Middle East Technical University

Department of Computer Engineering

`A Unified News Exchange Server `

Detailed Design Report

Goncagül DEMİRDİZEN

Hilal KARAMAN

Ali Anıl SINACI

Ferhat ŞAHİNKAYA

"NewsAgent"

by

i$T€ Yazılım

Fall, 2006

1 INTRODUCTION

The design period of a software project is the most important part of the schedule since it determines the other steps of the project. Until this time, we had specified our requirements and in the light of our requirement analysis reviews, we have prepared the initial design of our project. In this period we have understood the details and different aspects of the project more clearly and the system has been visualized in our minds precisely. As i$T€ team, in our detailed design report, we examined our design issues in initial design and specified our design in a detailed way. Through the design process, we aimed to design an efficient and modular system which satisfies all concept of the problem and tried to develop practical and applicable solutions to the problem. For this purpose, we specified our system modules clearly and determined the interactions between the modules and boundaries of the modules. We believe that our final design satisfies the concept of the problem and provides a reasonable and a modular solution to the problem.

1.1 Project Scope & Definition

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

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

1.2 Project Description

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

- Administrators will be people who are responsible from the management of newsgroups, users and user groups. Creating, removing new newsgroups or handling of undesirable articles in any of the newsgroups will be in the scope of his/her responsibilities. Moreover, they also deal with user management. When a candidate user requests to be a user of our system, administrators will be responsible to accept or reject their request and adding, deleting user and modifying user rights will also be responsibilities of administrators.
- Web users will be able to access newsgroups and articles through a graphical user interface. Web user will login to the system and after this authentication they will be able to realize all article-based and newsgroup-based operations according to their access level. An unauthenticated web user will be able to realize only part of these operations since their access levels will cover a small set of these operations. Web component will also provide management facilities for each user such as update user info, change login info etc. and a user-friendly interface will provide user to reach data, quickly.
- NNTP users will be able to access newsgroups through tin or NNTP clients, like Mozilla, Thunderbird or Microsoft Outlook Express. They will also be separated as authenticated and unauthenticated NNTP users. Authenticated NNTP users will be able to realize all article-based and newsgroup-based operations according to their access level. Unauthenticated NNTP users will be able to realize only part of these operations.
- RSS/Atom users will be able to receive feeds from newsgroups according to their wishes. We will create separate RSS and Atom feeds for every newsgroup and whenever, a new article is posted we will append this article as a new item to our feed tree of the related newsgroup and we will serialize it. We will also delete the old items in the feed and users will be able to access new data via their RSS/Atom readers.
- Atom is a little bit different from RSS in the sense that atom users will be able to send insertion data to the feeds directly or update or delete data from feeds. NewsAgent will provide this to the Atom end users and their insertion requests will be handled.
- We will present a mailing option for our users and users will be able to set / reset their

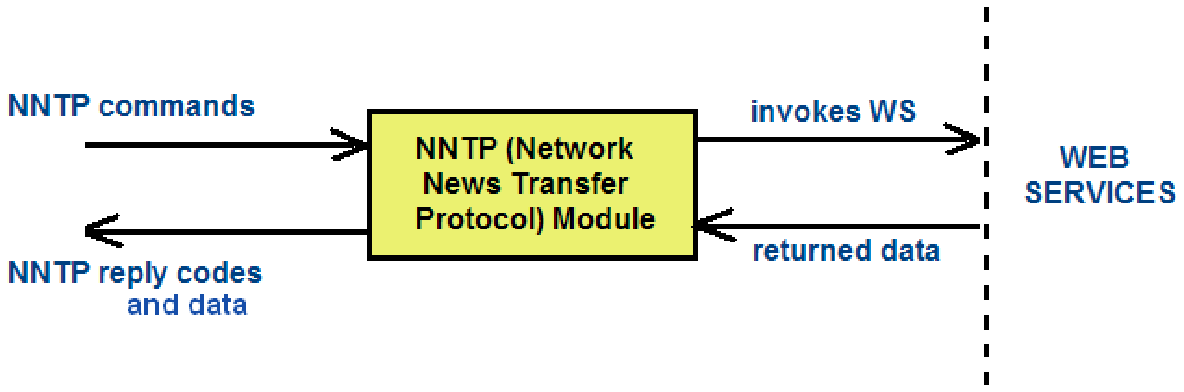
mailing option and as a result e-mails will be sent to these users if they want to receive post in a newsgroup via e-mail. Moreover, the users will be able to choose different receiving options such as instant, daily, weekly etc. Mail users will be able to receive mails from different newsgroups according to their wishes. Whenever a new article is posted, e-mails related to that article will be sent to the users who request to receive e-mail from that newsgroup according to their mail receiving criteria. Moreover, mail users will be able to send posts to newsgroups as a new thread or as a follow-up. When the user sends mail to the system we will check the user is registered and send a verification mail to the sender. If the sender approves, then the e-mails from registered users will be converted to article objects and inserted as articles into newsgroups.

- *NewsAgent* will contain several user groups and each user group will have different access rights. Authentication will specify access rights of each user and user will be able to access different newsgroups according to their rights and newsgroups that they are subscribed. In addition to user groups, also there will be a general access right which will not need authentication and user will be able to access some subset of newsgroups which is specified by the system administrators.
- *NewsAgent* will have a log mechanism in order to save all login information and any configuration made in the system. For this purpose, we keep login logs and configuration logs respectively. Log mechanism is important for security of the system in the sense that the reason of any failure can be found by the help of logs and also we will be able to keep track of the login logs denoting the users' login and the IP of the computer that they have logged in and any administrator configurations.
- *NewsAgent* will also provide extra features for the users. For example, web users will be able to communicate with online users by the help of instant messaging functionality and will be able to send messages to the offline users by the help of private messaging. These messages will be displayed to the receiver when he/she logins to the system through our web interface.

2 NEWSAGENT MODULES

2.1 NNTP Module

Our NNTP module provides the connection between NTTP clients and *NewsAgent*. The end-users connected via NNTP clients are served and their requests and the NNTP commands sent to the server as a result of these requests are handled by the help of NNTP module. The following figure shows the overview of our NNTP module basically.


NNTP commands
NNTP (Network News Transfer Protocol) Module
invokes WS
WEB SERVICES
returned data
NNTP reply codes and data

This module accepts USENET NNTP commands such as POST, LIST, ARTICLE commands which are sent as a result of posting article, listing newsgroups etc. requests and maps these commands to the corresponding web service invocations by parsing these NNTP commands and data. Then this module returns suitable NNTP reply codes and necessary data to the clients with respect to the values returned by the web services. During accesses to the server, system administrators may activate secure connections through SSL (Secure Sockets Layer) by publishing the server's public key. We will use OpenSSL and Java built-in libraries to handle this feature.

In our system, NNTP end-users are classified as authorized and unauthorized users. Unauthorized users can only reach only some subset of newsgroups, which are specified by system administrators adjusting the newsgroup access rights. In fact, that is reasonable, since user group of unauthorized users has access level to only these newsgroups. If the user is authorized, he/she will have a more extensive access than unauthorized users. However, there will also be different access rights between the user groups of registered users. They will have the right of do the actions that their access level contains. If user is registered, following functionalities will be provided to the user:

- The user logins to the system by entering his/her username and password. Username and password are controlled for validation from the database. If username-password combination is not valid, the authentication process fails and user cannot access the news as an authorized user. If the authentication process results in a success, a session will be created for the user and an access level is assigned to the user corresponding to the user group.
- After authentication process for registered users, they will have the right of posting/reading articles, listing articles of a newsgroup, listing newsgroups, etc.

2.2 Web Module

Our Web module will handle the requests and activities realized via our web interface. These requests and activities include administrative operations, news related activities, the user activities, login and sign up actions and the private and instant messaging activities. Therefore, our web module consists of 5 sub modules namely administration module, user affairs module, news access module, authentication and registration module and messaging module.

❖ Administration Module

Administration module deals with the administrative operations that the system administrators are responsible for in our system. Our web interface will include an administration interface for these operations and only the system administrators (users who are member of the admin user group) will be able to access and make configurations via this interface. The followings are the functionalities which our administration module handles.

- **Newsgroup Management:** Administrators will have the right of creating new newsgroups, deleting an existing newsgroup and make modifications on newsgroup access levels etc. Such kind of newsgroup related operations are handled in the concept of this module. Newsgroup information which the administrator decides to add or the newsgroup id that will be deleted or modified is obtained from the administrator through the administration interface and administration module invokes related web services which interact with the database layer in order to reflect the changes.

- **User Management:** Administrators will have the right of adding new users, deleting an existing user. Administration module also handles user related operations of the administrators. As in the newsgroup management, required input is obtained from the administrator and administration module invokes the related user management web service and interacts with the database.
- **User Group Management:** Administrators create new user groups, remove existing user groups and modify the user rights of the user groups in order to adjust the access rights to the existing newsgroups. Administration module invokes related web service for user group management and these web services retrieve necessary data from database or reflect the necessary changes to the database.
- **Log Management:** In our system, login actions and any configuration are saved in logs and administrators can list logs or make any changes such as deleting or modifying logs. Log management operations are also handled in administration module.

❖ User Affairs Module

User Affairs module deals with the user activities related to the user info or account info. There will be a user affairs interface in our system and the User Affairs module will be responsible for the actions and operations related to the user info. The followings are the user requests that the user affairs module handles.

User will be able to

- display user info.
- update user info.
- change password.

User Affairs module interacts with the database and the web service layer and when a user requests to display user info, it retrieves the user info from database and displays. This module gets the new information or data from the user and updates the related fields as a result of an update request.

