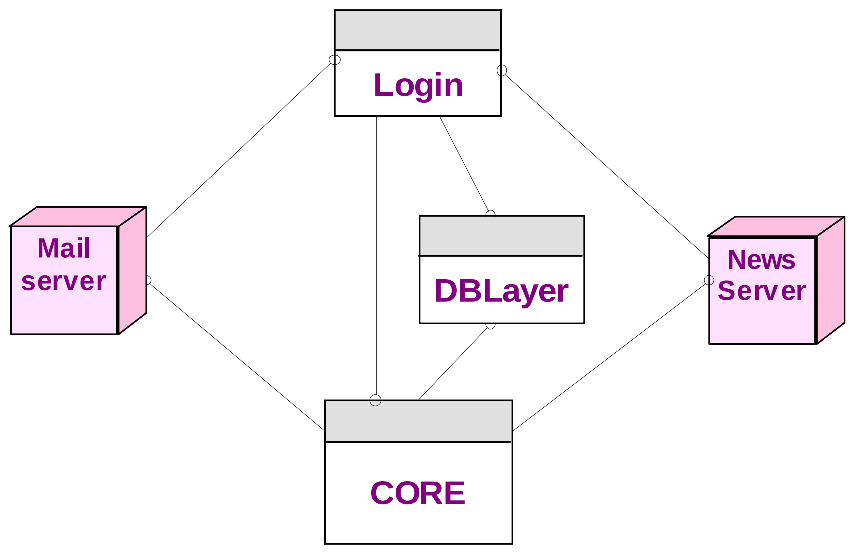


Figure 1: Interaction of Core, DBLayer, Login Modules

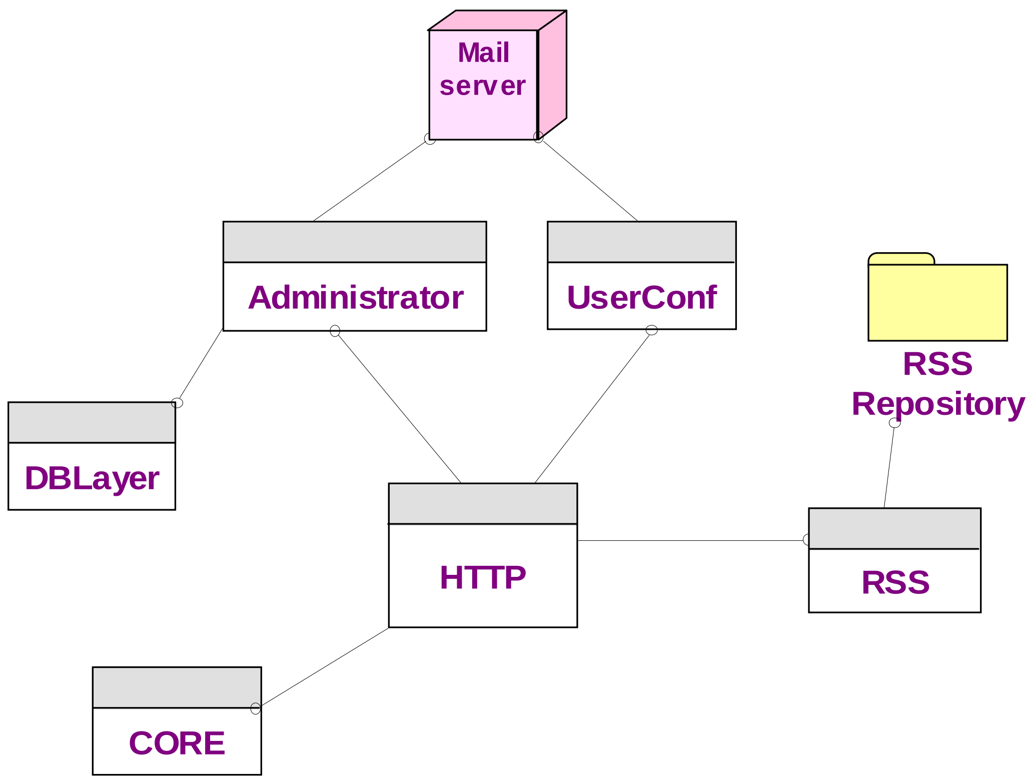


Figure 2: Interaction of Core, HTTP, DBLayer, RSS, UserConfiguration and Administrator Modules

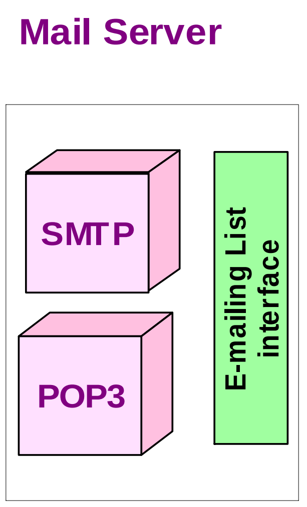


Figure 3: Mail Server Components

Since NNTP and SMTP modules will be used as built-in servers, we do not need to explain them among our main modules. As a brief information, we will use James Server [3] which provides a news server, mail server and an e-mailing-list interface for our product. The product holds the articles in two ways: file based or in its own database which we will prefer to use. Apart from the James Server the other 7 basic modules of the GÜVERCİN is explained as follows:

3.1 *Login Module*

Login is one of the most important modules of GÜVERCİN with Core Module and Administrator Module. The main functionality 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. According to our design, RSS clients do not have to be authenticated. The module just gathers the user information, e-mail address and password, and tries to establish a connection between the core module and both of NNTP and SMTP modules on behalf of the users. At first glance, this approach may seem inefficient but it is necessary since the commands of users will not be directly sent to the mail and news servers. Indeed, the mail and news clients connect to our core rather than directly connecting to the built-in news and mail servers. So for providing a secure connection for the messages which are sent from/to our system, a session must be created between the session and servers and the commands of the authenticated users must be delagated by the core part. To solve this problem, a login module is needed. For the final remark, this module considers the HTTP users as they are NNTP users by converting the HTTP requests to NNTP requests. So the module is also in contact with the HTTP module.

3.2 HTTP Module

This module plays a crucial role for our project. Although we will use a built-in package for HTTP news reader clients, the capabilities of this module can be listed 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
- Constituting a reservoir for RSS files and email list web interface.

3.3 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. By using this module the subscribed users of GÜVERCİN may change his/her personal details, e-mail list subscription settings and mail options. For example, a subscribed user may change the option of receiving mails as daily digest or seperately.

3.4 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. This module is also in contact with the database for fetching and updating system the information of system, users and server settings. The module plays a crucial role about the interaction of the components of our system.

3.5 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.

3.6 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.

3.7 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. The module works whenever a new message is posted to a newsgroup or e-mailing list which has a RSS feed option. So if such a sitution occurs, the RSS file for that message is created by the core and placed to the RSS storage by the RSS module.

4 Data Design

We have revised the entity relationship diagram drawn in the analysis requirements phase and altered data 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 built-in servers' own databases. Besides there is no need to store subscribed users' mail addresses since mailing list databases already store them.

4.1 Data Objects

4.1.1 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 have to store the user's information.

If the user is an administrator, he/she will be able to configure the system with his/her administrative rights.

- user_id
- username
- common_surname
- common_name
- user_type_id

4.1.2 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

4.1.3 Banned User

BannedUser entity is a 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

4.1.4 AllowedUser

AllowedUser entity is a user who has additional rights from a naive user. AllowedUser entity is added to system in case of the restricted newsgroups, i.e. there may be an administrator announcement newsgroup allowing only the administrators to send messages, forbidding 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

4.1.5 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

4.1.6 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

4.1.7 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 ban a unique id, which will be used as a primary key.

- ban_id
- ban_type
- ban_period

4.1.8 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

4.1.9 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

4.1.10 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

4.2 *Entity-Relationship Diagrams*

4.2.1 Data Objects

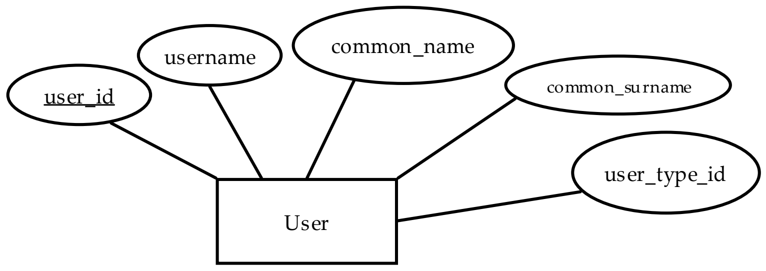


Figure 4: User Entity

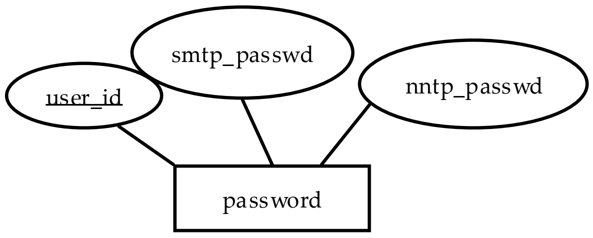


Figure 5: Password Entity

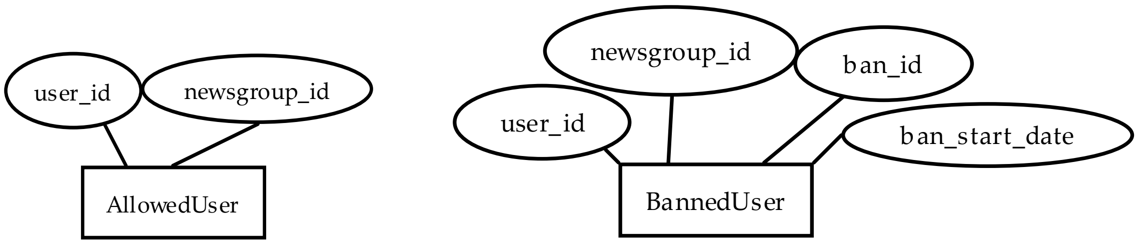


Figure 6: Users

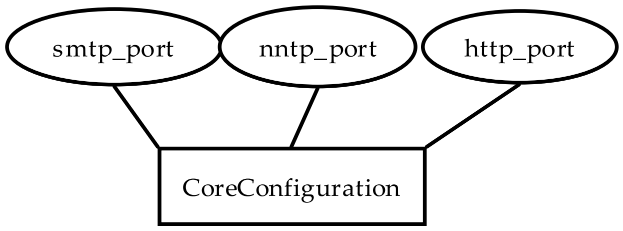
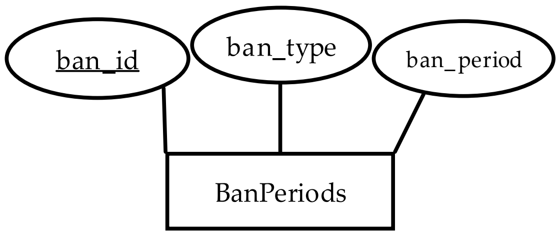


Figure 7: CoreConfiguration Entity



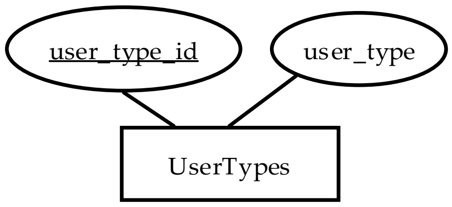


Figure 8: BanPeriods Entity

Figure 9: UserTypes Entity

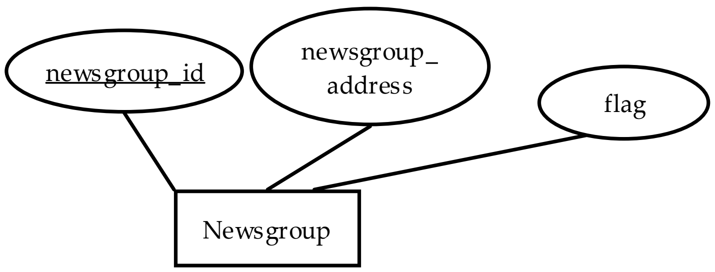


Figure 10: Newsgroup Entity

4.2.2 Entity-Relationship Diagram

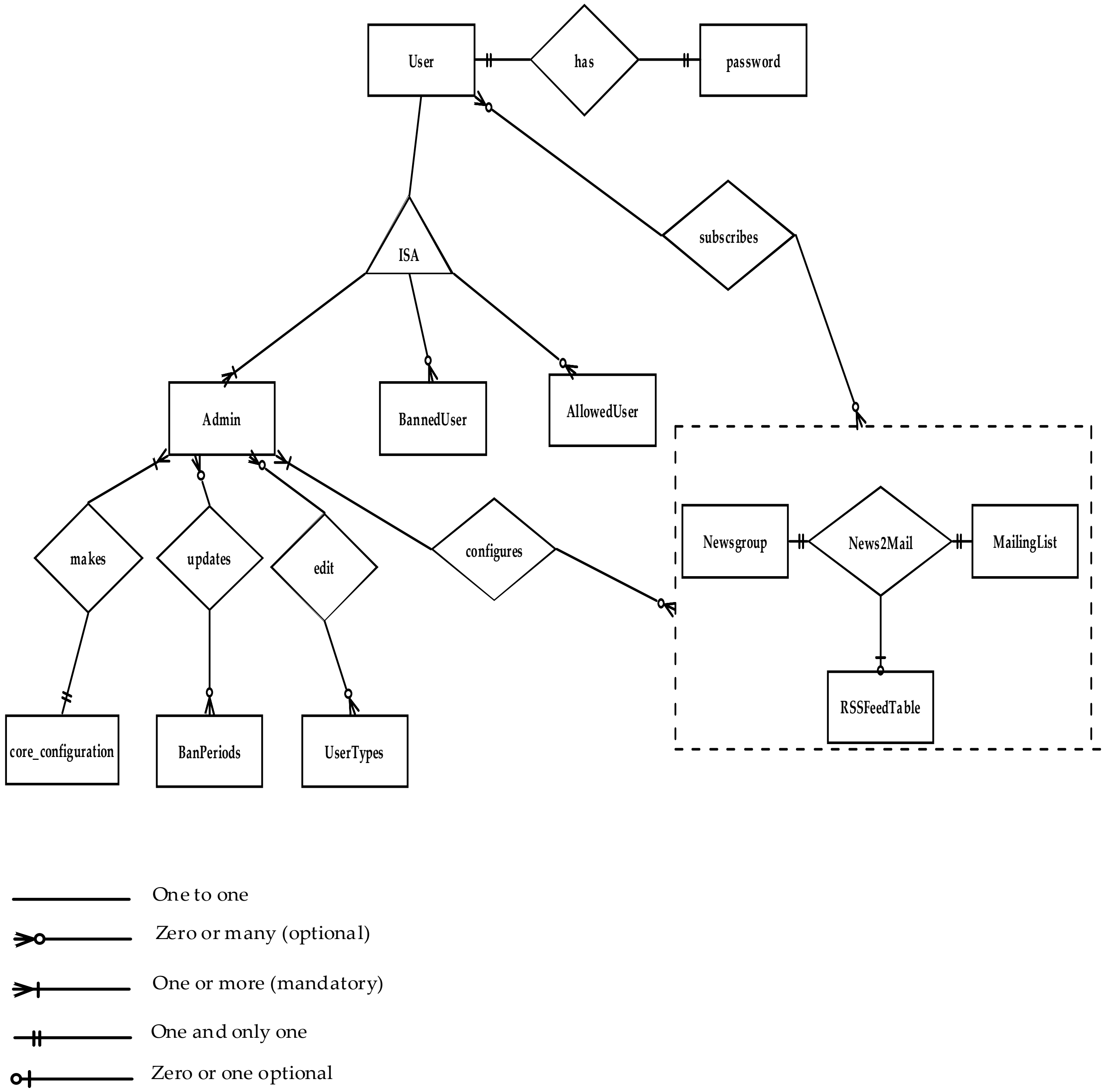


Figure 11: ER Diagram

4.3 Data Dictionary

4.3.1 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 one of the user types defined in UserTypes entity

4.3.2 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.

4.3.3 BannedUser

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

4.3.4 AllowedUser

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

4.3.5 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.

4.3.6 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.

4.3.7 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.

4.3.8 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.

4.3.9 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.

4.3.10 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.

4.4 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,

      constraint banneduser_uq unique(user_id, newsgroup_id)

);



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 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
);
```

5 Component Level Design

5.1 *Use Case Diagrams*

5.1.1 User

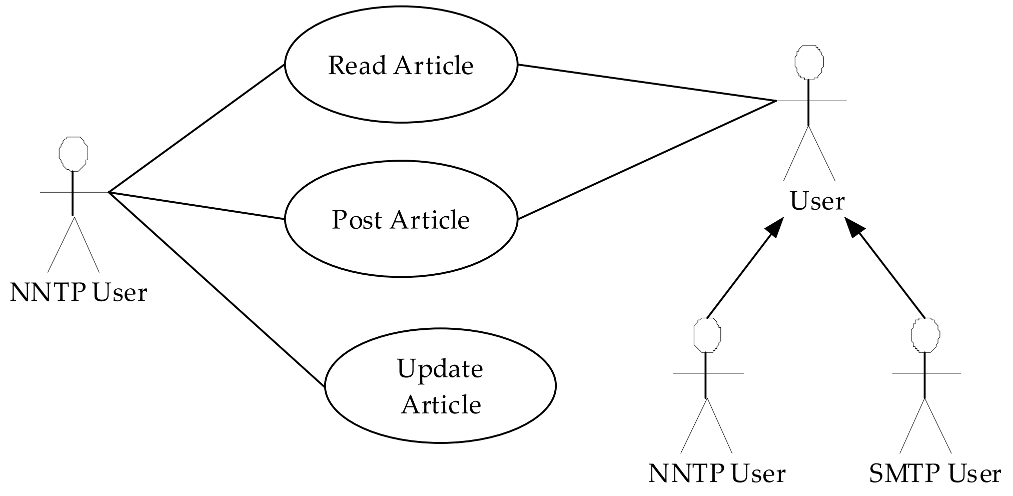


Figure 12: NNTP User

Our system treats NNTP and SMTP users in a different way since they are directly connecting to the core part of GÜVERCİN. The basic operations that they can do are reading and posting articles. Different from an SMTP client, an NNTP client may update the articles that he/she posted. In such a situation, the system sends the corrected article as mail to SMTP clients.

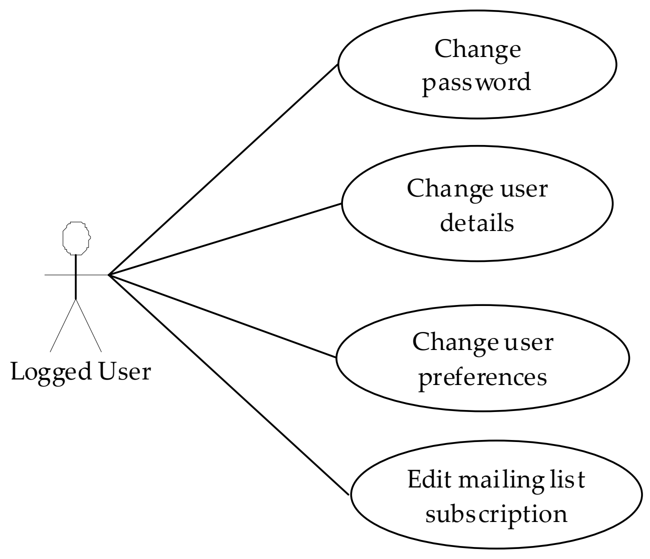
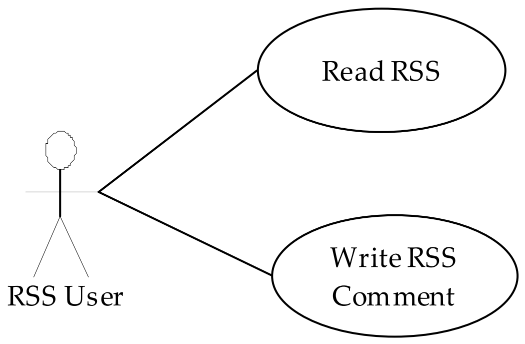


Figure 13: Logged User

A logged user may change the details of his/her membership information. The user uses the web interface for these operations. The operations that the user do with his/her membership options can be listed as follows:

- Changing login password
- Editing user details like name, surname, address info etc.
- Changing user preferences
- Editing mailing list preferences like mail receiving method(separate mails, daily digest or none), group subscription preferences, etc.



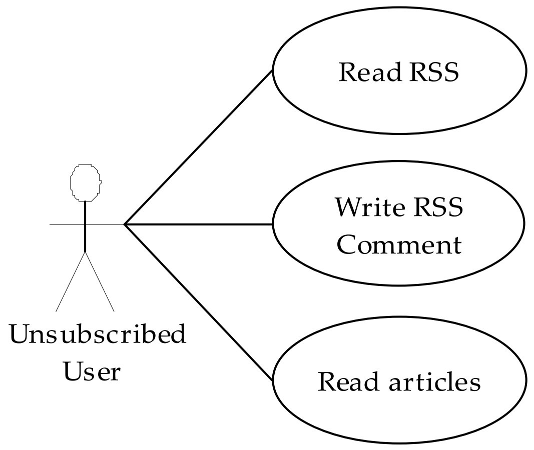


Figure 14: RSS User

Figure 15: Unsubscribed User

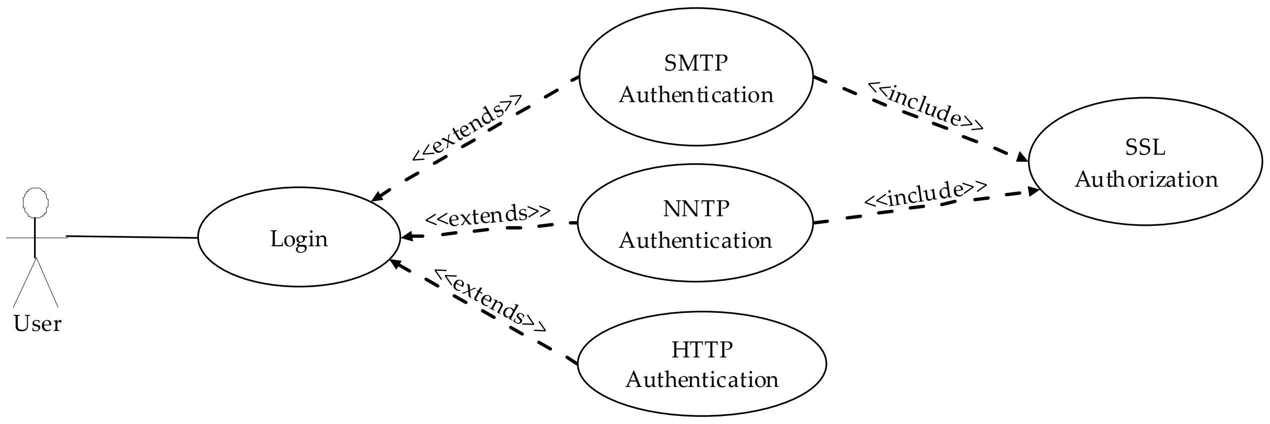


Figure 16: User

An unsubscribed user is like a 'guest' user in subscription required systems. It may only view the articles of allowed newsgroups by using the web interface and is not allowed the post threads or replies to the newsgroups. In addition to this capability, he/she may get the contents of newsgroups which have RSS capability and post comments on these articles.

A subscribed user can login to the system in three different ways: via news reader, e-mail account or web-interface. While NNTP and SMTP authentication requires SSL authorization, there is no such mechanism for HTTP authentication.

5.1.2 Admin

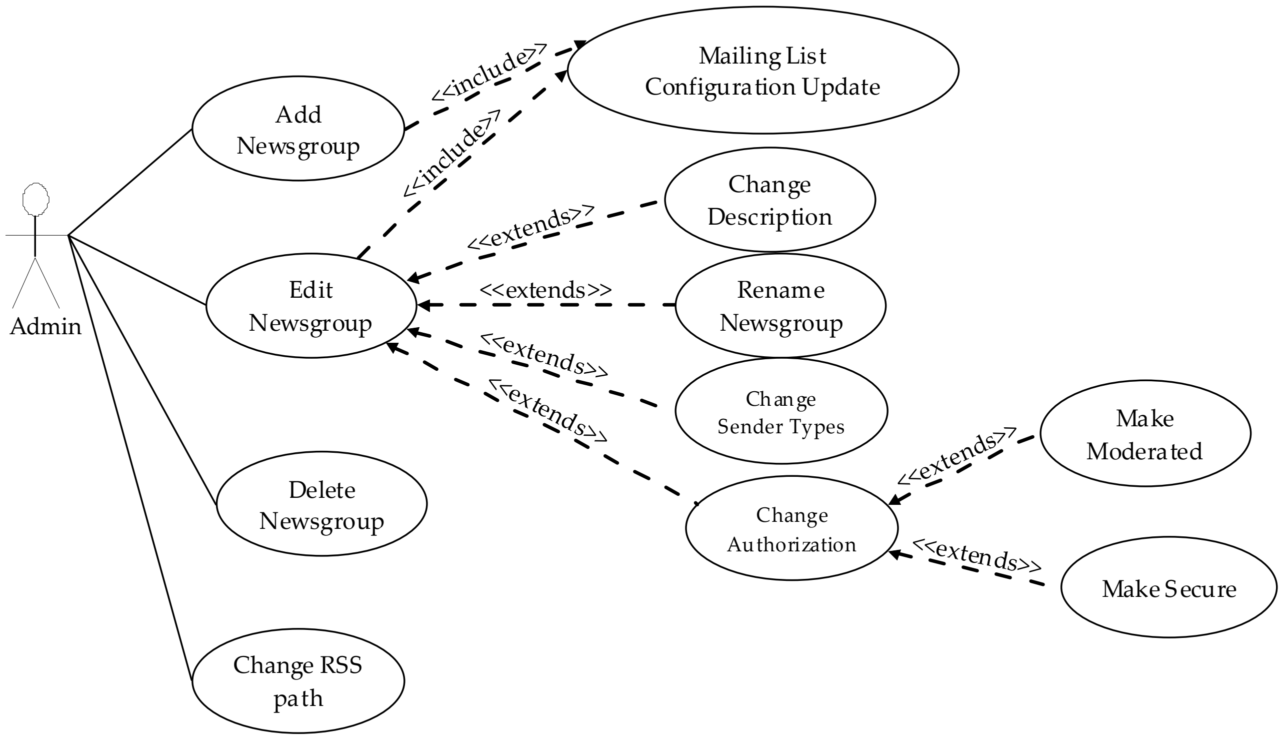


Figure 17: Newsgroup Configuration

As all other server systems, GÜVERCİN has a administrator who can configure the whole system. The operations that the administrator can perform on newsgroups are listed in four groups:

- *Adding a newsgroup to the system*: The e-mailing list that corresponds to this group is also created.

- *Editing an existing newsgroup*: The administrator can change the quote which describes the group, rename the group, change sender types and edit the authorization level. By 'change sender types' we mean that some groups may belong to specific user groups.

- *Deleting a newsgroup* : When the deletion operation is applied to a newsgroup, it also affects the corresponding mailing-list.

- *Changing RSS Path* : The administrator can change the common RSS path for the HTTP server.

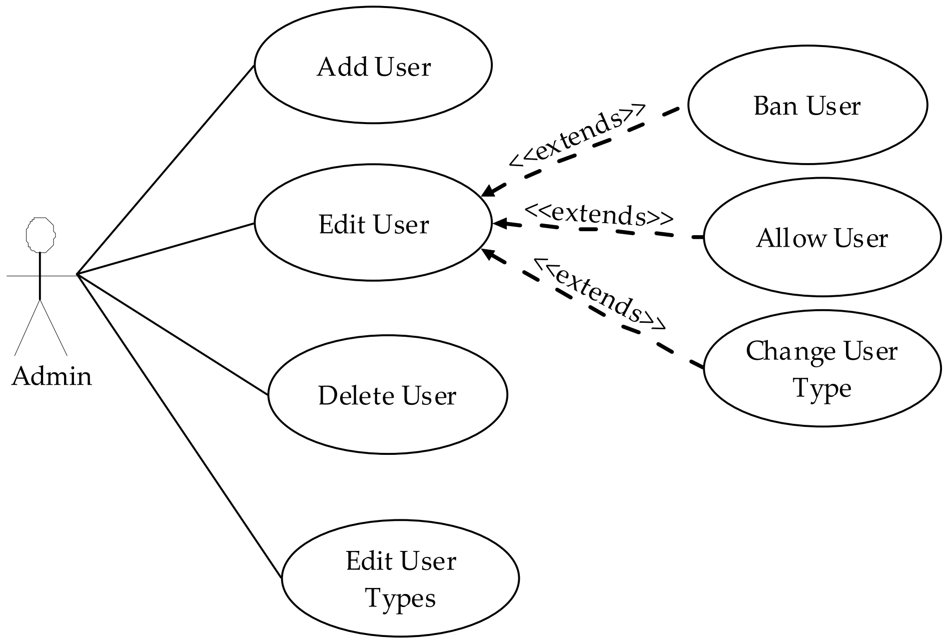


Figure 18: Users Configuration

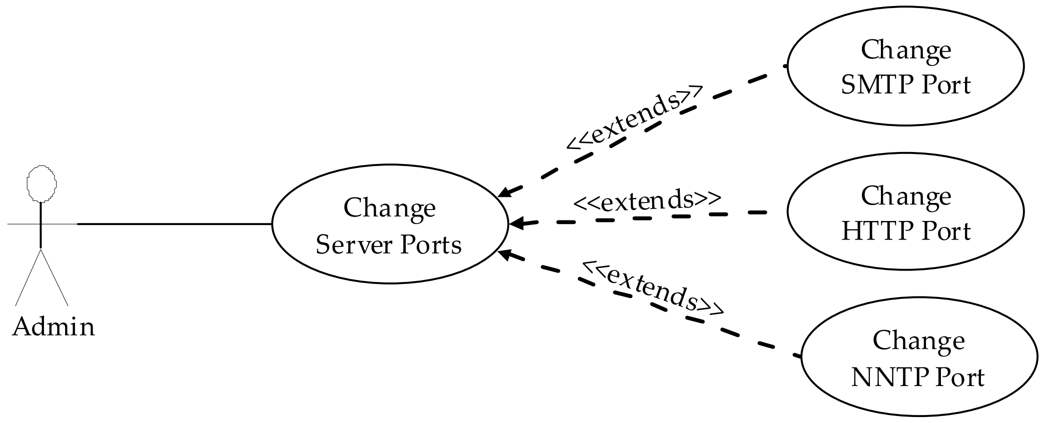


Figure 19: System Configuration

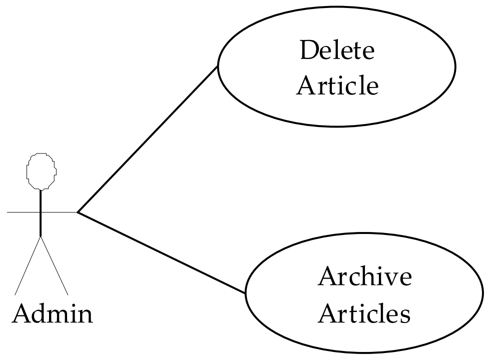
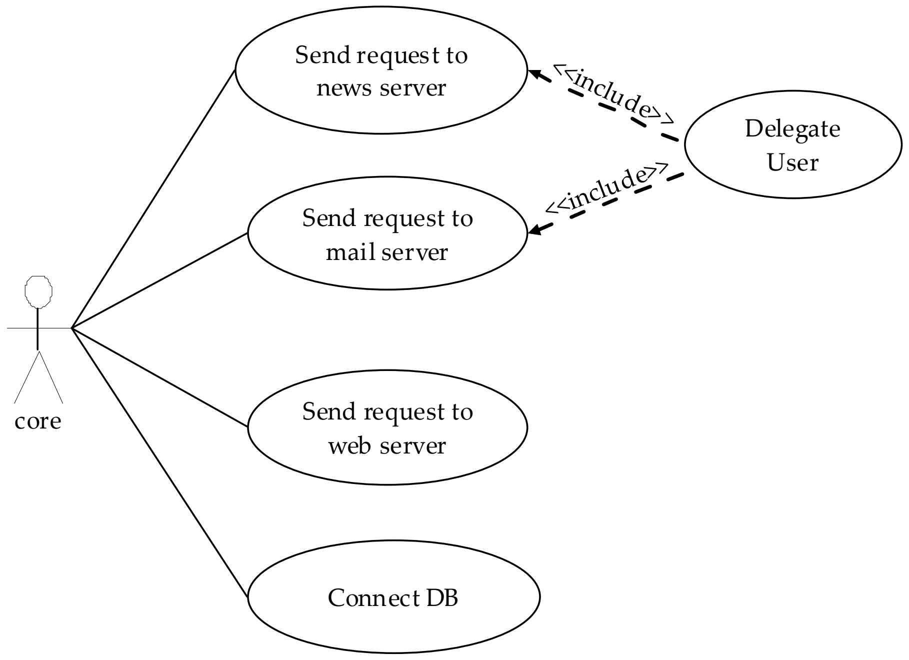


Figure 20: Article Management

In addition to the user operations, the administrator can configure the system by changing the port numbers of the built-in servers. This capability may be needed for security. The last two operations that the administrator can do on the system are deleting and archiving articles. Since administrator is also like a moderator for all newsgroups, he/she may delete some articles which contain harmful or undesired information. Archiving is also an important feature for our system. In addition to auto-archiving, the administrator may archive the articles manually.

5.1.3 Core



Send request to news server
<<include>>
Delegate User
<<include>>
Send request to mail server
core
Send request to web server
Connect DB

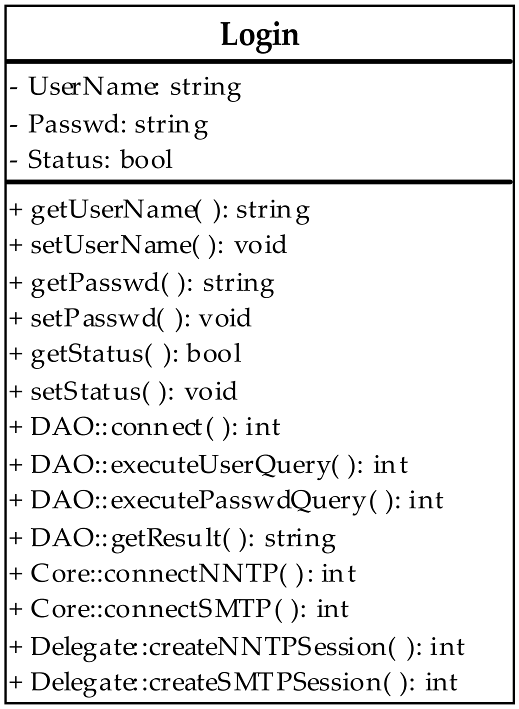
Figure 21: Core

Since the core part delegates the users' requests, it is also like a user for the built-in servers in our system. Basically we can explain the task of core part for the user sight as it just takes the command from/to user or one of built-in server and transfers it to the other one. Besides it is in contact with the database for some applications like retrieving user data, querying for the newsgroup-mailing list correspondence etc. A connection between the core and web server is needed for RSS operations and web-based news users.

5.2 Class Diagrams

5.2.1 Login Module

5.2.1.1 Login


Login
- UserName: string
- Passwd: string
- Status: bool
+ getUserName( ): string
+ setUserName( ): void
+ getPasswd( ): string
+ setPasswd( ): void
+ getStatus( ): bool
+ setStatus( ): void
+ DAO::connect( ): int
+ DAO::executeUserQuery( ): int
+ DAO::executePasswdQuery( ): int
+ DAO::getResult( ): string
+ Core::connectNNTP( ): int
+ Core::connectSMTP( ): int
+ Delegate::createNNTPSession( ): int
+ Delegate::createSMTPSession( ): int

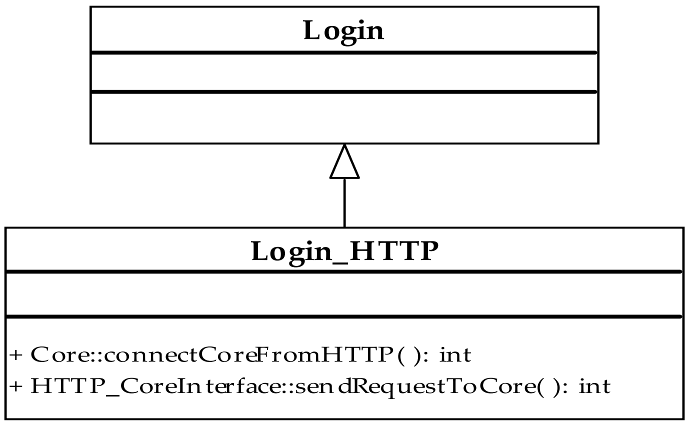
DAO::executeUserQuery( ): get the username from the database for creating the session on behalf of the user.

DAO::executePasswdQuery( ): get the password from the database for creating the session on behalf of the user.

Delegate::createNNTPSession(): create the session between the core and NNTP server for user delegation.

Delegate::createSMTPSession(): create the session between the core and SMTP server for user delegation.

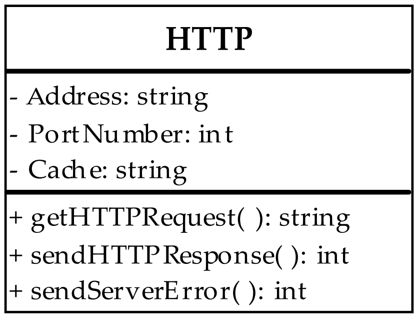
5.2.1.2 Login_HTTP


Login
Login_HTTP
+ Core::connectCoreFromHTTP( ): int
+ HTTP_CoreInterface::sendRequestToCore( ): int

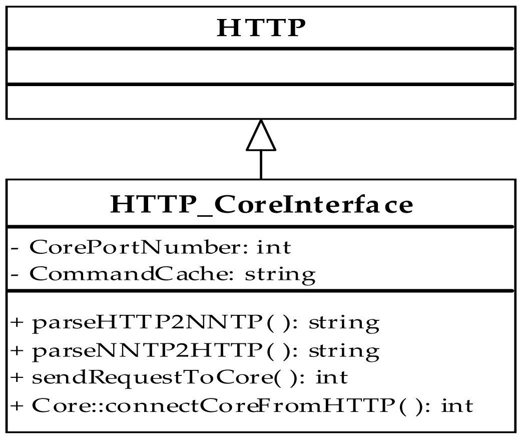
HTTP_CoreInterface::sendRequestToCore(): Invoke the core for creating the user's session.

5.2.2 HTTP Module

5.2.2.1 HTTP


HTTP
- Address: string
- PortNumber: int
- Cache: string
+ getHTTPRequest( ): string
+ sendHTTPResponse( ): int
+ sendServerError( ): int

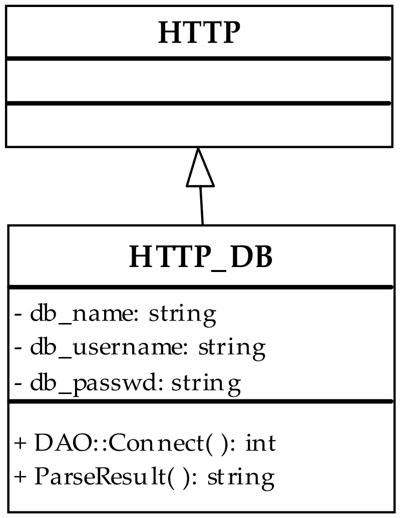
5.2.2.2 HTTP_CoreInterface


HTTP
HTTP_CoreInterface
- CorePortNumber: int
- CommandCache: string
+ parseHTTP2NNTP( ): string
+ parseNNTP2HTTP( ): string
+ sendRequestToCore( ): int
+ Core::connectCoreFromHTTP( ): int

parseHTTP2NNTP(): get the web news user's HTTP request and convert it to the related NNTP command in order to deliver to the core. The converted command is transferred to the core by using *sendRequestToCore()* function.

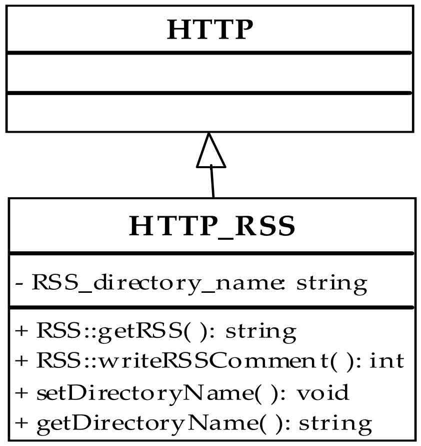
parseNNTP2HTTP(): get the NNTP response from the core and convert it to the related HTTP response in order to monitor it to the web news user.

5.2.2.3 HTTP_DB



DAO::Connect(): Connect to the database for configuration.

5.2.2.4 HTTP_RSS

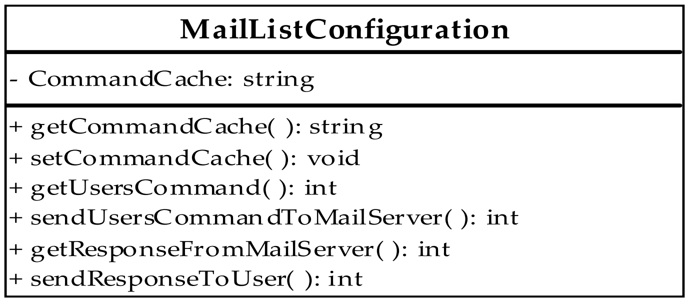

HTTP
HTTP_RSS
- RSS_directory_name: string
+ RSS::getRSS( ): string
+ RSS::writeRSSComment( ): int
+ setDirectoryName( ): void
+ getDirectoryName( ): string

RSS::getRSS(): get the RSS feed specified with the *RSS_directory_name* from the RSS data storage.

RSS::writeRSSComment(): write the RSS comment to the RSS file stored in the RSS file storage specified with the *RSS_directory_name*.

5.2.3 User Configuration Module

5.2.3.1 MailListConfiguration


MailListConfiguration
- CommandCache: string
+ getCommandCache( ): string
+ setCommandCache( ): void
+ getUsersCommand( ): int
+ sendUsersCommandToMailServer( ): int
+ getResponseFromMailServer( ): int
+ sendResponseToUser( ): int

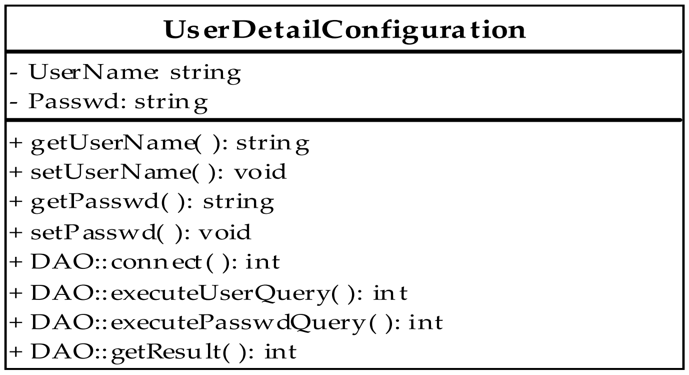
getUsersCommand(): get the users configuration command from the monitor.

sendUsersCommandToMailServer(): connect to the mailing list interface and deliver the configuration command.

getResponseFromMailServer(): get the response from the mailing list interface.

sendResponseToUser(): Deliver the feedback to the user.

5.2.3.2 UserDetailConfiguration