❖ News Access Module

News Access Module will be responsible for the article and newsgroup related operations. The user will request to list the newsgroups and news access module will interact with the database and retrieve the appropriate newsgroup according to the access level of the user group that the user belong to. With the list of the newsgroups, subscription or the mailing options will also be displayed to the user and the user will be able to subscribe/unsubscribe to the newsgroups or set/reset mail receiving options from these newsgroups. Mail receiving options will have different options such as instant, daily and weekly. The user will determine the period which he/she requests to receive mail for the articles in the newsgroup. For example, when weekly option is selected, the user will receive mail once a week for that newsgroup and receive the articles in that one week period.

When one of the newsgroups is selected, the article information of that group is retrieved and the header, author and date information of the articles are displayed. On the other hand, article operations are also handled in news access module. When the user selects one of the articles displayed, the get article web service is invoked and it retrieves the related article's text from the database and displays the article content. Moreover, posting operation is similar. Post article web service is invoked and it interacts with the database access and inserts the posted article to the database. On the other hand, the user will be able to cancel or update his/her articles and sort the newsgroups or articles in a newsgroup according to some criteria such as name, creation or post date etc.

❖ Authentication & Registration Module

This module will be responsible for the login and sign up operations. When a user enters his username and password in order to login through our web interface, authentication module will receive the username and password. Then related web service will be invoked to check whether the username password combination exist in database or not. For security reasons, password will be held in a MD5 (Message-Digest algorithm 5) format. This hashing technique will prevent anyone to access passwords of the users, directly. After authentication a session will be created for the user and the user group of the user will also be assigned.

Signing up to the system will also be realized via our web interface. A candidate user fills the registration form which will be displayed as a result of sign up request and submits this form. Registration module controls the validity of the form and interacts with the database and saves the user info. Moreover, this module sends a confirmation mail to the administrators. If the administrator accepts the user, the user group and the access rights are adjusted by the administrator and the username and randomly generated password are sent to the user. After this candidate user turns out to be a system user.

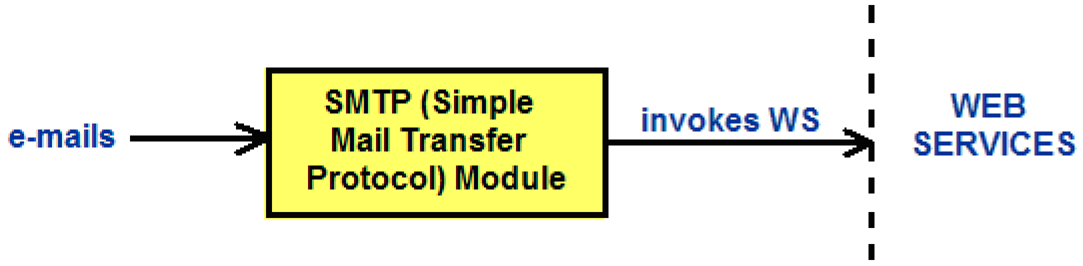
❖ **Messaging Module**

Messaging module is responsible for the instant and private messaging issues. This messaging concept is designed as an extra feature that *NewsAgent* presents to the users. User will display the online users and be able to communicate with the online users. By this way, some unnecessary data will not be sent as an article to the newsgroup. Users will send each other as instant message. Private messaging is also another new feature which is similar to instant messaging. Private messaging provides users to send messages to any other users – online or offline –. Private messages will be shown to recipient when he/she logs in to the system.

2.3 Mail Module

Mail module is responsible for the receiving e-mails and sending e-mails in our system.

- When our system receives an e-mail, first of all the system controls whether the sender is an authenticated mail client or not. If the sender is authenticated then a verification mail is sent to the sender whether he/she approves the insertion of the mail as an article. Such kind of verification is important in order to prevent spam mails. If the sender approves, the e-mail is converted to an article object, related web service is invoked and inserted to the database. The article will be added to a newsgroup which is specified in the address field of the mail content.



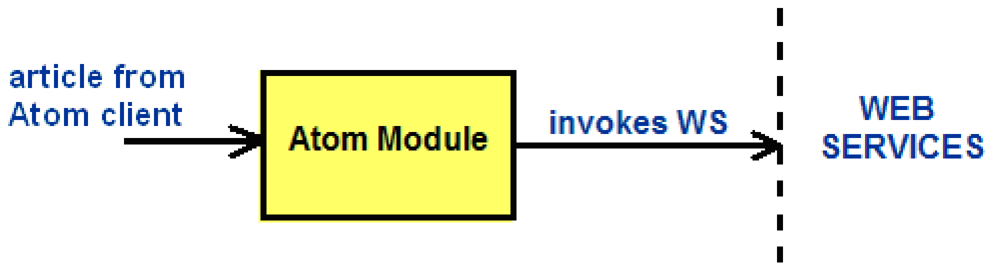
- Users can access articles in a newsgroup via e-mail depending on whether he/she sets his mailing options on. Of course, user will be able to receive mail from only newsgroups which he/she can subscribe corresponding to his/her user group. For a newsgroup, if the user requests articles as an e-mail according to the receiving option such as instant, daily, weekly, mail module generates e-mails from the articles and sends to the users.

2.4 RSS Module

Our system will provide RSS feeds for every newsgroup and RSS module will be responsible for the generation and control of these feeds. If the user wants to follow a newsgroup periodically, user can subscribe to the RSS feed of this newsgroup and by using an RSS reader, he/she can reach articles in the newsgroup. After every post operation, the RSS feed generator is called and the article is appended to the RSS feed of the newsgroup that the article is posted. After a period, the feed will become to be large and the old articles will be deleted from the feed.

2.5 Atom Module

Our system will also provide Atom feeds for every newsgroup and Atom module will be responsible for the generation and control of these feeds. Users will be able to subscribe to the Atom feed of this newsgroup and access articles in the newsgroup via their Atom readers. After every post operation, the Atom feed generator is called and the article is appended to the Atom feed of the newsgroup that the article is posted. After a period, the feed will become to be large and the old articles will be deleted from the feed.
Moreover, Atom protocol has some advantages on RSS such that Atom is more powerful for transferring binary data when compared to RSS and user can send insertion requests directly to the Atom feeds, which is not possible for RSS feeds. Our ATOM module handles these insertions, also deletion and update requests, which are carried in the same manner with HTTP requests by related web services.



2.6 Authentication Module

Authentication module is responsible for the authentication process and other modules interact with authentication module as a result of a login action or authentication necessity.

- As mentioned in previous modules, each user will be a member of a user-group which specifies the access level of the user. During authentication username will be checked for specifying whether username is in database or not.
- Username and password will be checked for correspondence between them by invoking the related web service and interacting with the database.
- For security reasons, password will be held in a MD5 (Message-Digest algorithm 5) [references: http://en.wikipedia.org/wiki/MD5] format. This hashing technique will prevent anyone to access passwords of the users, directly.
- After authentication a session will be created for the user and will live until the user logs out or times out after a period.
- A user who is not authorized to the system will be able to access only some subset of newsgroups and read only articles in these newsgroups.

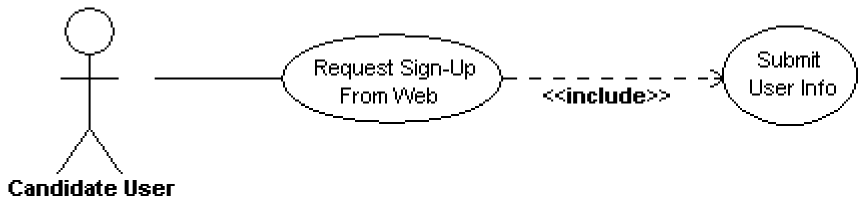
2.7 System Log Module

System Log module is responsible for the login log and configuration log operations. As mentioned before, every login operation and any configuration made in the system by system administrators are saved in login logs and configuration logs respectively. In this kind of a situation, log module inserts login log or configuration log by invoking the related log web service for saving the logs.

3 USE CASES

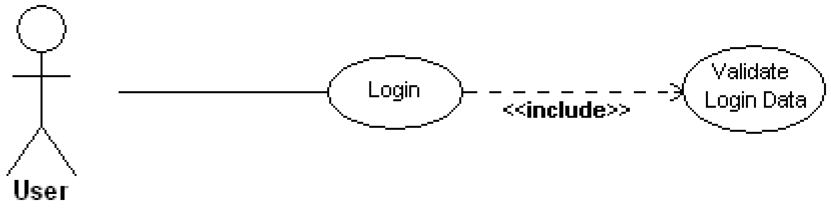
3.1 Use Case Diagrams

3.1.1 Signup Use Case



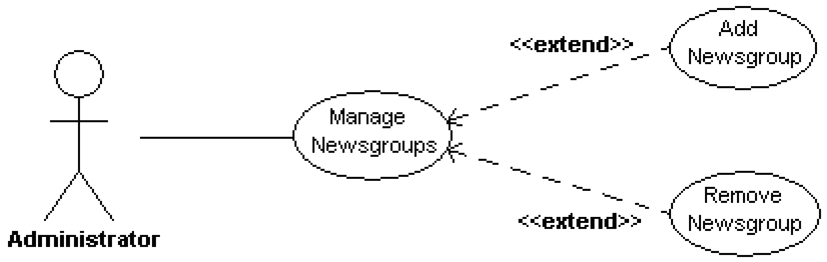
Flow of Events for sıgnup Use Case	
Objective	Allow candidate user to become a system user.
Precondition	Access to the system through web interface.
Main Flow	• Candidate user reaches the system via web module. • Clicks 'signup' button. • A user info form is displayed. Candidate user fills in this form without a missing point. • Candidate user submits the form.
Postcondition	User waits for a confirmation message from administrators. Then he/she becomes a user of the system.

3.1.2 Login Use Case

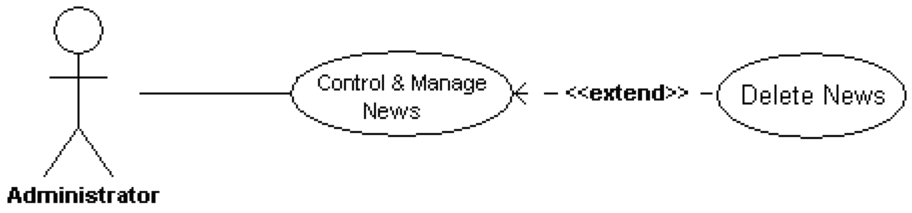


Flow of Events for LOGIN Use Case	
Objective	Allow users to be authenticated.
Precondition	
Main Flow	• User submits username and password. • System checks this login data from the database. • If it is not verified, user is rejected. • If the username and password is correct, user is authenticated. • User rights are determined according to this data. • Administrative rights are also determined by login data.
Postcondition	User is allowed to get into the system.

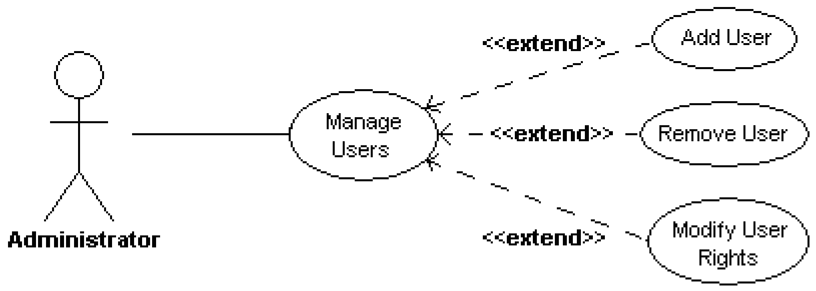
3.1.3 Administrative Use Cases



Flow of Events for MAnage newsgroups use case	
Objective	Allow administrator to add or remove newsgroups
Precondition	Administrative rights (The user must be logged in as an administrator)
Main Flow	• Administrator chooses 'manage newsgroups' option. • In order to add a new newsgroup, administrator interacts with 'add newsgroup' interface, specifies necessary information about the newsgroup and adds the newsgroup. • In order to remove a new newsgroup, administrator interacts with 'delete newsgroup' interface, selects the newsgroup and removes it.
Postcondition	New newsgroup is added to the system. Deleted newsgroup does not exist in the system anymore.

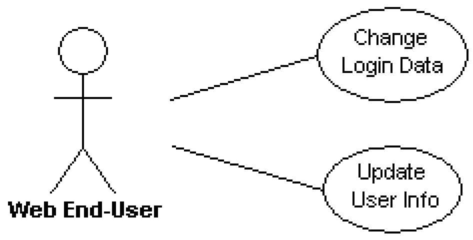


Flow of Events for control & manage news	
Objective	Allow administrator to control and manage news in order to provide a qualified environment.
Precondition	Administrative rights (The user must be logged in as an administrator)
Main Flow	• Administrator controls news that are posted to the server. • He/she has the right to delete news in case that it doesn't suit the newsgroup or generally, the system. • Or administrators may warn the users about the messages they sent, instead of deleting the news.
Postcondition	Some messages may be deleted.



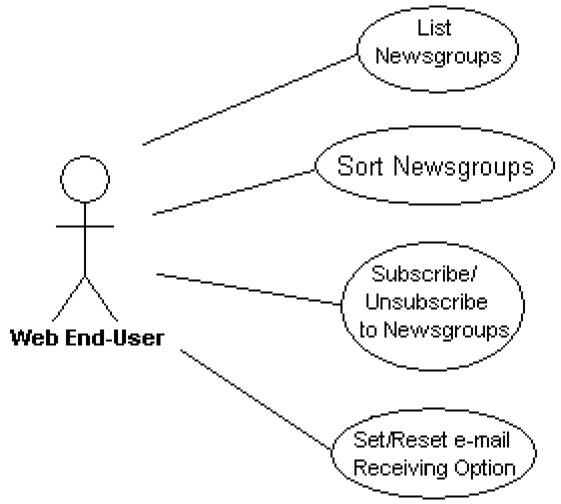
Flow of Events for manage users use case	
Objective	Allow administrator to add/remove users and modify user's rights
Precondition	Administrative rights (The user must be logged in as an administrator)
Main Flow	• Administrators interact with 'manage users' interface. • By approving submitted user forms, they can add a new user. • Administrators have the right to remove a user who does not satisfy the requirements to be a user from the system. • Administrator can also change the user's rights. These rights determine the user's access permission. Administrator can chage user's access permissions.
Postcondition	New user is added. A user is removed. User rights are modified for any specified user.

3.1.4 Web Client Use Cases

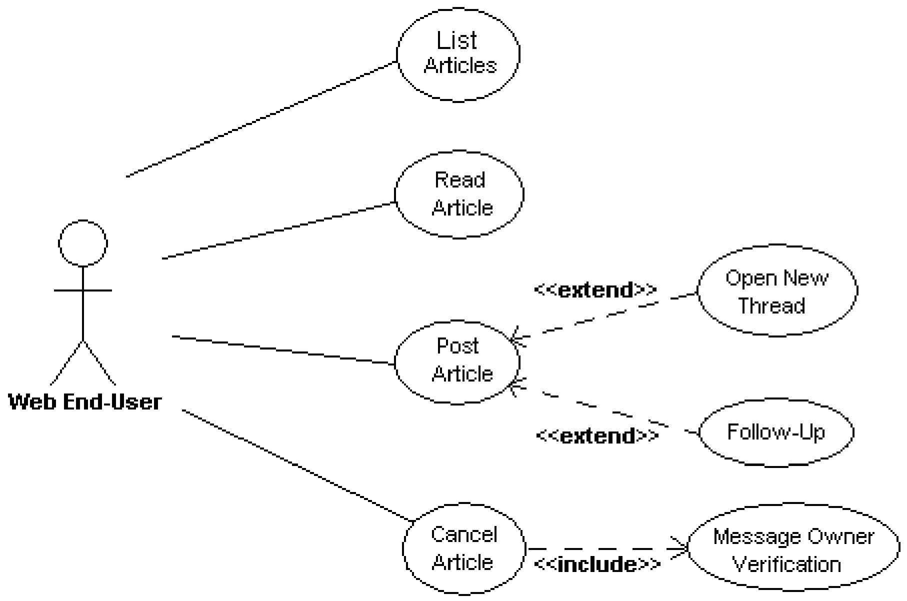


Flow of Events for WeB Clıent's update Account ınfo use case	
Objective	Allow users to modify their user information and change their password.
Precondition	User must be logged into our system through web by interacting with our web module.
Main Flow	• After logging in, user can select 'Update Account Info' option. • Then the user's account info is displayed. • Some fields will be displayed disabled. That is, the user will not be able to change this info. For instance, *username*. • Login data means username and password. User will be able to

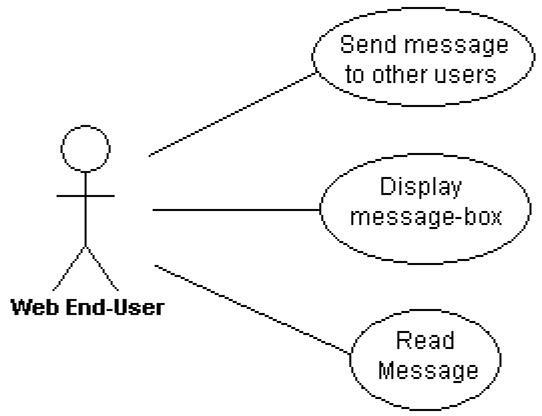
	change password, by first entering the old password. If old password is not confirmed, system will not allow the user to change password. If it is correct the password will be updated with the new entered one. • User info means name, surname, phone, etc. User will be able to change his/her user info. These changes will be reflected to the database.
Postcondition	User's password or user info has changed and inserted into the database.


Web End-User
List Newsgroups
Sort Newsgroups
Subscribe/ Unsubscribe to Newsgroups
Set/Reset e-mail Receiving Option

Flow of Events for WeB Clıent's Newsgroup use cases	
Objective	Allow users to list/sort newsgroups, subscribe/unsubscribe to newsgroups, set/reset e-mail receiving option through our web module.
Precondition	User must be logged into our system through web by interacting with our web module.
Main Flow	• After logging in, user can select 'List Newsgroups' option. Then newsgroups are listed. User can list his/her subscribed newsgroups or all newsgroups of the news server. • Sorting mechanism can differ according to the user's choice. User can sort them alphabetically, according to date, etc. • After listing these newsgroups, user can select any of them and subscribe to that newsgroup. Or vice versa, the user can unsubscribe from a newsgroup that he/she is already subscribed to. • For the newsgroups that the user is subscribed to, the user can set/reset mail receiving option. If it is set, articles that are posted to that newsgroup are sent to the user as e-mail.
Postcondition	Subscriptions or changes in e-mail receiving options are inserted into the database.