```
UserDetailConfiguration
- UserName: string
- Passwd: string
+ getUserName( ): string
+ setUserName( ): void
+ getPasswd( ): string
+ setPasswd( ): void
+ DAO::connect( ): int
+ DAO::executeUserQuery( ): int
+ DAO::executePasswdQuery( ): int
+ DAO::getResult( ): int
```

DAO:executeUserQuery(): execute the user's personal preference change request with connecting to the database.

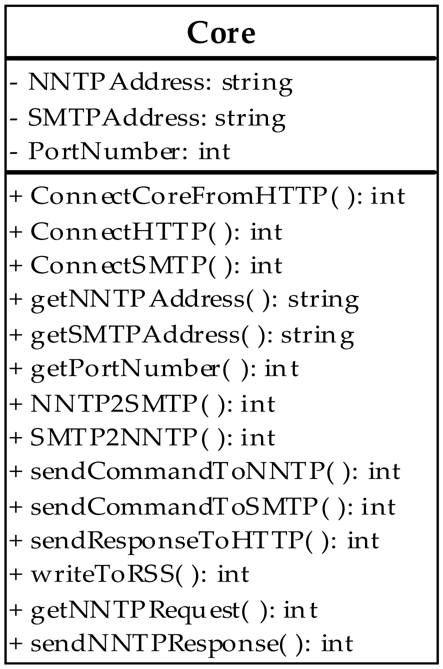
DAO:executePasswdQuery(): used for the authorization of the user.

DAO:getResult(): get the user's personal details in order show in the monitor.

DAO:connect(): connect to the database with username and password of the client.

5.2.4 Core Module

5.2.4.1 Core

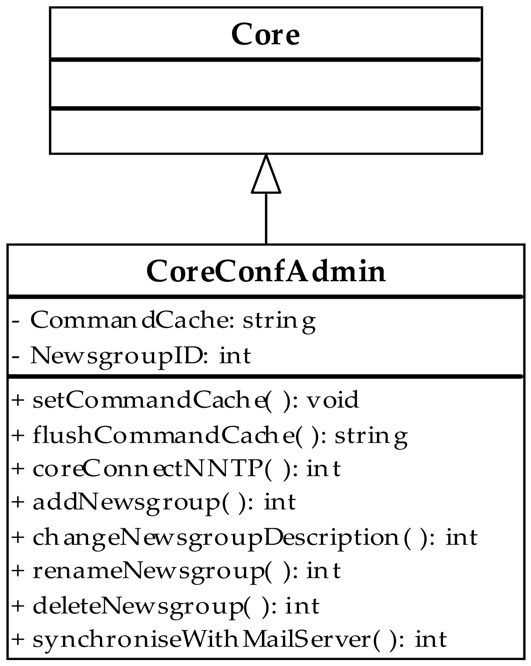


NNTP2SMTP(): Convert the user's NNTP command to the related SMTP command for synchronizing mail and news servers.

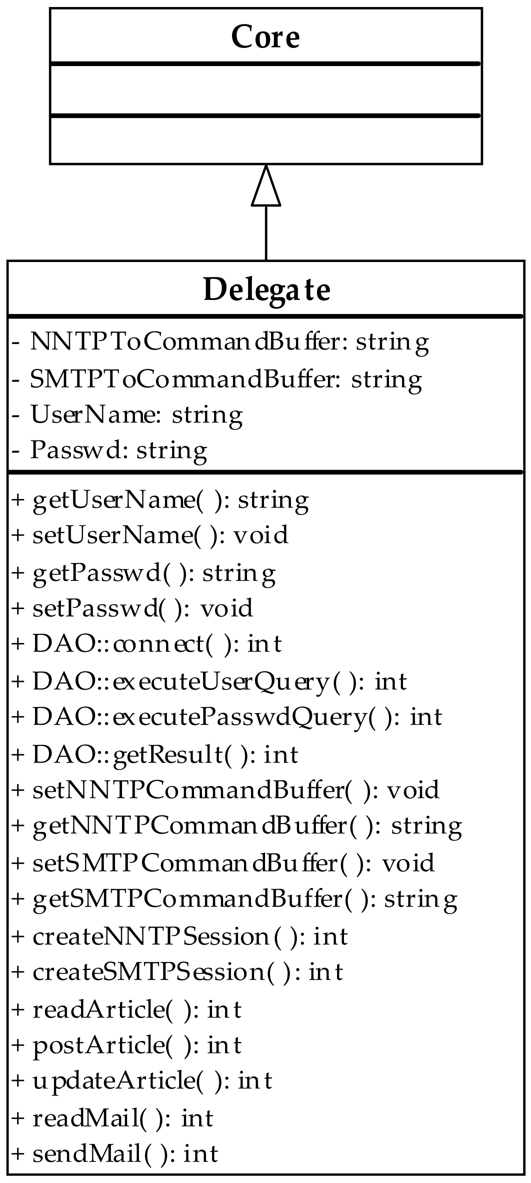
SMTP2NNTP(): Convert the user's SMTP command to the related NNTP command for synchronizing mail and news servers.

Both functions are used for synchronizing the newsgroups and corresponding mailing-lists. After a successful synchronization *writeToRSS()* function is used in case the newsgroup (or the mailing-list) has the RSS Feed option.

5.2.4.2 CoreConfAdmin


Core
CoreConfAdmin
- CommandCache: string
- NewsgroupID: int
+ setCommandCache( ): void
+ flushCommandCache( ): string
+ coreConnectNNTP( ): int
+ addNewsgroup( ): int
+ changeNewsgroupDescription( ): int
+ renameNewsgroup( ): int
+ deleteNewsgroup( ): int
+ synchroniseWithMailServer( ): int

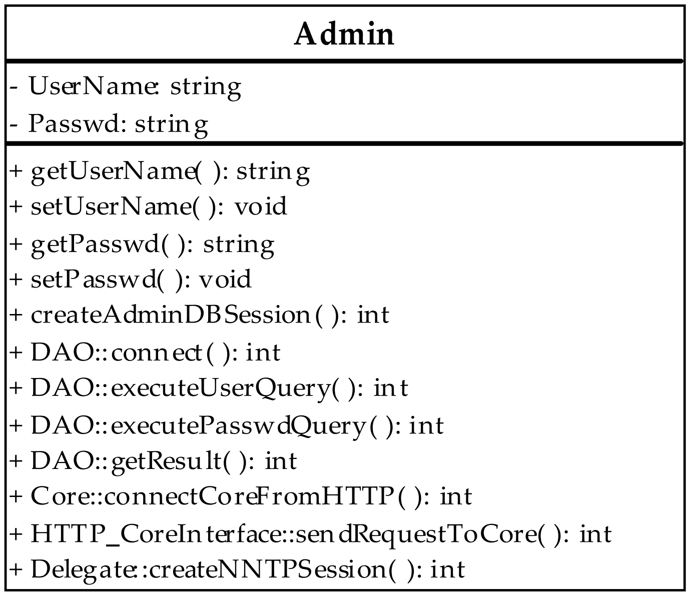
5.2.4.3 Delegate


Core
Delegate
- NNTPToCommandBuffer: string
- SMTPToCommandBuffer: string
- UserName: string
- Passwd: string
+ getUserName( ): string
+ setUserName( ): void
+ getPasswd( ): string
+ setPasswd( ): void
+ DAO::connect( ): int
+ DAO::executeUserQuery( ): int
+ DAO::executePasswdQuery( ): int
+ DAO::getResult( ): int
+ setNNTPCommandBuffer( ): void
+ getNNTPCommandBuffer( ): string
+ setSMTPCommandBuffer( ): void
+ getSMTPCommandBuffer( ): string
+ createNNTPSession( ): int
+ createSMTPSession( ): int
+ readArticle( ): int
+ postArticle( ): int
+ updateArticle( ): int
+ readMail( ): int
+ sendMail( ): int

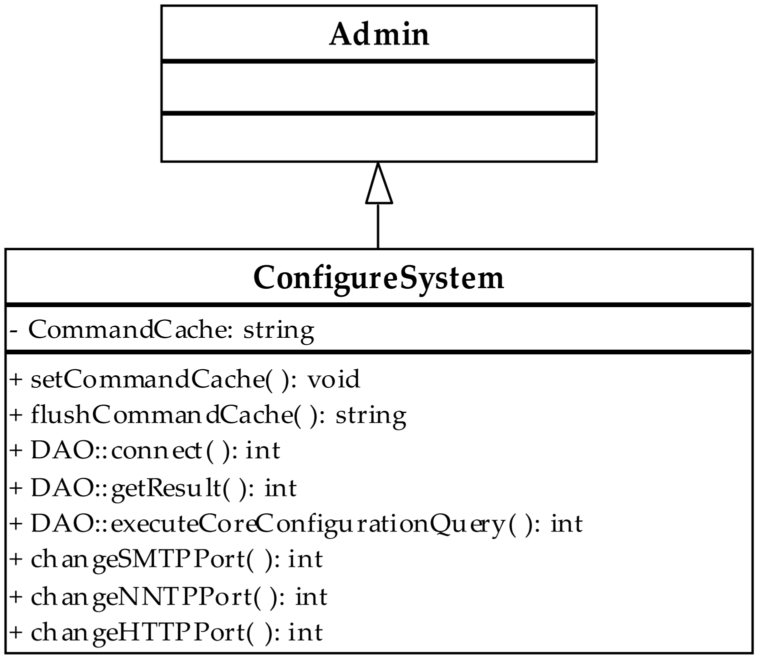
There are three main operations for NNTP users. For these operations *readArticle()*, *postArticle()* and *updateArticle()* functions are used. For the SMTP clients' side, *readMail()* and *sendMail()* functions are important. These operations must be delegated since the session is established between the core part and servers.

5.2.5 Admin Module

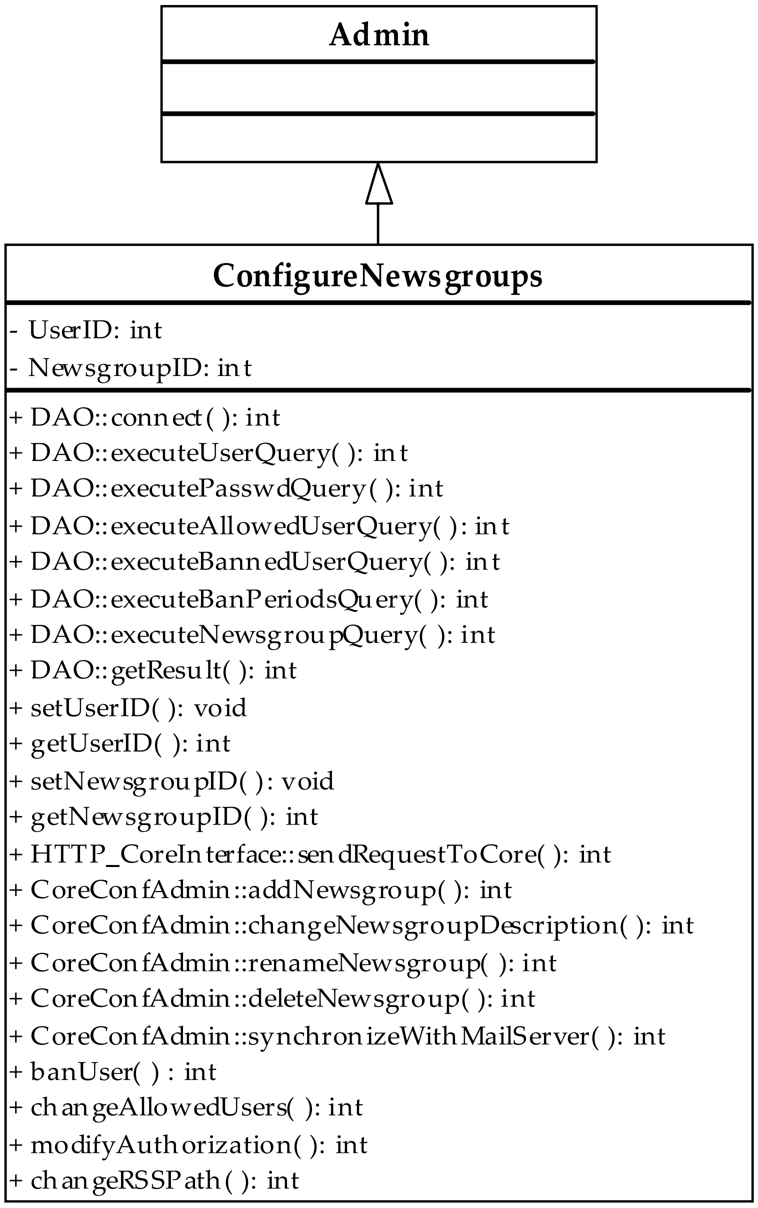
5.2.5.1 Admin


Admin
- UserName: string
- Passwd: string
+ getUserName( ): string
+ setUserName( ): void
+ getPasswd( ): string
+ setPasswd( ): void
+ createAdminDBSession( ): int
+ DAO::connect( ): int
+ DAO::executeUserQuery( ): int
+ DAO::executePasswdQuery( ): int
+ DAO::getResult( ): int
+ Core::connectCoreFromHTTP( ): int
+ HTTP_CoreInterface::sendRequestToCore( ): int
+ Delegate::createNNTPSession( ): int

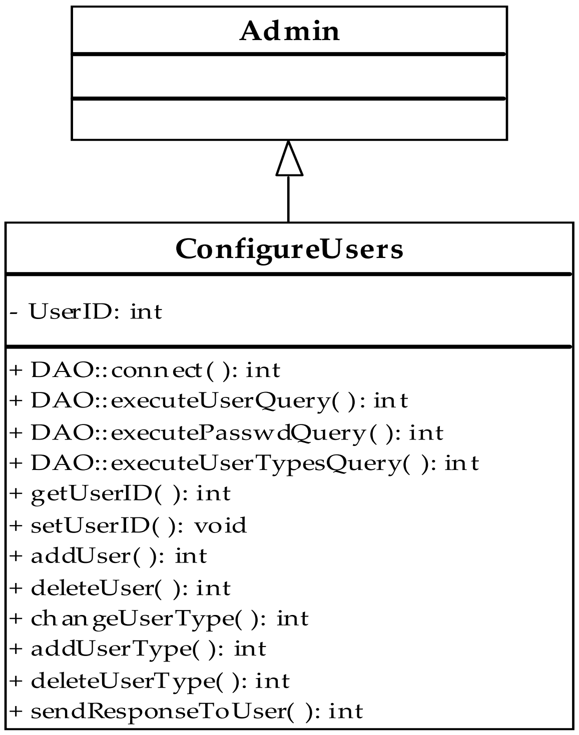
5.2.5.2 ConfigureSystem



5.2.5.3 ConfigureNewsgroups

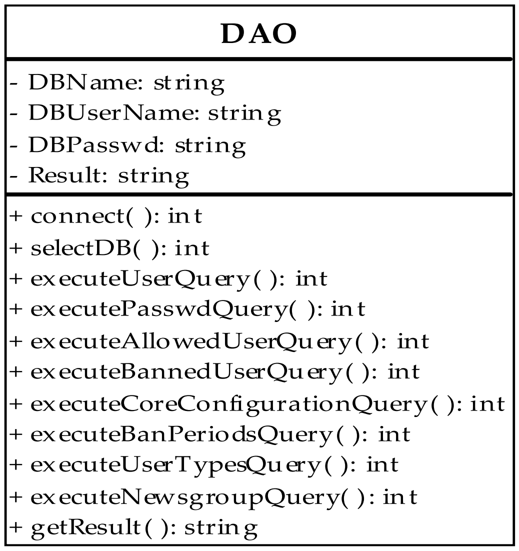

Admin
ConfigureNewsgroups