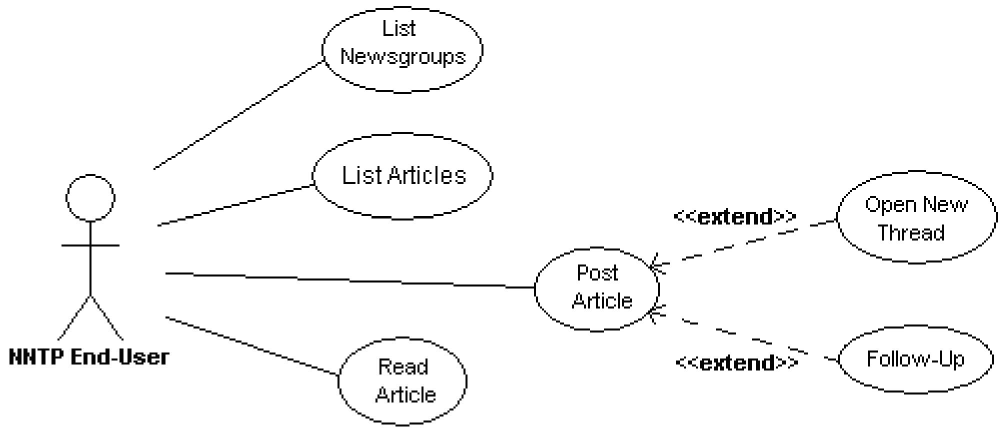


Flow of Events for WeB Clıent's ARTıcle use cases	
Objective	Allow users to list, read, post or cancel articles through our web module.
Precondition	User must be logged into our system through web by interacting with our web module.
Main Flow	• After selecting a newsgroup, user can list the articles belonging to that group by clicking the name of that newsgroup. • Articles are listed. Read articles are displayed in a different color. • Then, by clicking on the header, user can display the content of that article. • User can post a reply to this article by clicking 'post a reply' option. • Instead of posting reply to any article, user can open a new thread. • In both situations, a form is displayed. User fills in the required parts and sends the article. • If the user wants to delete the article after sending, he/she can select 'delete article' option. In order to delete the article, owner of that article must be him/her.
Postcondition	New article is inserted into database or an article is deleted from database.



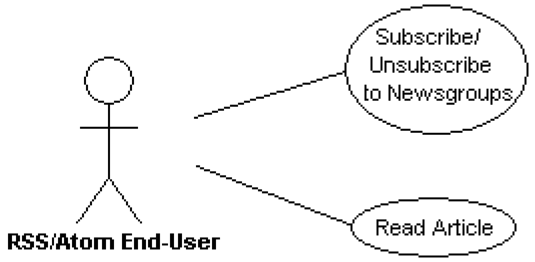
Flow of Events for WeB Clıent’s messages use cases	
Objective	Allow users to send messages to other online/offline users.
Precondition	User must be logged into our system through web by interacting with our web module.
Main Flow	• If the user wants to send message to another user, he/she can select the user by double clicking his/her username in users list which we will display. • When the user is selected, a pop-up window is displayed and the message is expected to be written there. • Then the user presses “send” button, and the message is sent to recipient. • If the user wants to see the messages that are sent to him/her, the user will firstly press “message inbox” button and overview of all messages (sender, date, etc.) are displayed. • The user can select any of these messages by clicking on the header, and the message body is displayed.
Postcondition	Sent messages are inserted into database with sender and receiver info.

3.1.5 NNTP End-user Use Cases

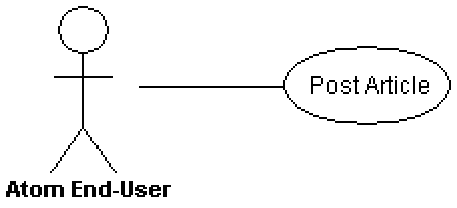


Flow of Events for NNtp end-user use cases	
Objective	Allow users to list available newsgroups, list articles of a specified newsgroup, read an article among these listed ones and post a new article (either as a new thread or as a follow-up to an existing article) through NNTP clients.
Precondition	User must be connected to our system through an NNTP client (such as Outlook Express, Thunderbird, etc.) by authentication.
Main Flow	Flows of these events are mainly the same. Only the nntp response and request codes differ. • NNTP client sends a message specifying the end-user's request. • We map the action that corresponds to this message. • Action is performed and reply code ant required information is sent to the client.
Postcondition	Then end-user is able to list newsgroups, list articles, read an article or post an article to the news server through nntp according to the action he/she performed.

3.1.6 RSS/Atom End-user Use Cases

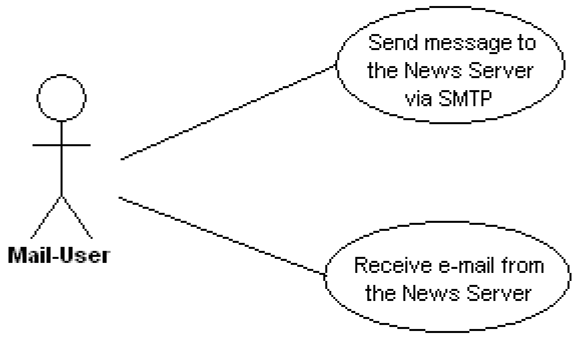


Flow of Events for Rss/atom end-user use cases	
Objective	Allow users to reach the latest news of our server through news readers.
Precondition	User has to subscribe to our newsgroups through RSS/Atom reader.
Main Flow	• We will produce RSS/Atom feeds for each of our newsgroups. • Feeds will be protected. So the user will be asked for username & password by the news reader or attach username password to the url, in order to subscribe to newsgroups. • When subscribed, latest news will be displayed in the news reader. • User will be able to read any article among displayed ones.
Postcondition	---



Flow of Events for atom end-user's post artıcle use case	
Objective	Allow Atom end-users to make comments and reply to our news.
Precondition	User has to subscribe to our newsgroups through Atom reader.
Main Flow	• User will be able to post article to that newsgroup.
Postcondition	Posted article is inserted into database under according newsgroup.

3.1.7 Mail User Use Cases


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

Flow of Events for atom end-user's post artıcle use case	
Objective	Allow users to get the news from the server or send news to the server without logging into the system.
Precondition	User has to set mail receiving option for the newsgroup from our web module.
Main Flow	• If the user's mail receiving option is set for a newsgroup, when a new article is sent to that newsgroup, the article will be sent to the user's mail-box. • In reverse direction, user will be able to send article to the news server as e-mail by entering newsgroup@newsagent.com into "to" field.
Postcondition	Article(s) is(are) sent to the mail user, according to the period he/she has specified. (daily or weekly). Posted article is inserted into database under according newsgroup.

3.2 Use Case Scenarios

Administrator:

Login: An administrator has to login to the system in order to realize administrative roles. There will be a web user interface for administrative roles. After validation of login information, the administrator will be able to manage newsgroups, users and news.

Manage Newsgroups: Administrator may add new newsgroups and remove existing newsgroups in the content of the managing newsgroups scenario.

Manage Users: Administrator may add and remove users and modify the user rights. Administrator will control users and will be able to restrict the user rights. There will be specified user roles and rights, however, new rights can be granted to the users and existing rights may be withdrawn.

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

Candidate User:

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

Web End-User:

Login: The user will login to the system in order to realize user roles. After validation of user login information, the user will be able to list, subscribe/unsubscribe, and sort newsgroups and post, read, cancel and sort articles.

List Newsgroups: The user will be able to list the newsgroups. In the concept of listing newsgroups scenario, a user may list all newsgroups or the newsgroups that he/she has been subscribed.

Sort Newsgroups: The user will be able to sort the newsgroups according to some criteria. These criteria can be alphabetical order, order according to date, etc.

Subscribe / Unsubscribe to Newsgroups: After listing the newsgroups, the user will be able to subscribe and unsubscribe to the newsgroups.

List Articles: The user will be able to list articles belonging to any newsgroup, clicking the name of that newsgroup.

Read Article: The user reads articles.

Post Article: The user posts articles. In the concept of posting articles, the user may open a new thread or follow up to an existing article.

Set & Reset Mail Receiving Options: The user will be able to request to receive e-mail for the articles posted. The user may want to receive e-mail for specified newsgroups or want to receive e-mail for all newsgroups. Also the user may want to cancel the mail receiving option and then no e-mails will be sent to the user from that newsgroup.

Update User Info: The user will be able to update user information such as his/her personal information registered when signing up, e-mail address etc.

Change Login Data: The user may change login information. Generally user id of a user is not allowed to be changed for most of the systems however the users may need to change their passwords.

Send Message to Other Users: The user will be able to send messages to other users. If the receiver user is online, then he/she will immediately receive the message. If the user is offline, he/she will receive the message when he/she logs into the system. User will be selected from the list of users, which we display.

Display Message Box: The user will display message inbox in order to see the overview of messages that are sent to him/her.

Read Message: When any message is selected from this inbox, contents of it will be displayed.

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

The user will be able to realize the actions such as login, list newsgroups and articles, post and read article as web users do. Only difference will be how we handle these requests. We will send responses to the messages that we receive from nntp clients, according to the user's needs. Results of these actions will be the same as web-user.

RSS/Atom End-User:

Subscribe / Unsubscribe to Newsgroups: RSS/Atom end-users will be able to subscribe and unsubscribe to specific newsgroups. Each newsgroup will have its own feed so that the user receives only the news from subscribed newsgroups.

Read Articles: As all users do, RSS users will read the news.

Post Article: Only Atom users will be able to post article to our system through a news reader, not RSS users.