- UserID: int
- NewsgroupID: int
+ DAO::connect( ): int
+ DAO::executeUserQuery( ): int
+ DAO::executePasswdQuery( ): int
+ DAO::executeAllowedUserQuery( ): int
+ DAO::executeBannedUserQuery( ): int
+ DAO::executeBanPeriodsQuery( ): int
+ DAO::executeNewsgroupQuery( ): int
+ DAO::getResult( ): int
+ setUserID( ): void
+ getUserID( ): int
+ setNewsgroupID( ): void
+ getNewsgroupID( ): int
+ HTTP_CoreInterface::sendRequestToCore( ): int
+ CoreConfAdmin::addNewsgroup( ): int
+ CoreConfAdmin::changeNewsgroupDescription( ): int
+ CoreConfAdmin::renameNewsgroup( ): int
+ CoreConfAdmin::deleteNewsgroup( ): int
+ CoreConfAdmin::synchronizeWithMailServer( ): int
+ banUser( ) : int
+ changeAllowedUsers( ): int
+ modifyAuthorization( ): int
+ changeRSSPath( ): int

5.2.5.4 ConfigureUsers



5.2.6 DBLayer Module

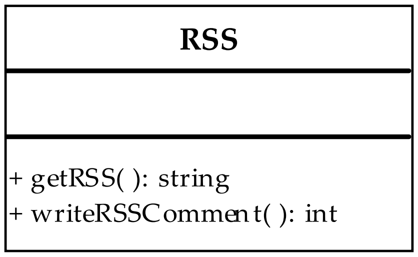
5.2.6.1 DAO


DAO
- DBName: string
- DBUserName: string
- DBPasswd: string
- Result: string
+ connect( ): int
+ selectDB( ): int
+ executeUserQuery( ): int
+ executePasswdQuery( ): int
+ executeAllowedUserQuery( ): int
+ executeBannedUserQuery( ): int
+ executeCoreConfigurationQuery( ): int
+ executeBanPeriodsQuery( ): int
+ executeUserTypesQuery( ): int
+ executeNewsgroupQuery( ): int
+ getResult( ): string

DAO module is a layer placed on the database of the system. Other classes use the functions of DAO class in order to interact with database.

5.2.7 RSS Module

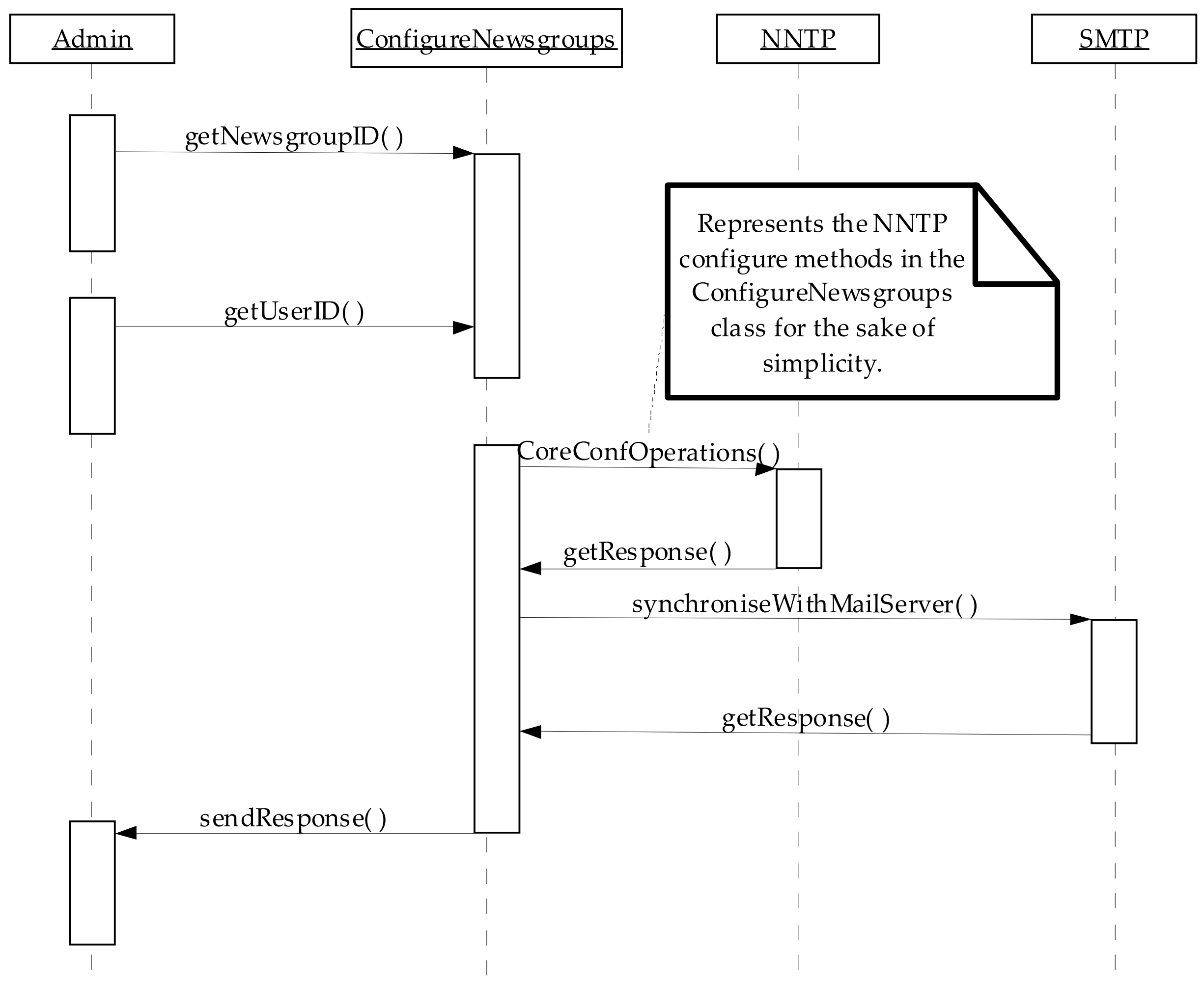
5.2.7.1 RSS



5.3 *Sequence Diagrams*

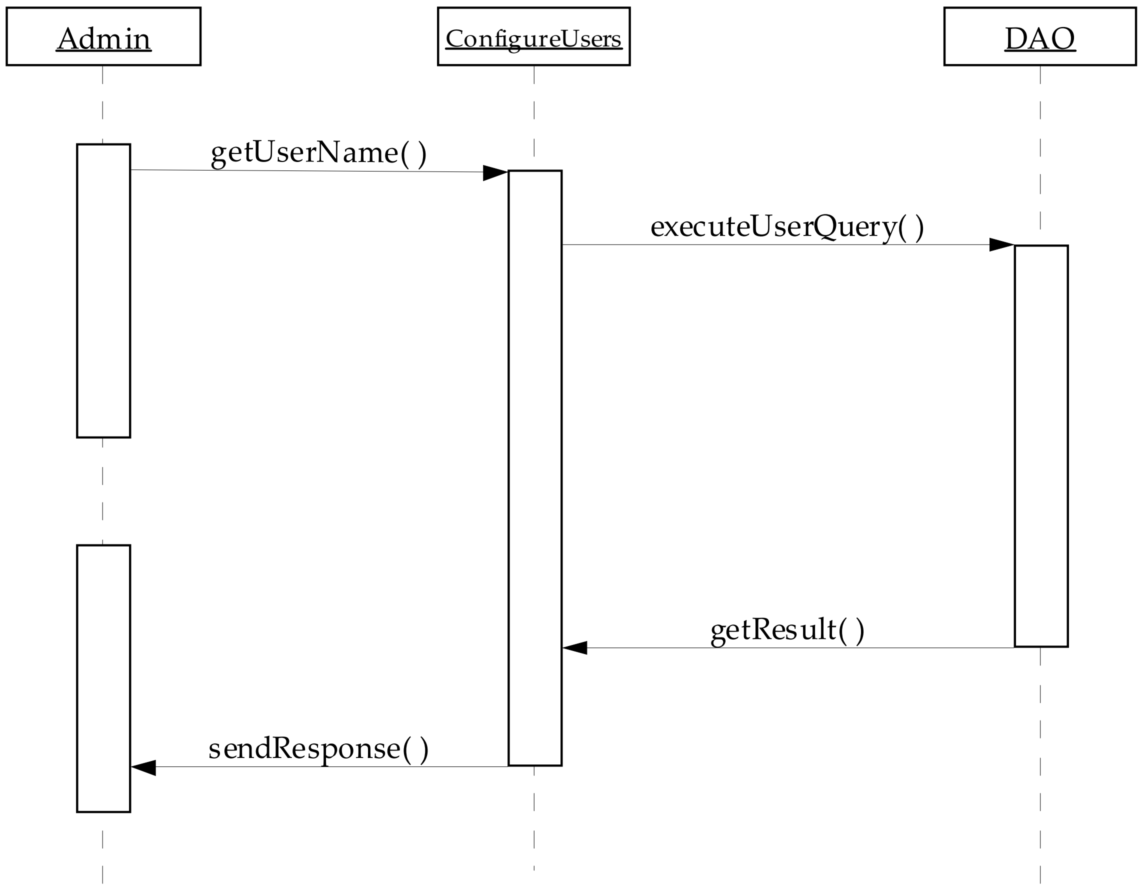
5.3.1 Admin

5.3.1.1 Newsgroup Configuration Process


Admin
ConfigureNewsgroups
NNTP
SMTP
getNewsgroupID( )
getUserID( )
Represents the NNTP configure methods in the ConfigureNewsgroups class for the sake of simplicity.
CoreConfOperations( )
getResponse( )
synchroniseWithMailServer( )
getResponse( )
sendResponse( )

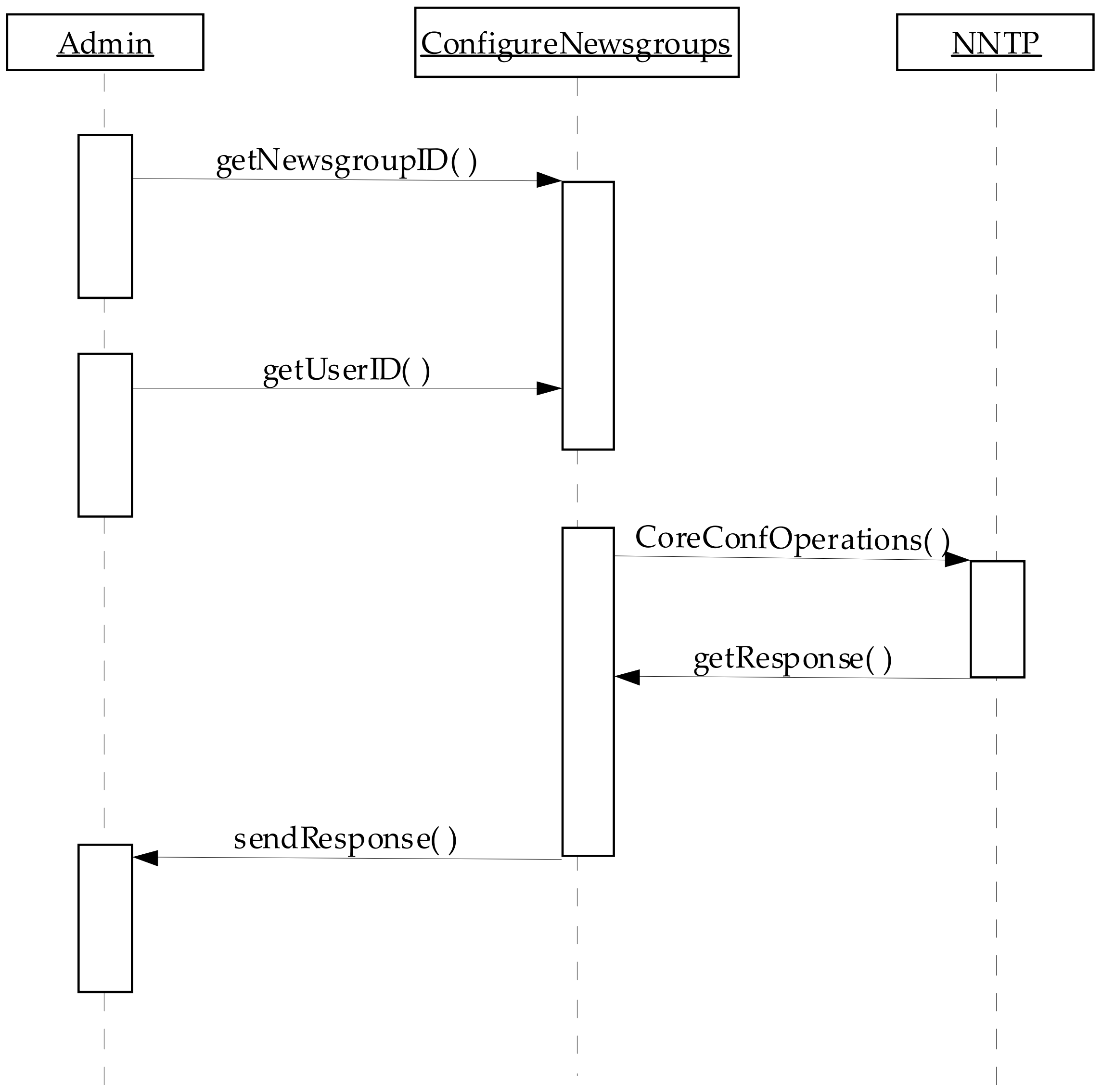
Administrator first gets the id of the newsgroup that will be configured. Then he/she gets the userID in case of a user addition or deletion operation on the newsgroup. The operation is first applied to the newsgroup, if it is successful the corresponding e-mailing list is also configured

5.3.1.2 User Management Process



Administrator first gets the username of the user which will be modified(add/delete), then the user query is executed.

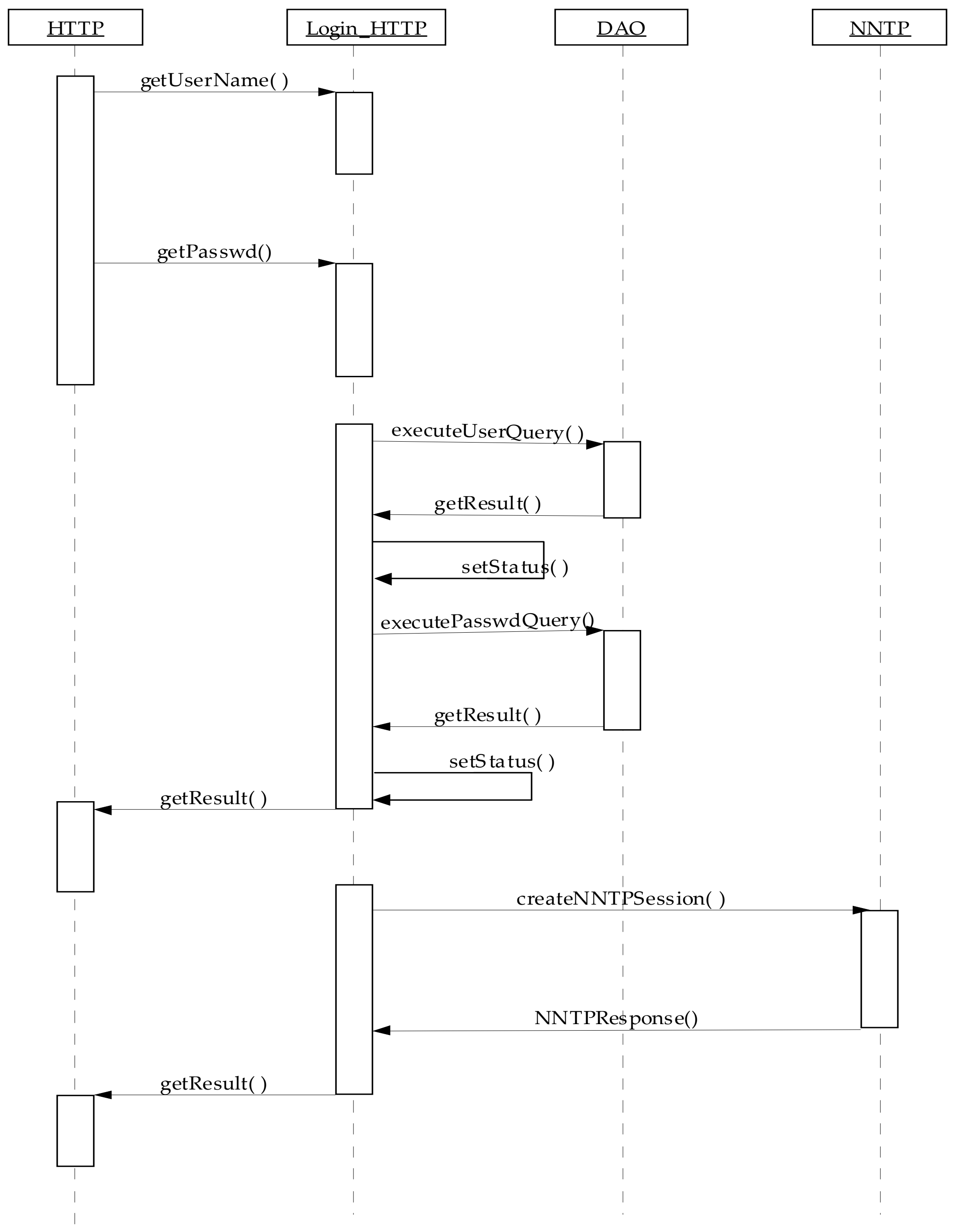
5.3.1.3 Newsgroup – User Configuration Process



For banning a user from a newsgroup or allowing a user for applying NNTP operations for a specific newsgroup, ids of the newsgroup and user is fetched.

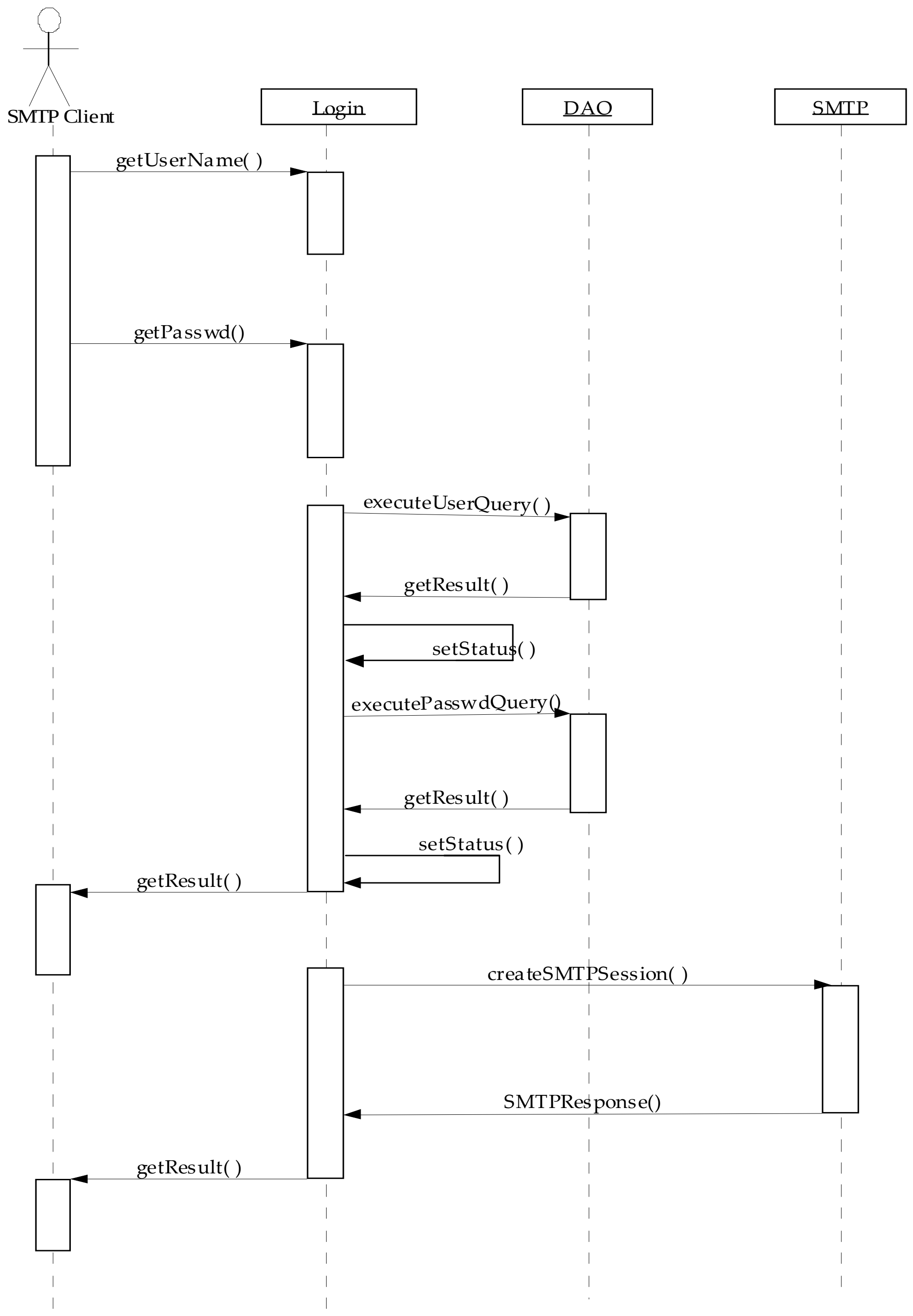
5.3.2 Login Process

5.3.2.1 HTTP Login Process

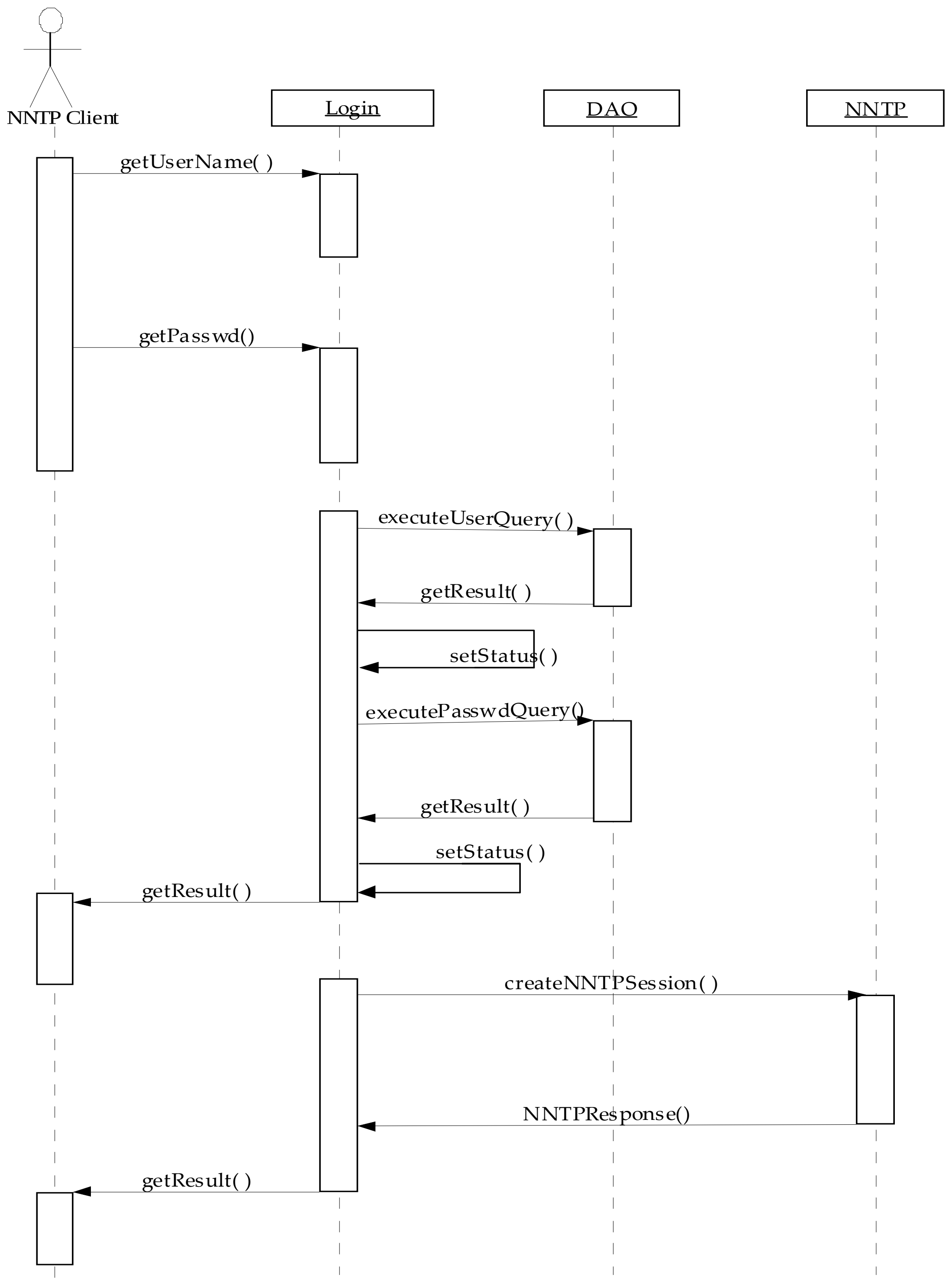


HTTP Login is some kind like a NNTP Login. The system first gets the username and password and checks if it is true. After this authorization ‘create NNTP session for the user’ command is delivered to the core module.

5.3.2.2 SMTP Login Process

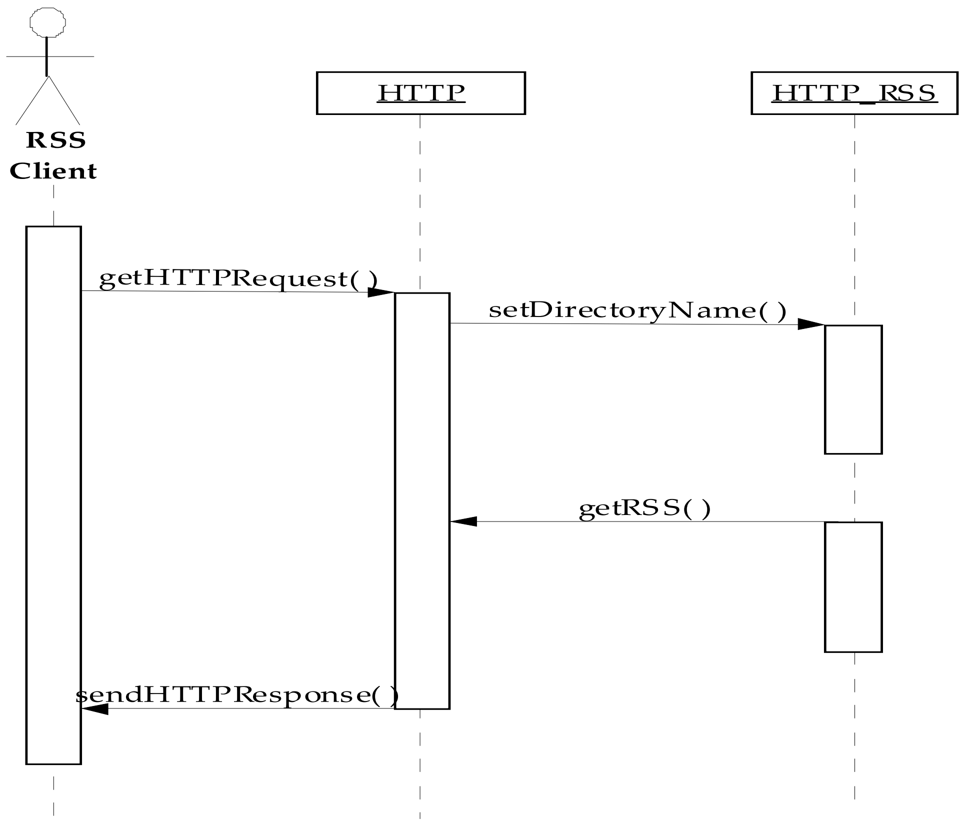


5.3.2.3 NNTP Login Process



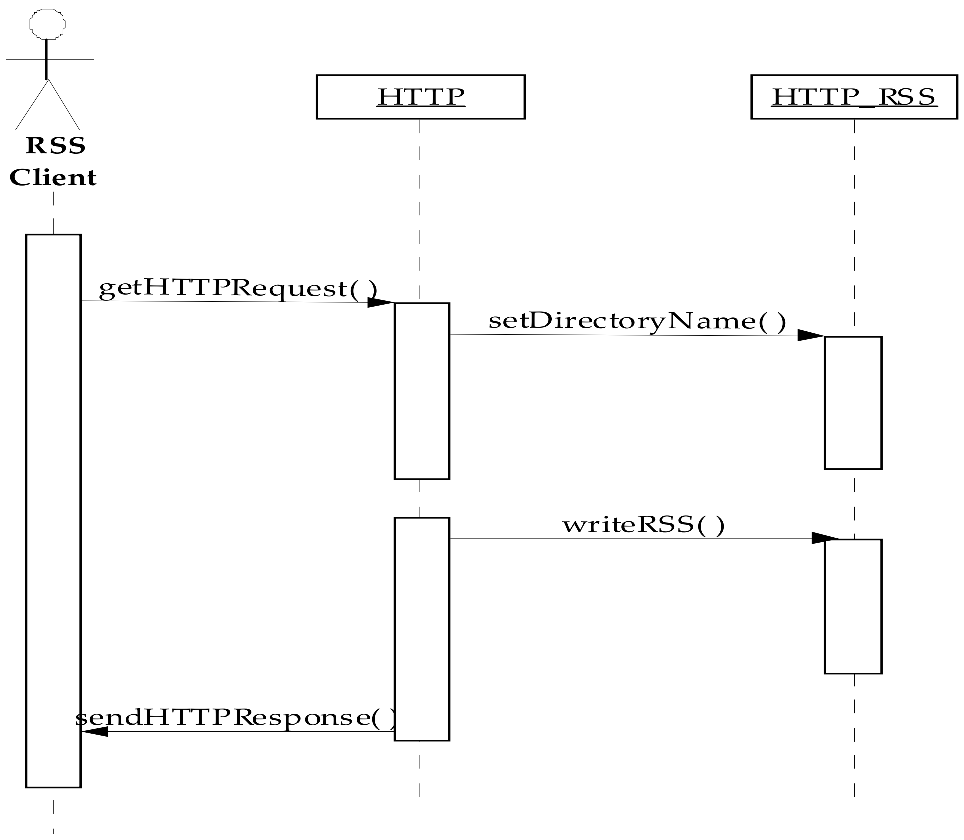
5.3.3 RSS

5.3.3.1 RSS Read Process



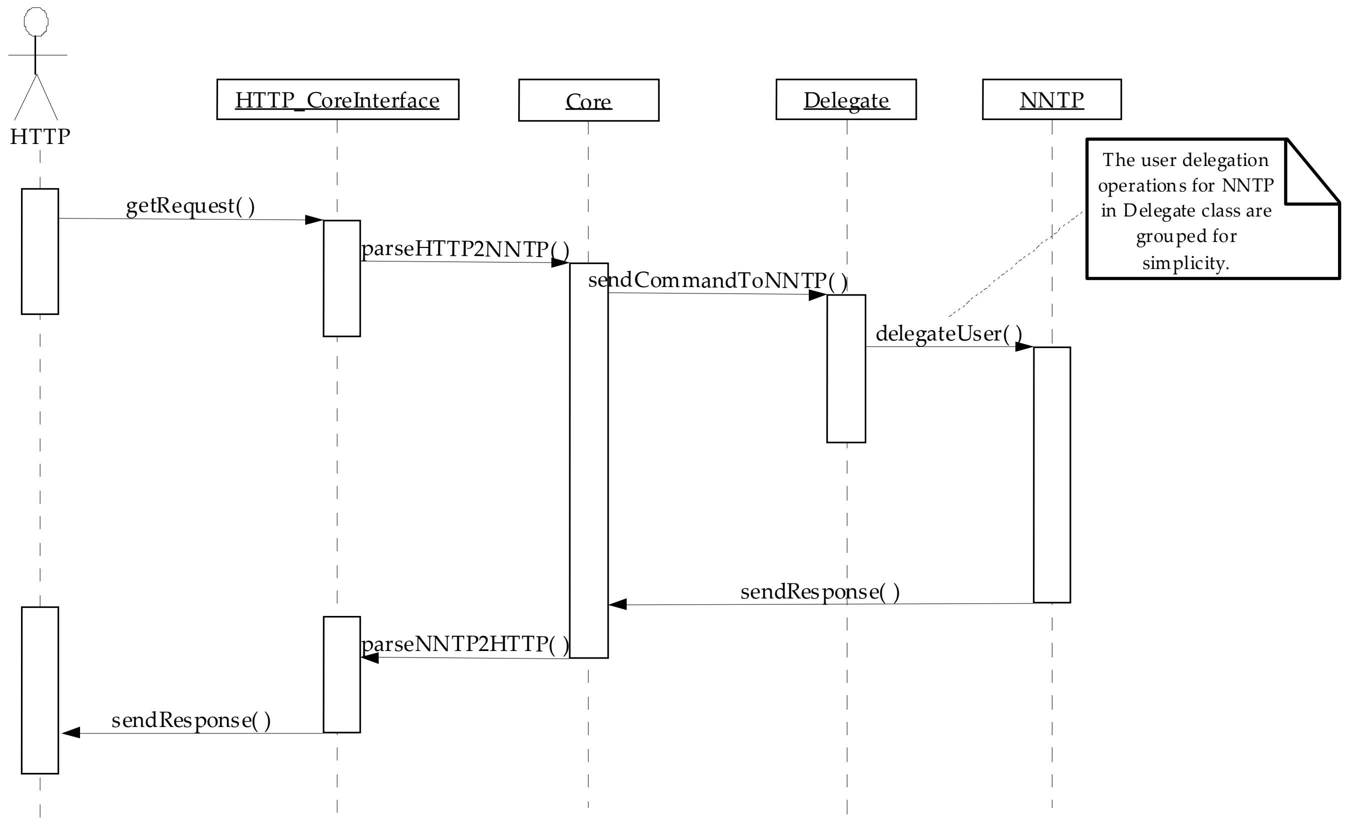