Mail User

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

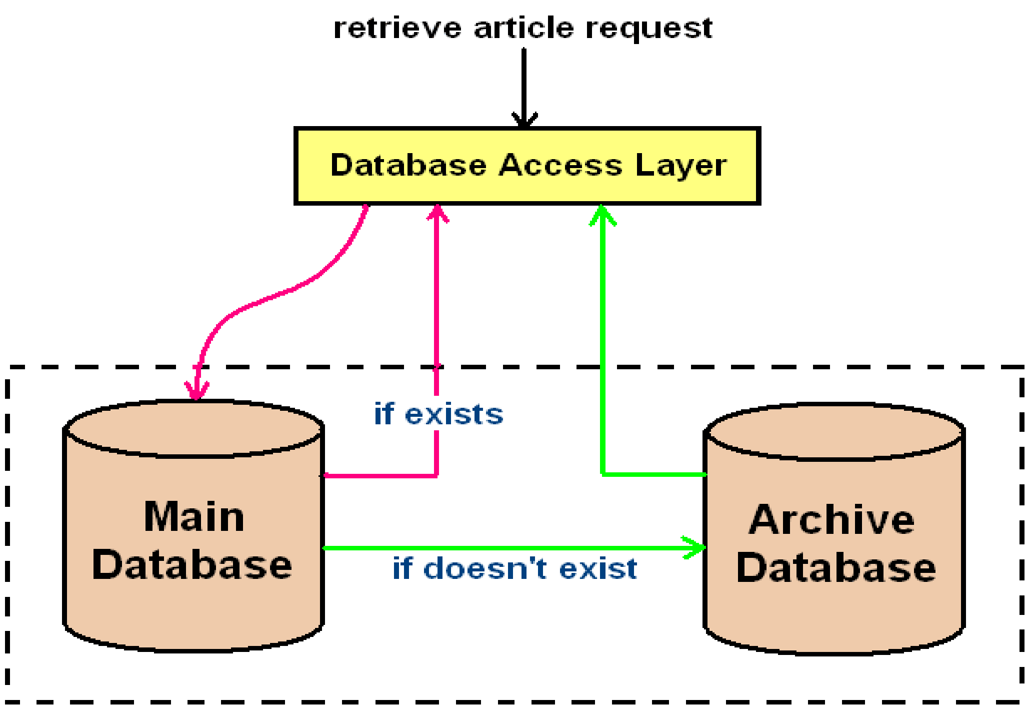
Send Message to the News Server: Mail users send messages to the server through SMTP protocol.

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

4 MODELING

4.1 Data Modeling

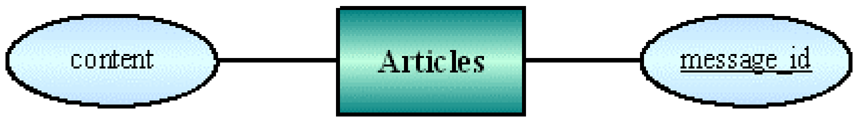
As you all know, in a unified news exchange server the data design and storage of the news, articles, newsgroups are the most important issues since the efficient access, consistent and stable data are really valuable. Moreover, the data design constructs the fundamentals of a system and the other parts or layers of the system are built on this basic structure. Therefore, in order to construct a consistent and a powerful system, one has to begin with a consistent data design. Keeping these in mind, we decided to store our data in database. In our system, we will store our data in 2 different databases. In the following figure, you can see how the mechanism works.

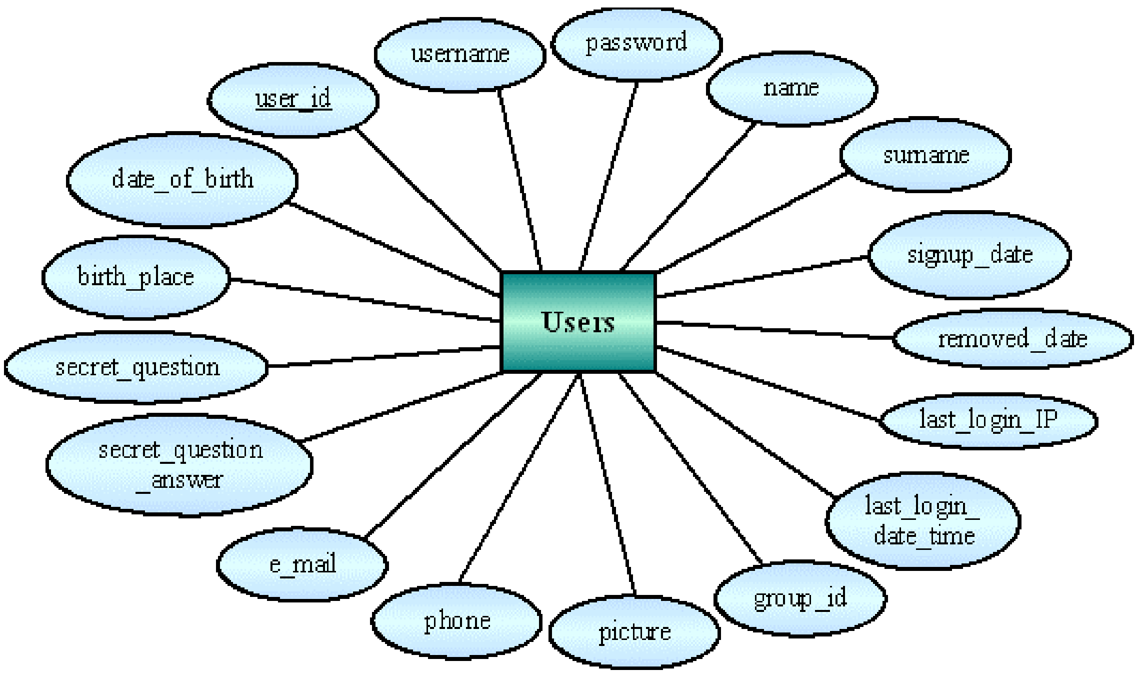


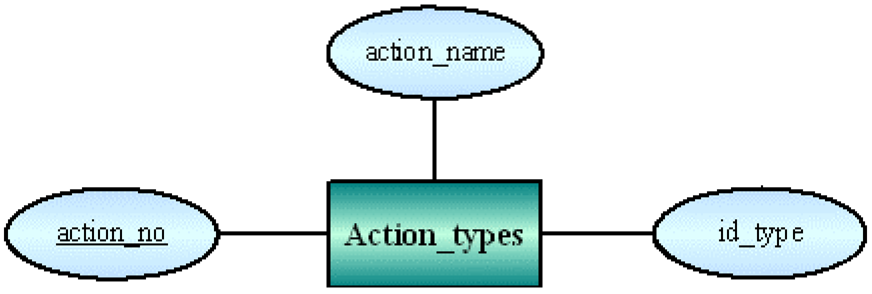
The main database will be used to store main data such as articles, users, newsgroups, etc. Other database will be used as an archive to store older articles and newsgroups. Also the relation between newsgroups and articles will be stored in another table. These older articles will not be stored in main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

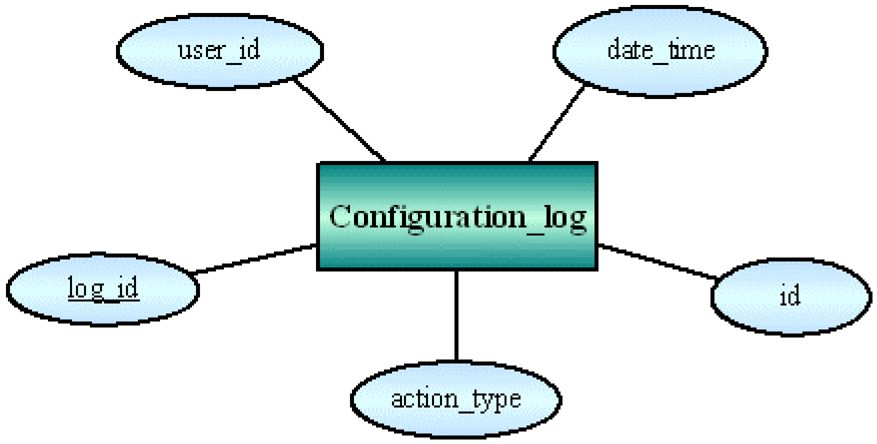
4.1.1 Entity-Relationship Diagrams

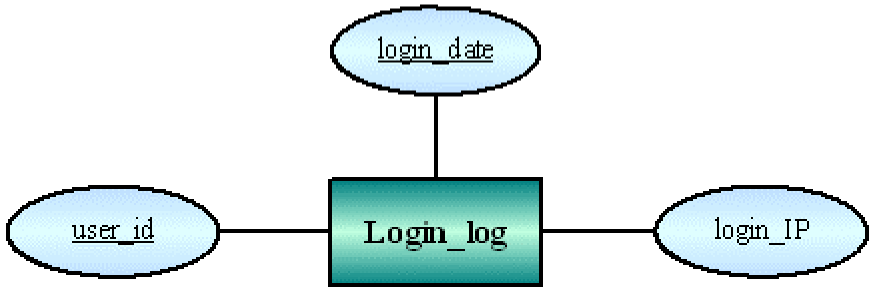
ER Diagrams For Main Database

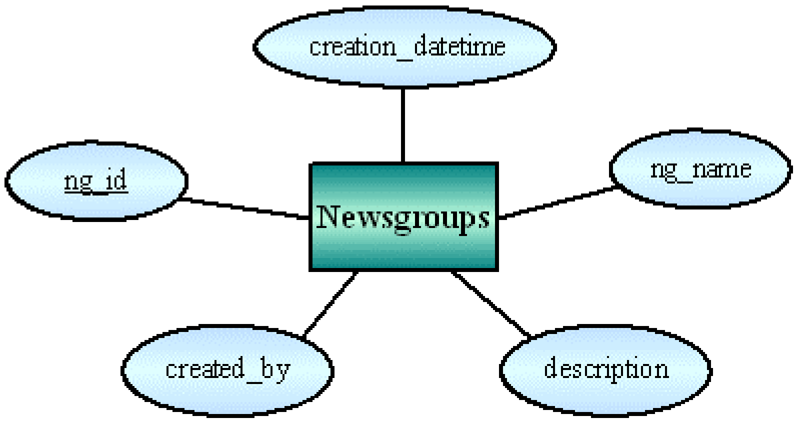


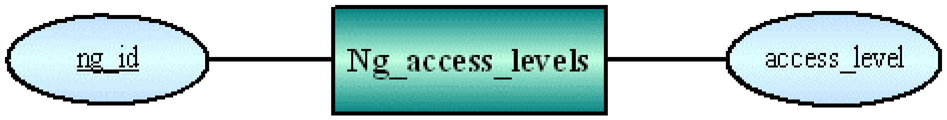

Users
user_id
username
password
name
surname
signup_date
removed_date
last_login_IP
last_login_ date_time
group_id
picture
phone
e_mail
secret_question _answer
secret_question
birth_place
date_of_birth