RSS Client just asks for the RSS feed file which corresponds to the URL that he/she specified. The HTTP module asks HTTP_RSS module for that file and returns the result to the user. Same procedure works for the 'RSS Write Process'.

5.3.3.2 RSS Write Process



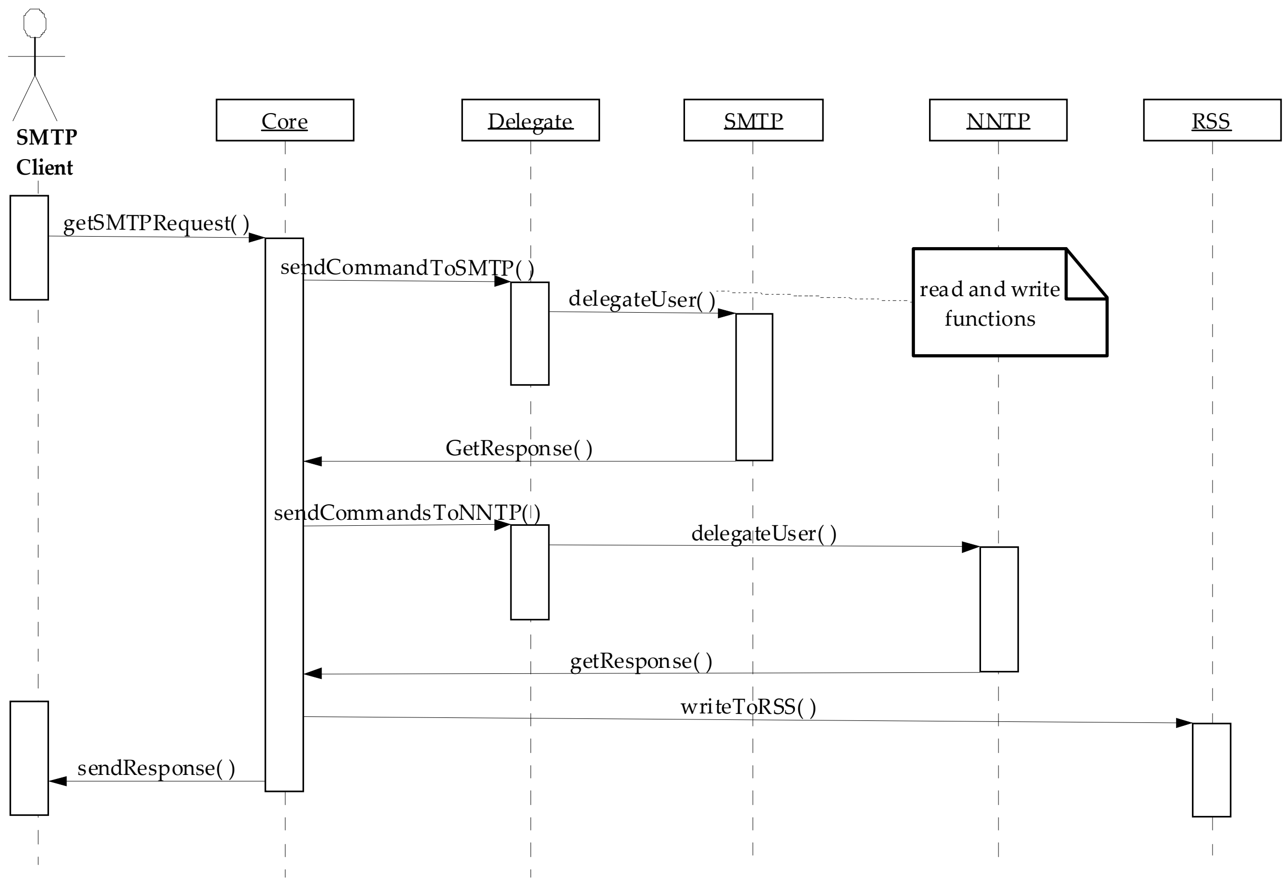
5.3.4 Logged User Access

5.3.4.1 HTTP Access Process

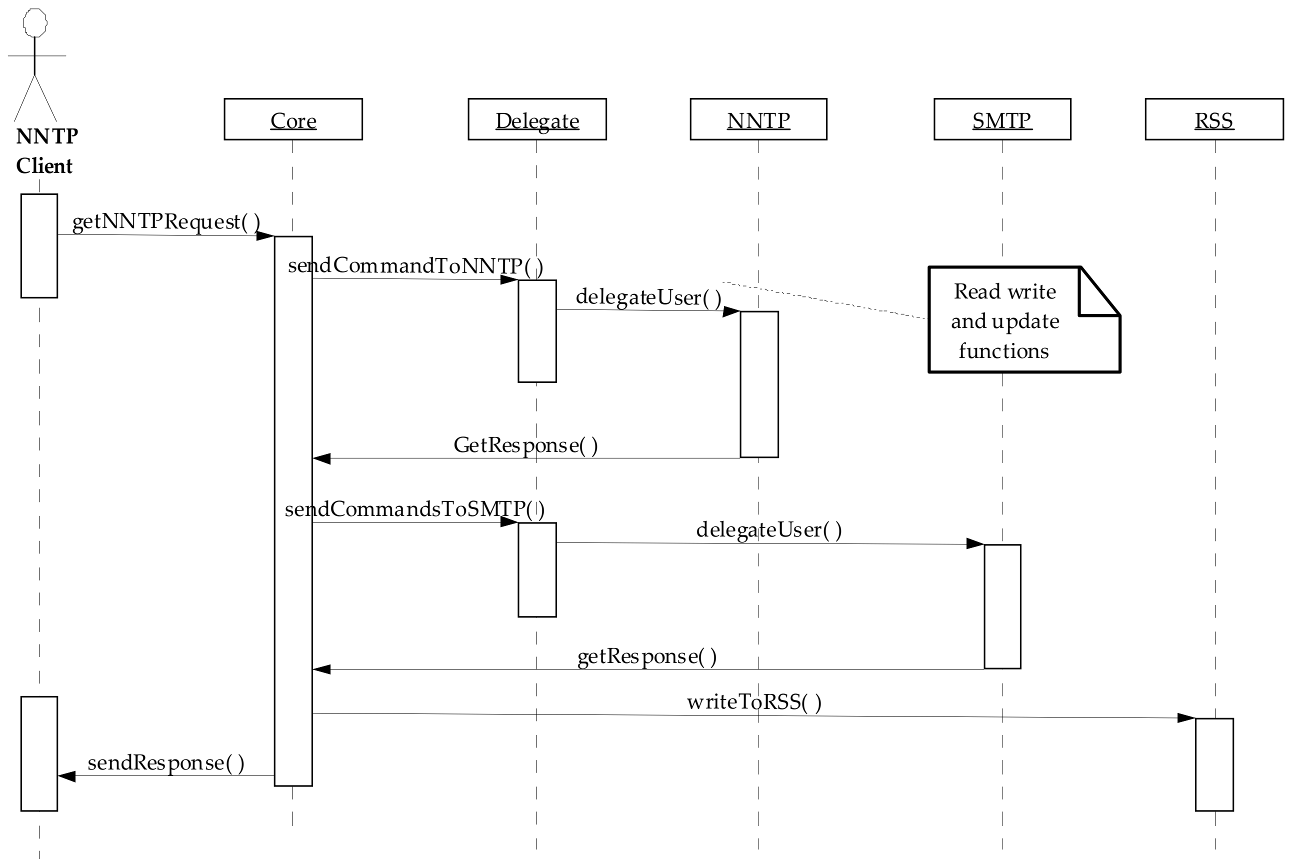


Commands of the logged HTTP clients are delivered to the core by converting them to the related NNTP commands. Core delegates the users' request and sends the response that came from the NNTP server to the HTTP server by converting the response to from NNTP to HTTP.

5.3.4.2 SMTP Access Process



5.3.4.3 NNTP Access Process



SMTP and NNTP access processes are the most vital operations for our system. Both commands are first delegated to the related server (if it is coming from NNTP, commands is delivered to news server) and if this operation succeeds same command is converted in order to delegate it to the other server (SMTP or NNTP server). If the newsgroup on which the operation is running has a RSS Feed option the RSS file of that newsgroup is created or updated.

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.

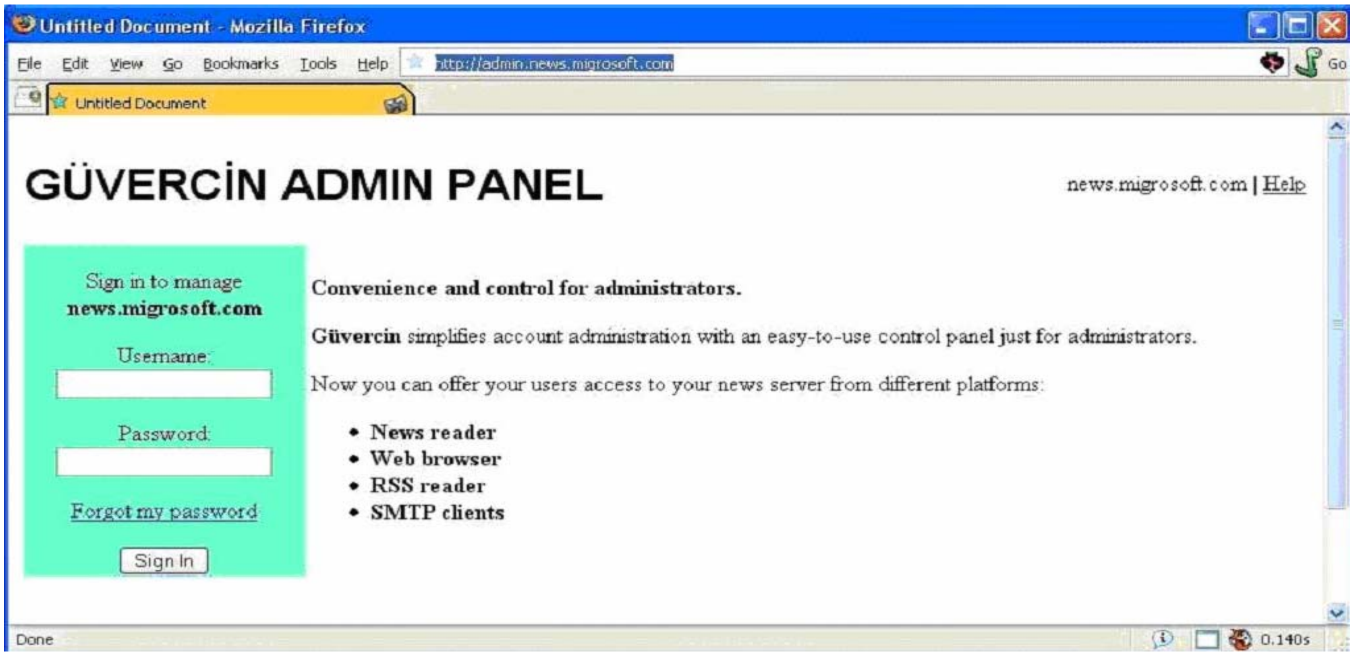
6.2.2 Admin Panel

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:


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://admin.news.migrosoft.com
Untitled Document
Done
0.140s

GÜVERCİN ADMIN PANEL

news.migrosoft.com | Help

Sign in to manage
news.migrosoft.com

Username:

Password:

Forgot my password

Sign In

Convenience and control for administrators.

Güvercin simplifies account administration with an easy-to-use control panel just for administrators.

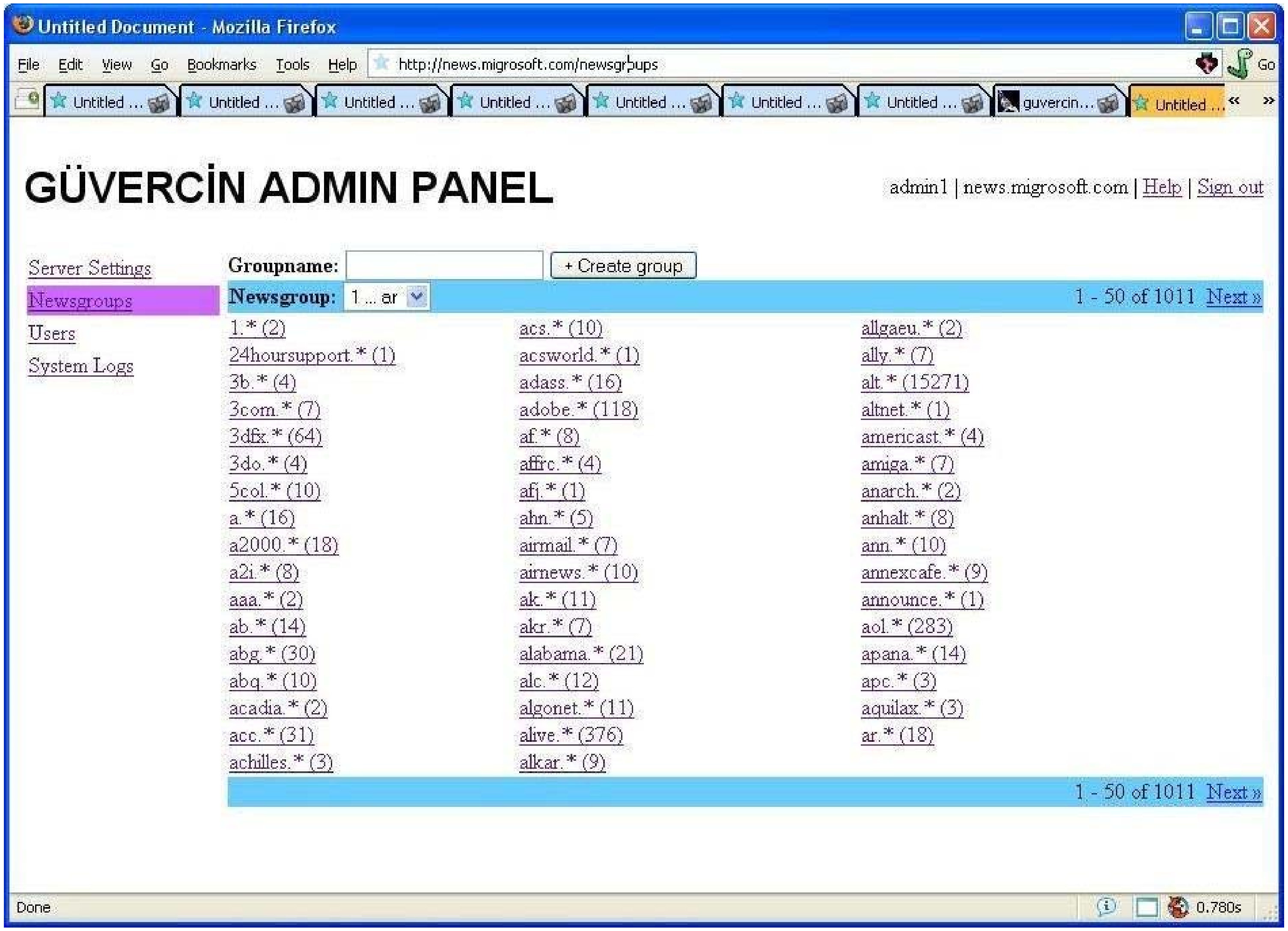
Now you can offer your users access to your news server from different platforms:

- **News reader**
- **Web browser**
- **RSS reader**
- **SMTP clients**

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



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)	acs.* (10)	allgaeu.* (2)
24hoursupport.* (1)	acsworld.* (1)	ally.* (7)
3b.* (4)	adass.* (16)	alt.* (15271)
3com.* (7)	adobe.* (118)	altnet.* (1)
3dfx.* (64)	af.* (8)	americast.* (4)
3do.* (4)	affrc.* (4)	amiga.* (7)
5col.* (10)	afj.* (1)	anarch.* (2)
a.* (16)	ahn.* (5)	anhalt.* (8)
a2000.* (18)	airmail.* (7)	ann.* (10)
a2i.* (8)	airnews.* (10)	annexcafe.* (9)
aaa.* (2)	ak.* (11)	announce.* (1)
ab.* (14)	akr.* (7)	aol.* (283)
abg.* (30)	alabama.* (21)	apana.* (14)
abq.* (10)	alc.* (12)	apc.* (3)
acadia.* (2)	algonet.* (11)	aquilax.* (3)
acc.* (31)	alive.* (376)	ar.* (18)
achilles.* (3)	alkar.* (9)	

1 - 50 of 1011 Next »

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.

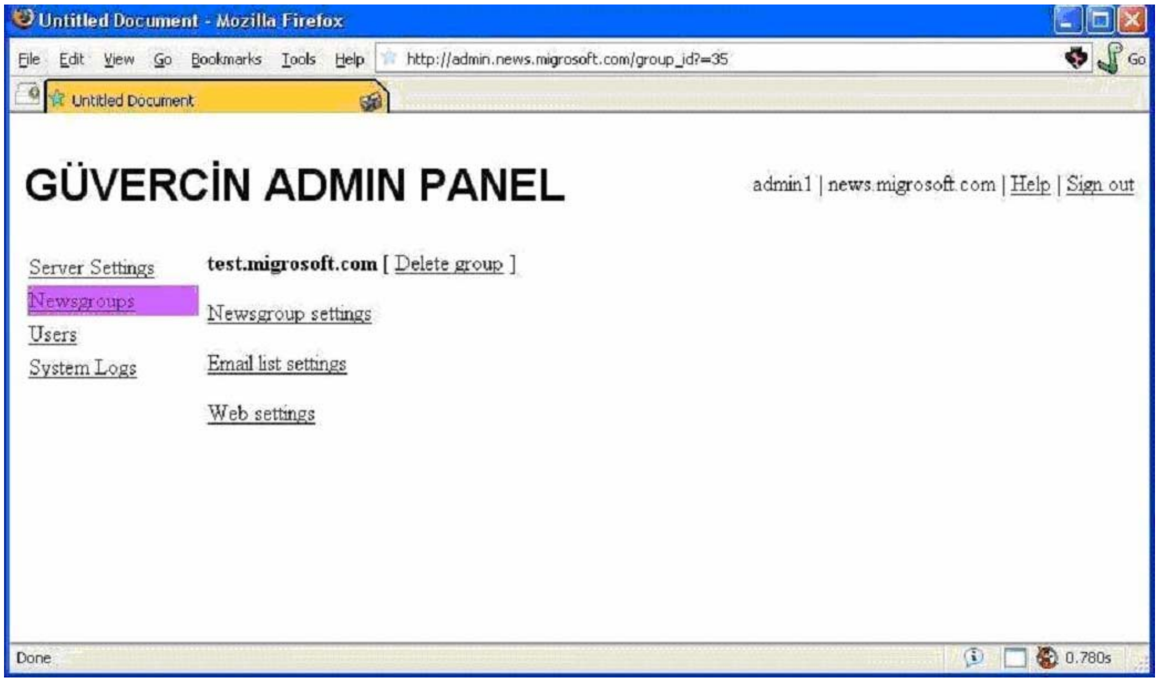
The user can see the newsgroups in an interval according to alphabetic order by the combo menu. The newsgroups are listed here in the group hierarchy. To illustrate the link "announce.*" refers to the group of groups

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:


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://admin.news.migrosoft.com/group_id?=35
Untitled Document
Done
0.780s

GÜVERCİN ADMIN PANEL

admin1 | news.migrosoft.com | Help | Sign out

Server Settings