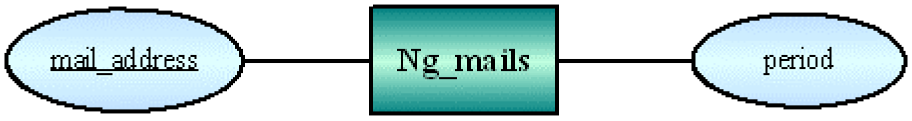

Action_types
action_name
action_no
id_type

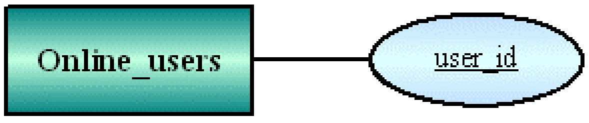

Configuration_log
user_id
date_time
log_id
id
action_type

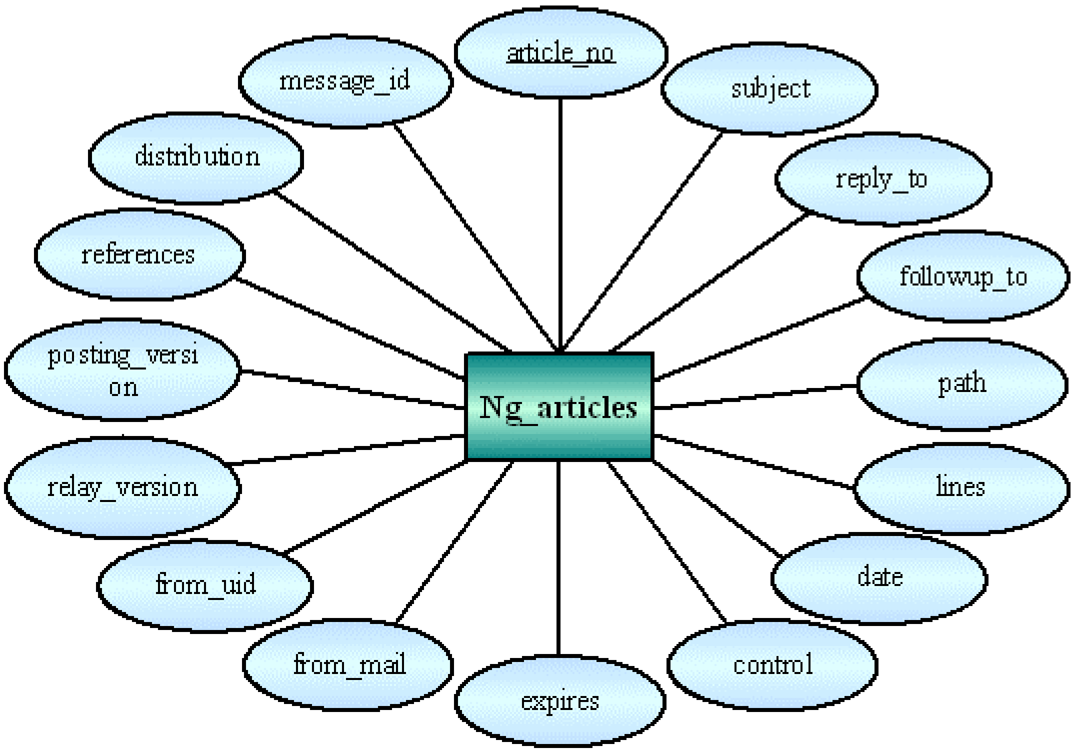

login_date
user_id
Login_log
login_IP

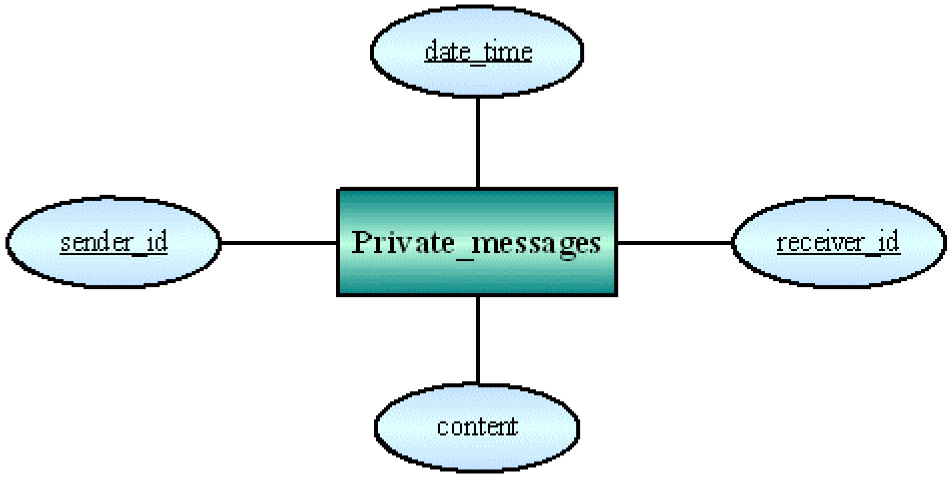

creation_datetime
ng_id
Newsgroups
ng_name
created_by
description

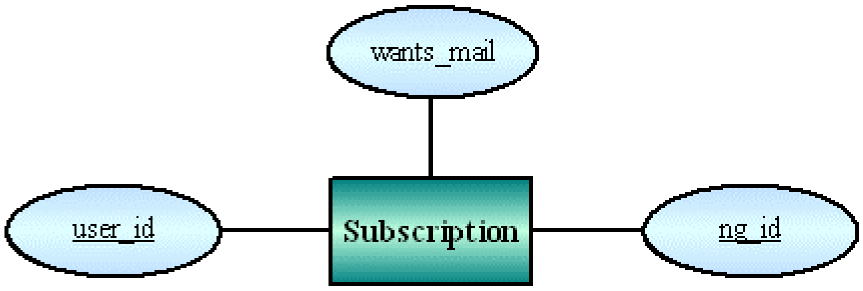

ng_id
Ng_access_levels
access_level

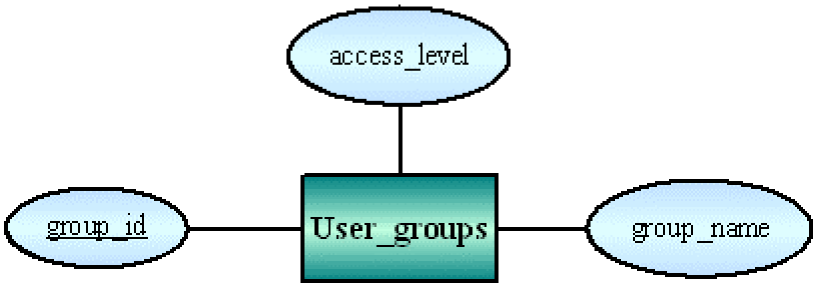

mail_address
Ng_mails
period


Online_users
user_id

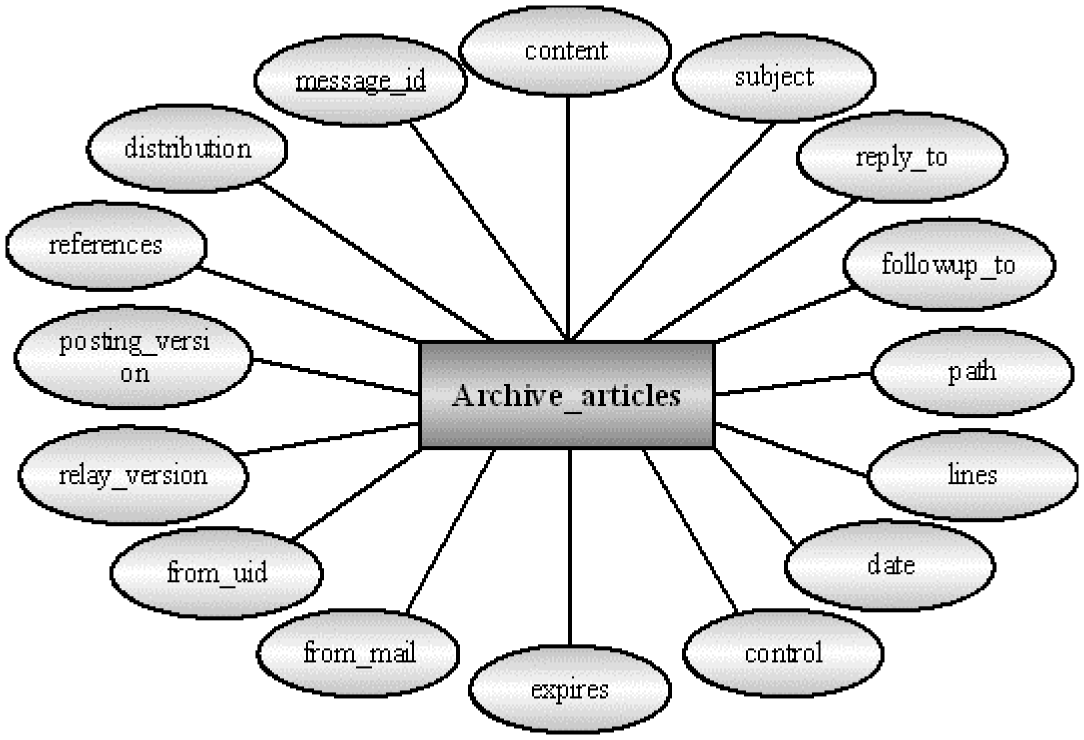

article_no
message_id
subject
distribution
reply_to
references
followup_to
posting_version
path
Ng_articles
relay_version
lines
date
from_uid
from_mail
control
expires