Newsgroups
Users
System Logs

test.migrosoft.com [Delete group]

Newsgroup settings

Email list settings

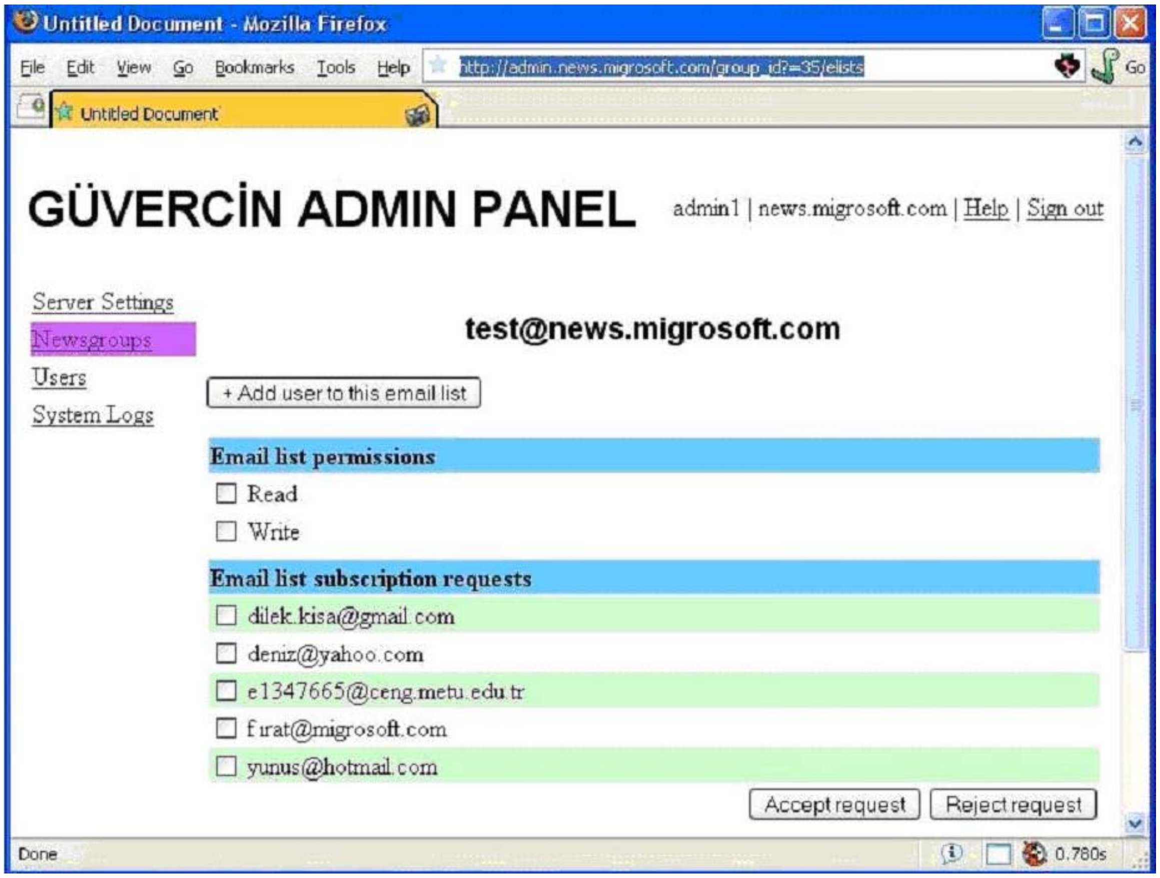
Web settings

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.


Untitled Document - Mozilla Firefox

http://admin.news.migrosoft.com/group_id?=35/elists

GÜVERCİN ADMIN PANEL

admin1 | news.migrosoft.com | Help | Sign out

Server Settings
Newsgroups
Users
System Logs

test@news.migrosoft.com

+ Add user to this email list

Email list permissions

☐ Read
☐ Write

Email list subscription requests

☐ dilek.kisa@gmail.com
☐ deniz@yahoo.com
☐ e1347665@ceng.metu.edu.tr
☐ firat@migrosoft.com
☐ yunus@hotmail.com

Accept request | Reject request

Done

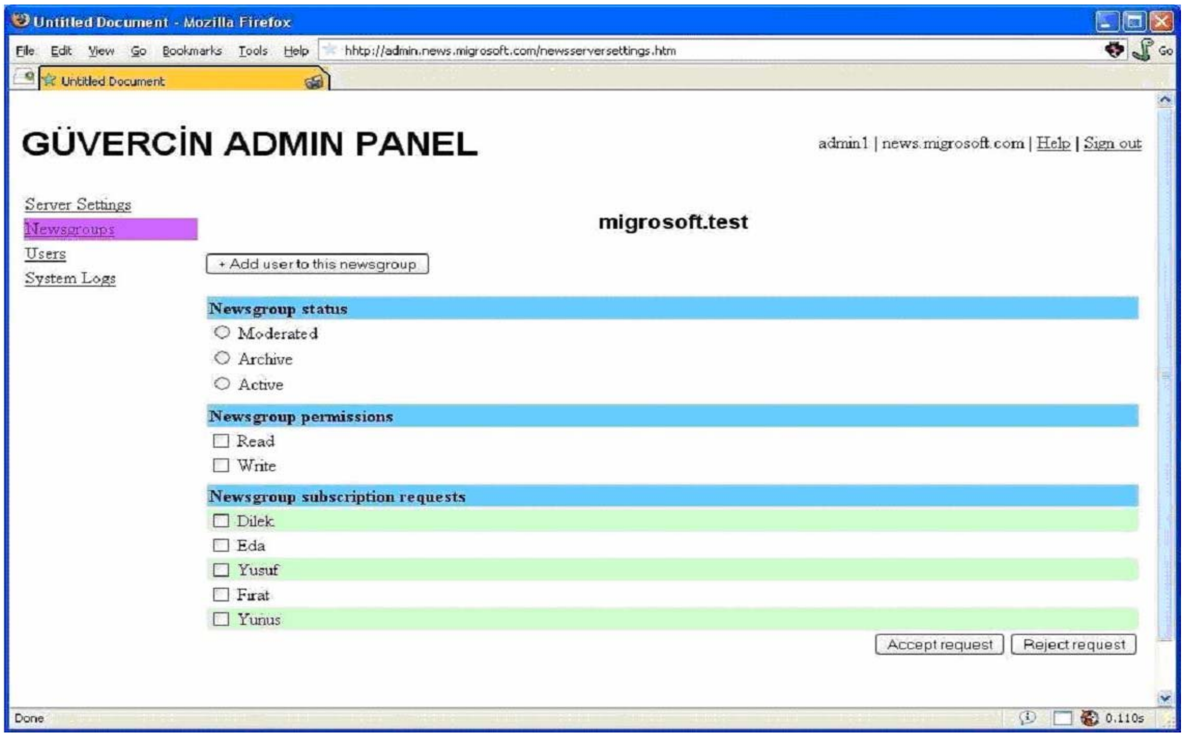
In the newsgroup settings page for a newsgroup the status is set one of the followings:

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.

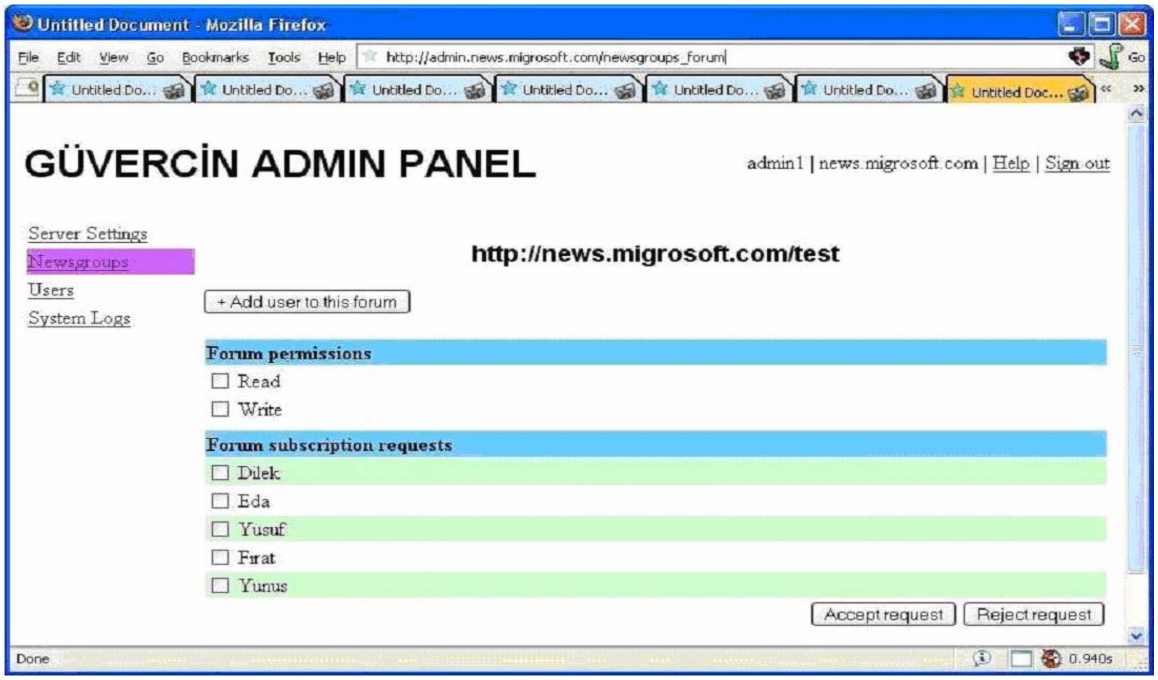
All the Subscription Requests part in these three settings windows corresponding to a single group will not occur if no authentication is needed.


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
hhtp://admin.news.migrosoft.com/newsserversettings.htm
Untitled Document
GÜVERCİN ADMIN PANEL
admin1 | news.migrosoft.com | Help | Sign out
Server Settings
Newsgroups
Users
System Logs
migrosoft.test
+ Add user to this newsgroup
Accept request
Reject request
Done

Newsgroup status
- ○ Moderated
- ○ Archive
- ○ Active

Newsgroup permissions
- ☐ Read
- ☐ Write

Newsgroup subscription requests
- ☐ Dilek
- ☐ Eda
- ☐ Yusuf
- ☐ Fırat
- ☐ Yunus


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://admin.news.migrosoft.com/newsgroups_forum
GÜVERCİN ADMIN PANEL
admin1 | news.migrosoft.com | Help | Sign out
Server Settings
Newsgroups
Users
System Logs
+ Add user to this forum
Accept request
Reject request
Done

http://news.migrosoft.com/test

Forum permissions

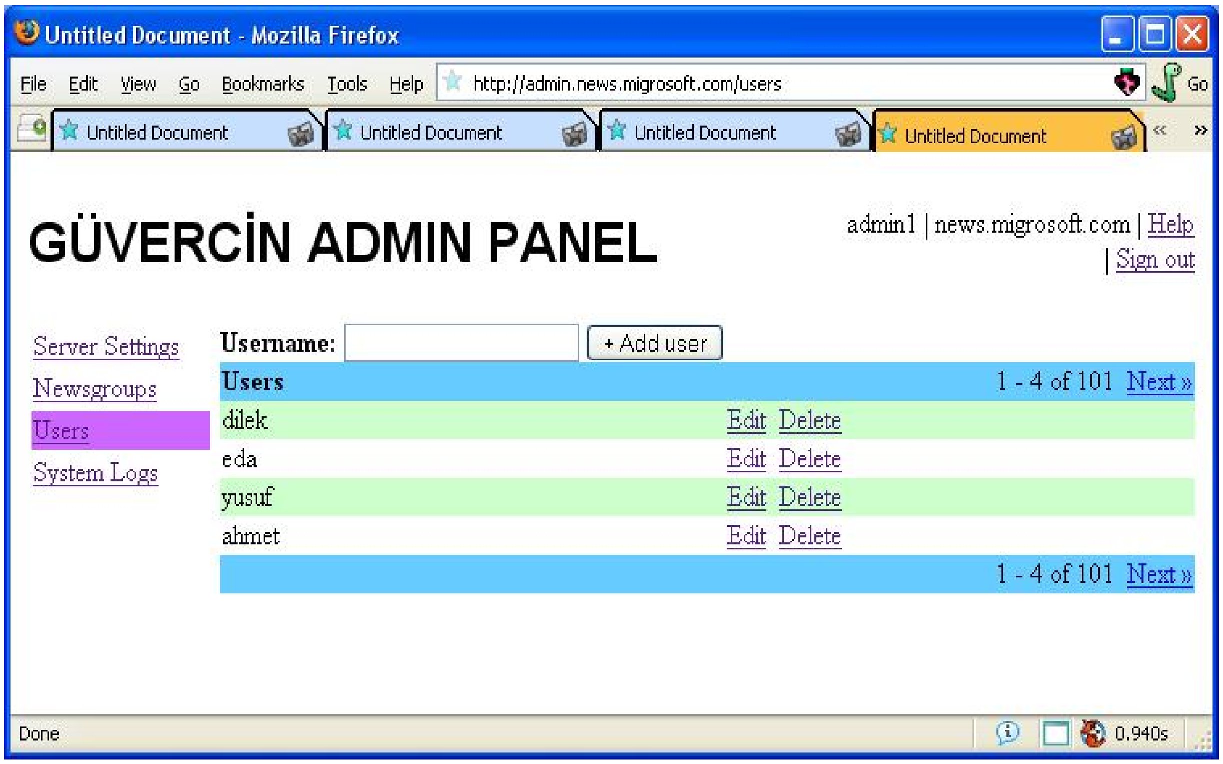
- ☐ Read
- ☐ Write

Forum subscription requests

- ☐ Dilek
- ☐ Eda
- ☐ Yusuf
- ☐ Firat
- ☐ Yunus

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.


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://admin.news.migrosoft.com/users
Untitled Document
Untitled Document
Untitled Document
Untitled Document
Done
0.940s

GÜVERCİN ADMIN PANEL

admin1 | news.migrosoft.com | Help | Sign out

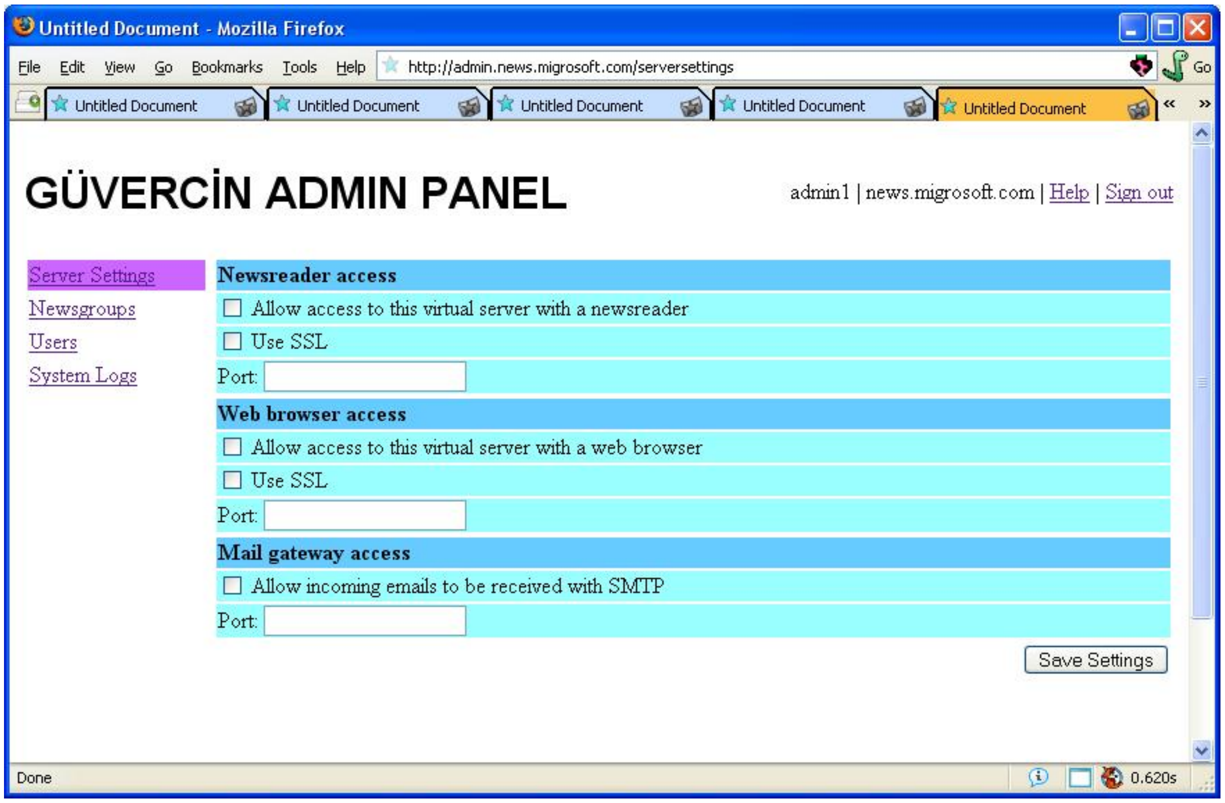
Server Settings
Newsgroups
Users
System Logs

Username: + Add user

Users		1 - 4 of 101 Next »
dilek	Edit Delete	
eda	Edit Delete	
yusuf	Edit Delete	
ahmet	Edit Delete	
		1 - 4 of 101 Next »

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).


Untitled Document - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://admin.news.migrosoft.com/serversettings
Untitled Document
Done
0.620s

GÜVERCİN ADMIN PANEL

admin1 | news.migrosoft.com | Help | Sign out

Server Settings
Newsgroups
Users
System Logs

Newsreader access

☐ Allow access to this virtual server with a newsreader

☐ Use SSL

Port:

Web browser access

☐ Allow access to this virtual server with a web browser

☐ Use SSL

Port:

Mail gateway access

☐ Allow incoming emails to be received with SMTP

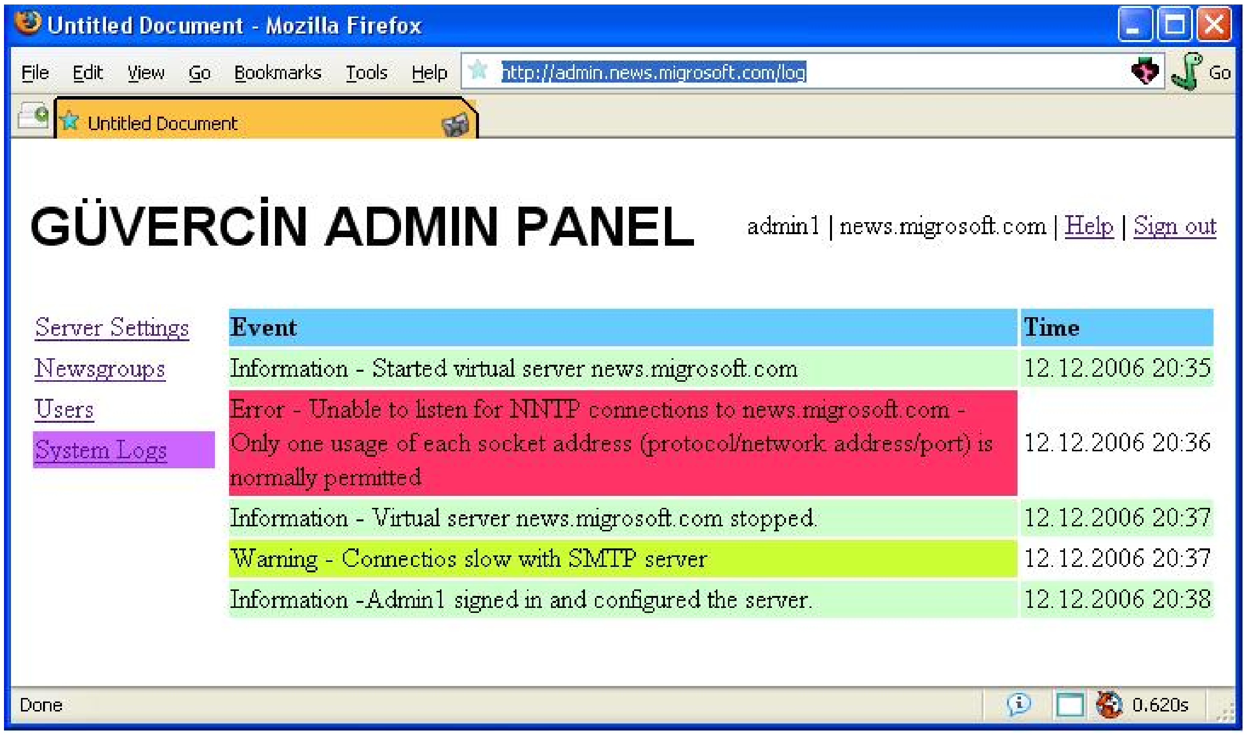
Port:

Save Settings

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.



Untitled Document - Mozilla Firefox

http://admin.news.migrosoft.com/log

GÜVERCİN ADMIN PANEL

admin1 | news.migrosoft.com | Help | Sign out

Server Settings
Newsgroups
Users
System Logs

Event	Time
Information - Started virtual server news.migrosoft.com	12.12.2006 20:35
Error - Unable to listen for NNTP connections to news.migrosoft.com - Only one usage of each socket address (protocol/network address/port) is normally permitted	12.12.2006 20:36
Information - Virtual server news.migrosoft.com stopped.	12.12.2006 20:37
Warning - Connectios slow with SMTP server	12.12.2006 20:37
Information -Admin1 signed in and configured the server.	12.12.2006 20:38

Done

7 Prototype Implementation

So far the team has developed *a PYTHON script* with the following purposes. Although the team has initially decided on PERL, the last decision of the team is to change the implementation language from PERL to PYTHON. Both languages are script languages but PYTHON is designed as an object-oriented language and has a simpler syntax. The project design is object-oriented; to map classes from Güvercin design to PYTHON will be easier.

This script **i**s *a basic implementation of the CORE Module in the design.* As stated before in the design,this module is the center module which connects to the NNTP and SMTP servers and provides message passing to all channels.

This script connects to *the NNTP server* specified in the configuration file "***nntp2mail***", periodically checks if a new message has been posted to the newsgroups listed in the "***newsgroups***" file. If there exists a new post:

- This message is sended to the email address specified in the configuration file "***nntp2mail***" via the SMTP server specified again in the configuration file.
- This message is appended to the xml file used for rss feed for that newsgroup.

Script produces a configuration file "***nntp2mail***" when run with "*-config*" parameter. Once the file is formed , the script can be configured by changing this file. This configuration file comprises the following information:

- NNTP server to connect NNTP server to get news articles from.
- Port that NNTP server is listening on.
- NNTP username and password to login with.
- Email address to deliver news articles to
- SMTP server used to send email

New newsgroups can be added or the existing newsgroups can be removed by "addgrp *groupname"* and "rmgrp *groupname"* commands. These commands update the "newsgroups" file.

An SMTP server is needed to send mails. For implementation APACHE JAMES is installed. APACHE JAMES has a user repository which is used by both the built-in NNTP server and the built-in SMTP server. The script connects to this NNTP server and sends mail to the specified users using this SMTP server.

8 Testing Strategies

The team will prepare a detailed Test Specification Report in three months. Therefore in these sections only the intentions are stated. This section also is a brief summary of the responsibilities of the modules.

8.1 Unit Testing

Modules will be tested for functionality and performance before integration. For the time being we represent the questions to be asked for each module.