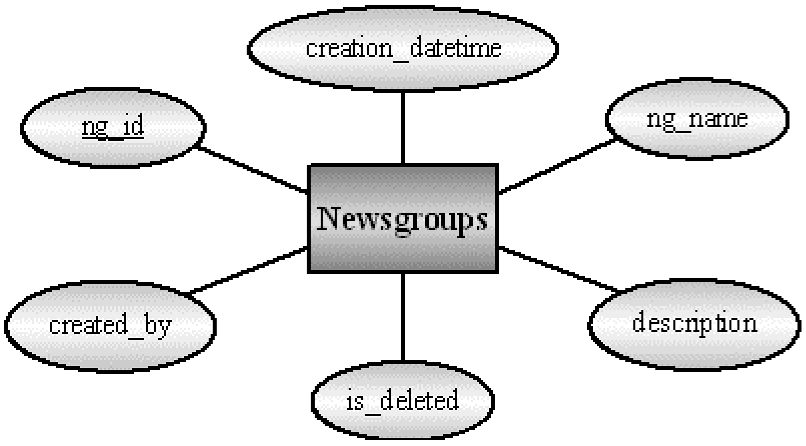

date_time
sender_id
Private_messages
receiver_id
content

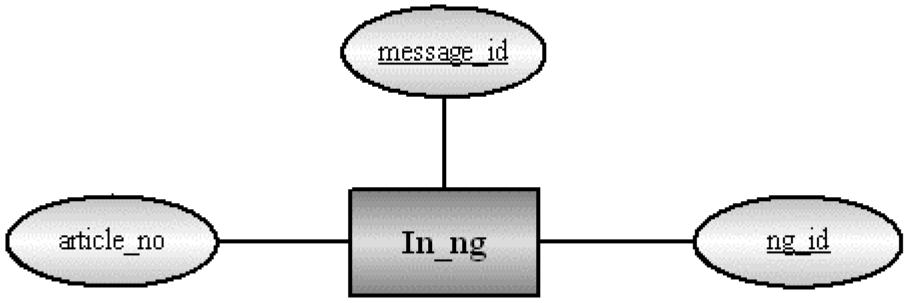

wants_mail
user_id
Subscription
ng_id



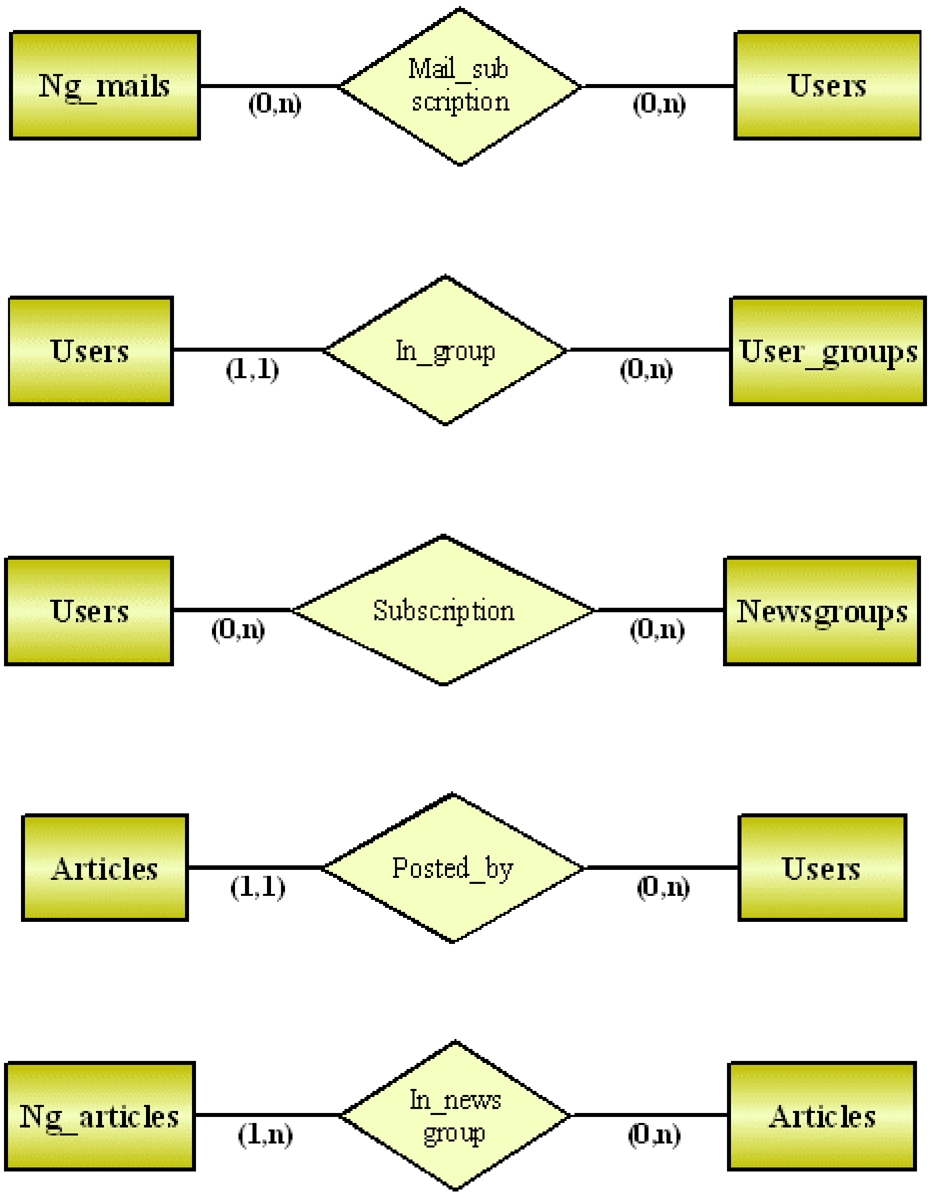
ER Diagrams For Archive Database





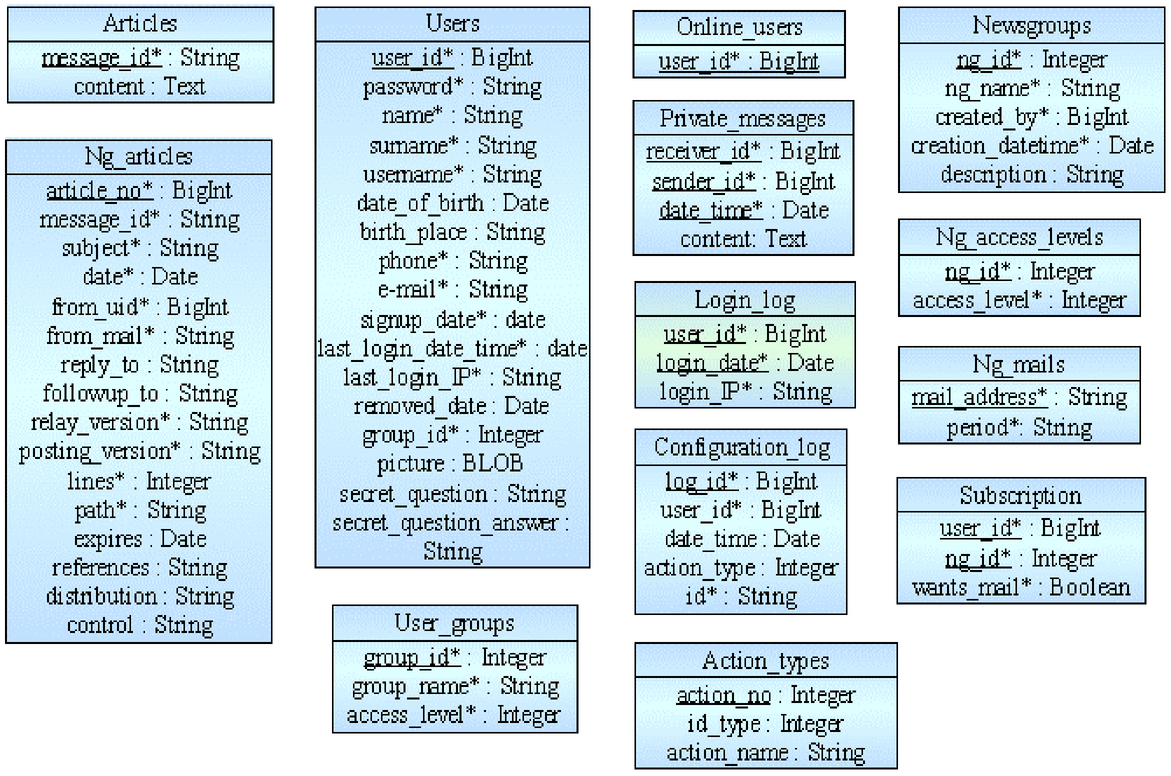

message_id
article_no
In_ng
ng_id

Relations

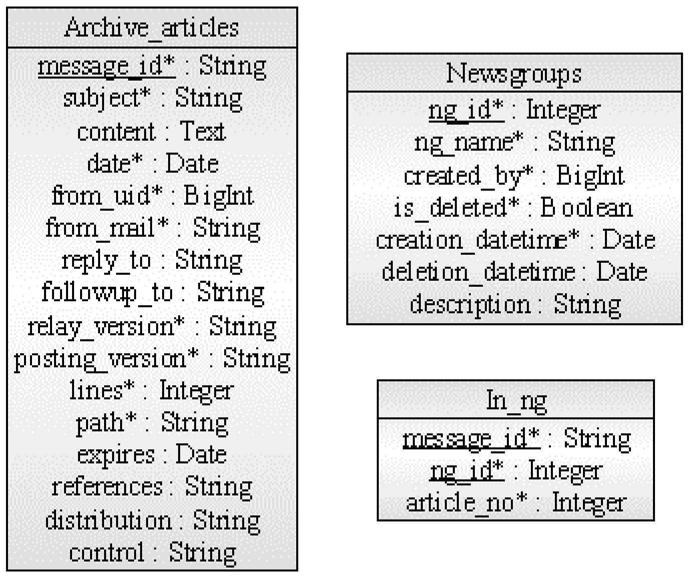

Ng_mails
(0,n)
Mail_sub scription
(0,n)
Users
Users
(1,1)
In_group
(0,n)
User_groups
Users
(0,n)
Subscription
(0,n)
Newsgroups
Articles
(1,1)
Posted_by
(0,n)
Users
Ng_articles
(1,n)
In_news group
(0,n)
Articles