- **Login Module**
  - 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?

8.2 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.

8.3 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.

9 Project Schedule

9.1 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

- Forum Module Implementation
- RSS Module Implementation
- Basic Admin GUI Implementation and Testing
- Prototype Integration and Testing
- Prototype Release
- Security Implementation
- SMTP2NNTP Module Implementation
- NNTP2SMTP Module Complete Implementation
- SMTP2NNTP Module Complete Implementation
- Core System Integration
- Alpha Testing
- Beta Testing and Debugging
- Core System Release

Work Package 3 – Admin Panel

- GUI Design Completed
- GUI Implementation Completed

- Panel Implementation
- Alpha Testing
- Beta Testing and Debugging

Work Package 4 – Finalizing Activities

9.2 Gantt Chart

Our updated Gantt chart can be seen in the Appendix.

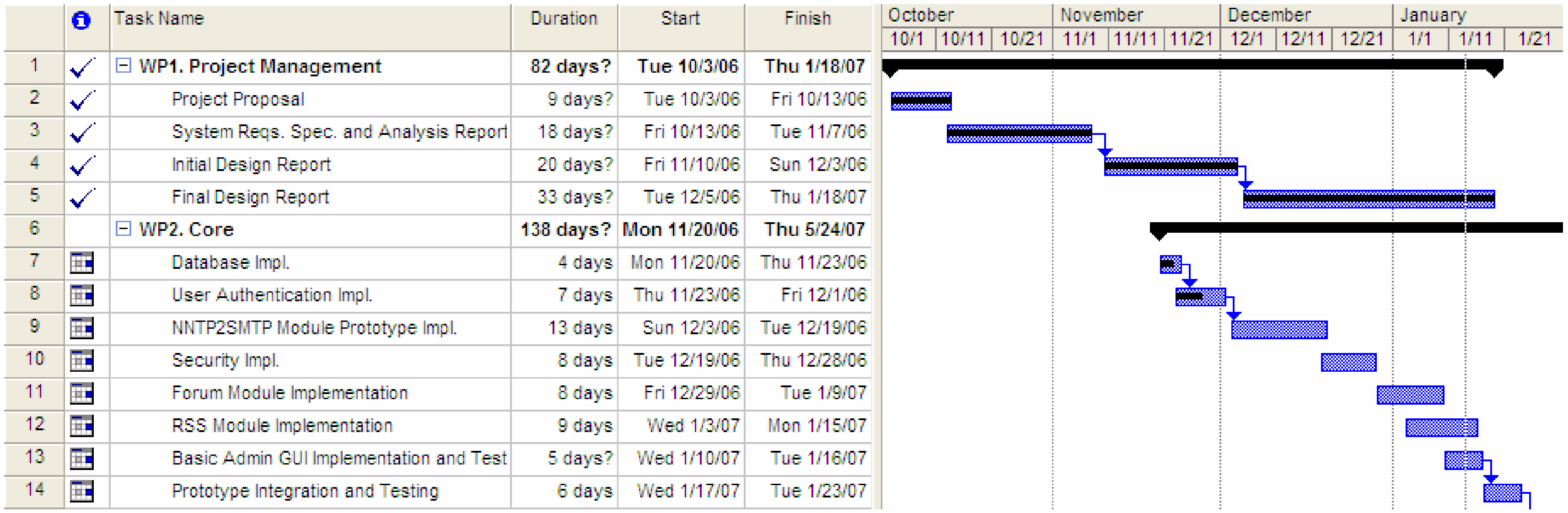
References

[1] inet-tr.org.tr/inetconf8/bildiri/105.doc

[2] http://www.apache.org/

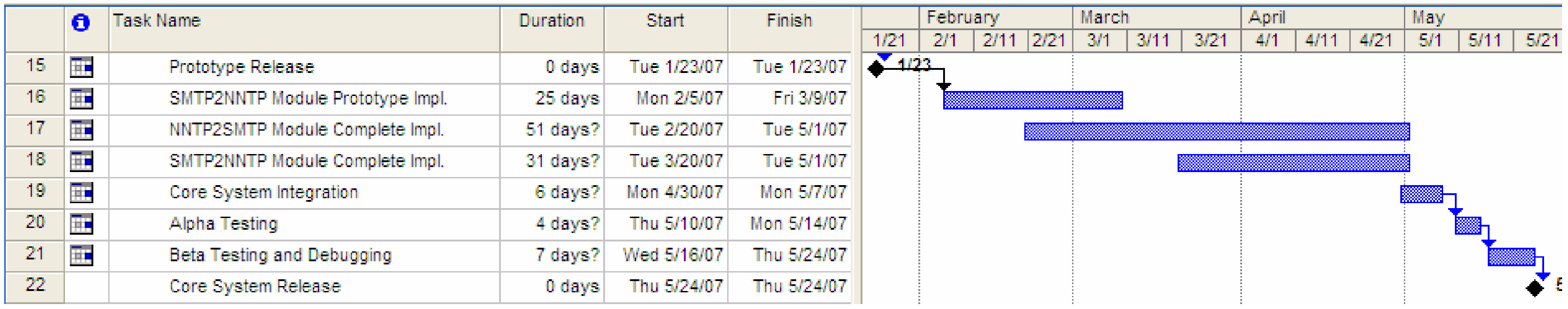
[3] http://james.apache.org/

Appendix



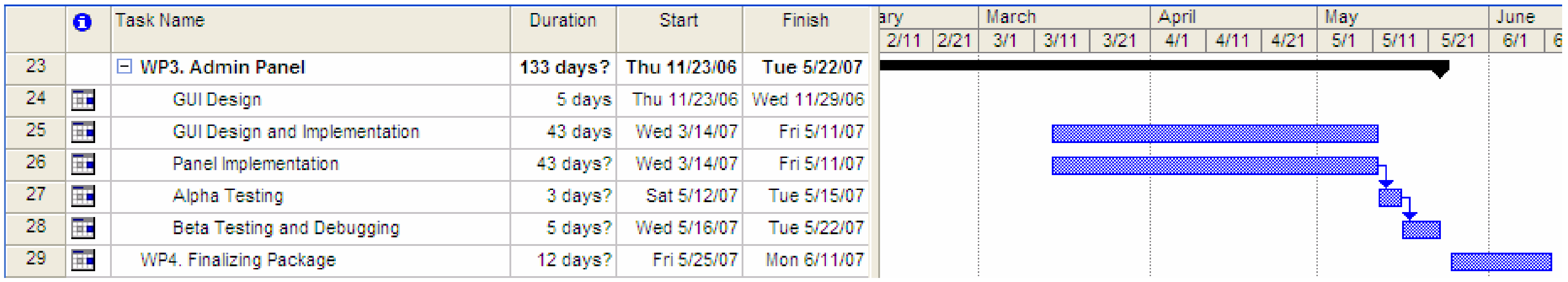
	ℹ	Task Name	Duration	Start	Finish
1	✓	⊟ **WP1. Project Management**	**82 days?**	**Tue 10/3/06**	**Thu 1/18/07**
2	✓	Project Proposal	9 days?	Tue 10/3/06	Fri 10/13/06
3	✓	System Reqs. Spec. and Analysis Report	18 days?	Fri 10/13/06	Tue 11/7/06
4	✓	Initial Design Report	20 days?	Fri 11/10/06	Sun 12/3/06
5	✓	Final Design Report	33 days?	Tue 12/5/06	Thu 1/18/07
6		⊟ **WP2. Core**	**138 days?**	**Mon 11/20/06**	**Thu 5/24/07**
7	▦	Database Impl.	4 days	Mon 11/20/06	Thu 11/23/06
8	▦	User Authentication Impl.	7 days	Thu 11/23/06	Fri 12/1/06
9	▦	NNTP2SMTP Module Prototype Impl.	13 days	Sun 12/3/06	Tue 12/19/06
10	▦	Security Impl.	8 days	Tue 12/19/06	Thu 12/28/06
11	▦	Forum Module Implementation	8 days	Fri 12/29/06	Tue 1/9/07
12	▦	RSS Module Implementation	9 days	Wed 1/3/07	Mon 1/15/07
13	▦	Basic Admin GUI Implementation and Test	5 days?	Wed 1/10/07	Tue 1/16/07
14	▦	Prototype Integration and Testing	6 days	Wed 1/17/07	Tue 1/23/07

Figure 23: First Part of the Gantt Chart



		Task Name	Duration	Start	Finish
15		Prototype Release	0 days	Tue 1/23/07	Tue 1/23/07
16		SMTP2NNTP Module Prototype Impl.	25 days	Mon 2/5/07	Fri 3/9/07
17		NNTP2SMTP Module Complete Impl.	51 days?	Tue 2/20/07	Tue 5/1/07
18		SMTP2NNTP Module Complete Impl.	31 days?	Tue 3/20/07	Tue 5/1/07
19		Core System Integration	6 days?	Mon 4/30/07	Mon 5/7/07
20		Alpha Testing	4 days?	Thu 5/10/07	Mon 5/14/07
21		Beta Testing and Debugging	7 days?	Wed 5/16/07	Thu 5/24/07
22		Core System Release	0 days	Thu 5/24/07	Thu 5/24/07

Figure 24: Second Part of the Gantt Chart



		Task Name	Duration	Start	Finish
23		⊟ **WP3. Admin Panel**	**133 days?**	**Thu 11/23/06**	**Tue 5/22/07**
24		GUI Design	5 days	Thu 11/23/06	Wed 11/29/06
25		GUI Design and Implementation	43 days	Wed 3/14/07	Fri 5/11/07
26		Panel Implementation	43 days?	Wed 3/14/07	Fri 5/11/07
27		Alpha Testing	3 days?	Sat 5/12/07	Tue 5/15/07
28		Beta Testing and Debugging	5 days?	Wed 5/16/07	Tue 5/22/07
29		WP4. Finalizing Package	12 days?	Fri 5/25/07	Mon 6/11/07

Figure 25: Third Part of the Gantt Chart