4.1.2 Entity-Sets

Entity Sets For Main Database



Entity Sets For Archive Database



4.1.3 Data Descriptions

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

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

Data Descriptions for Main Database

Articles

Data	**Type & Size**	**Format**
message_id*	VARCHAR – 40	Text (UNIQUE)
content	TEXT	Text

Ng_articles

Data	**Type & Size**	**Format**
article_no*	BIGSERIAL	Number (AUTOINC)
message_id*	VARCHAR – 40	Text (UNIQUE)
subject*	VARCHAR – 60	Text
date*	DATETIME	Date/time
from_uid*	BIGINT	Number
from_mail*	VARCHAR – 40	Text
reply_to	VARCHAR – 40	Text
followup_to	VARCHAR – 40	Text
relay_version*	VARCHAR – 60	Text
posting_version*	VARCHAR – 60	Text
lines*	INTEGER	Number
path*	VARCHAR – 60	Text
expires	DATETIME	Date/time
references	VARCHAR – 60	Text
distribution	VARCHAR – 60	Text
control	VARCHAR – 60	Text

Users

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

user_id*	BIGSERIAL	Number (AUTOINC)
password*	VARCHAR – 20	Text is hidden. ********
name*	VARCHAR – 20	Text
surname*	VARCHAR – 20	Text
username*	VARCHAR – 40	Text (UNIQUE)
date_of_birth	DATE	Date
birth_place	VARCHAR – 20	Text
phone*	VARCHAR – 40	Text
e-mail*	VARCHAR – 40	Text
signup_date*	DATETIME	Date/time
removed_date	DATETIME	Date/time
group_id*	INTEGER	Number
picture	BLOB	Binary
last_login_IP*	VARCHAR – 20	Text
last_login_date_time*	DATE	Date
secret_question	VARCHAR – 40	Text
secret_question_answer	VARCHAR – 40	Text

User_groups

Data	**Type & Size**	**Format**
group_id*	INTEGER	Number
group_name*	VARCHAR – 60	Text
access_level*	INTEGER	Number

Newsgroups

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

Ng_mails

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

period	VARCHAR – 10	Text

Ng_access_levels

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

Subscription

Data	Type & Size	Format
user_id*	BIGINT	Number
ng_id*	INTEGER	Number
wants_mail*	BOOL	Yes/no

Login_Log

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

Action_Types

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

Configuration_Log

Data	Type & Size	Format
log_id*	BIGINT	Number (AUTOINC)
user_id*	BIGINT	Number
date_time*	DATETIME	Date/time
action_type*	INT	Number
id*	BIGINT	Number

Online_users

Data	Type & Size	Format
user_id*	BIGINT	Number

Private_messages

Data	Type & Size	Format
receiver_id*	BIGINT	Number
sender_id*	BIGINT	Number
date_time*	DATETIME	Date/time
content	TEXT	Text

Data Descriptions for Archive Database

Articles

Data	Type & Size	Format
message_id*	VARCHAR – 40	Text (UNIQUE)
subject*	VARCHAR – 60	Text
content	TEXT	Text
date*	DATETIME	Date/time
from_uid*	BIGINT	Number
from_mail*	VARCHAR – 40	Text
reply_to	VARCHAR – 40	Text
followup_to	VARCHAR – 40	Text
relay_version*	VARCHAR – 60	Text
posting_version*	VARCHAR – 60	Text
lines*	INTEGER	Number
path*	VARCHAR – 60	Text
expires	DATETIME	Date/time
references	VARCHAR – 60	Text
distribution	VARCHAR – 60	Text
control	VARCHAR – 60	Text

Newsgroups

Data	Type & Size	Format
ng_id*	INTEGER	Number

ng_name*	VARCHAR – 60	Text
created_by*	BIGINT	Number
is_deleted*	BOOLEAN	Yes/no
creation_datetime*	DATETIME	Date/time
deletion_datetime	DATETIME	Date/time
description	VARCHAR – 60	Text

In_ng

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

4.1.4 Entity Descriptions

Entity & Relation Descriptions for Main Database

Articles

In our main database, we store all articles in a single table. For each newsgroup we create a table and store overviews, not contents, of articles belonging to this newsgroup. When the user selects a newsgroup, required headers for threading will be retrieved from overview tables. This provides us to increase the system speed when our clients are connected to our system via newsreaders that support overview database. If the user clicks an article to see its contents, it is retrieved from main articles table. Keeping the content of articles in a single table prevents multiple holding of the same article in different newsgroup tables in case of cross posting. So, this entity contains message_id and content of the messages only.

message_id*: Required `Message-ID` standard header is held in string *message_id*. This attribute uniquely defines a message. The same message ID cannot be assigned to another article because this id is created by the clients according to their systems and merging this data with some information of the server.

content: This field is held in text format and stores the content of the article.

Ng_articles

Ng_articles is a general name for lots of possible tables. When a new newsgroup is created, an article table is created for that newsgroup with a specifying name. For example, if a group named `Music` is created, a table named `Music_articles` is also created. This table contains all necessary information (except content) about articles belonging to that table. This way is chosen in order to prevent the database from multiple storage of the content of same article when posted to different groups at the same time.

Some attributes are used for holding standard data for USENET messages and some attributes are assigned by us locally for managing articles easily.

In USENET message format, [6] there are some required headers and some optional headers. We hold these required headers and some of the optional headers in our database, in order to obey universal USENET message standards. Below, the table's attributes are explained.

article_no*: This number specifies each article in the group uniquely; hence article_no is the primary key of the *Ng_articles* entity. System assigns a unique number to each article in a newsgroup to manage them more easily.

message_id*: This field is also held with article number because news readers may want request any article by means of the universal message-ids. This is a foreign key referencing to the Articles table.

subject*: Required `Subject` standard header is held in string *subject*. It is assigned by sender and briefly defines what the article is about.

date*: Required `Date` standard header is held in *date* in date/time format. It is the time that the article is posted to the network.

from_uid*: This is a local assignment that is required to know which user has posted the article. It is a foreign key for this entity referencing *user_id* of *Users* entity.

from_mail*: Required `From` standard header is held in string *from_mail*. It is the mail address of the sender of that article. This is a default mail address and foreign key which references the attribute *e-mail* of *Users* entity.

reply_to: Optional `Reply-To` standard header is held in string *reply_to*. This string holds the optional mail address of the sender if he/she wants to get mail for that article to the specified address instead of *from_mail*.

followup_to: Optional `Followup-To` standard header is held in string *followup_to*. If this is not empty, all follow-ups to the article will be posted to the newsgroups specified in this field. If it is empty, follow-ups will be posted to the newsgroup(s) that the message was originally posted.

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

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

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

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

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

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

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

Users

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

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

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

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

username*: This string field holds the username of the user, it is unique for each user.

password*: This string field is the matched password for the username of the user .
date_of_birth: This date typed attribute holds the birth date of the user.
birth_place: This string typed attribute holds the birth place of the user.
phone*: This string field holds the cell phone number of the customer.
e-mail*: This text field holds the mail address of the customer.
signup_date*: This field holds the date and time that the user has signed up. This field is of type date/time.
removed_date: This field is usually empty but if a user is removed from the database, this field holds the date and time that the user is removed from the system.
group_id*: Group id specifies which user group the user belongs to. This is a foreign key referencing *group_id* attribute of *User_groups* entity.
picture: Users can upload their pictures to the system. This picture is held in *picture* field in BLOB format.
last_login_date_time: Date and time of last login of the user is kept for security.
last_login_IP: IP of the computer that the user last logged in is also kept.
secret_question: Secret question is kept in string format. It is asked in a case that the user forgets his/her password.
secret_question_answer: Secret question's answer is kept in string format also. It is used in a case that the user forgets his/her password.

User_groups

This entity holds information about user groups. Each user will be a member of a pre-determined user group. Each user group will have an access level. These access levels will be used to determine whether a user will be able to access a specified newsgroup or not. Since administrators will be treated in the same manner with other users, there is no need to create a distinct administrator table. Administrative rights will be determined by user groups.
group_id*: This number specifies each user group uniquely; hence *group_id* is the primary key of the *User_groups* entity.
group_name*: This string field holds the name of the usergroup.

access_level*: This integer field holds the access level of the user. For instance, if it is 1, it means full access.

Newsgroups

This entity holds information about newsgroups. When a newsgroup is added, listed information about that group is added to the table.

ng_id*: This number specifies each newsgroup uniquely; hence n*g_id* is the primary key of the *Newsgroups* entity.

ng_name*: This string field holds the name of the newsgroup.

created_by*: This big integer typed field holds information about who created this newsgroup. This is a foreign key of this entity referencing *user_id* attribute of *Users* entity.

creation_datetime*: This field holds the date and time that the newsgroup is created. This field is of type date/time.

description: This string field holds a brief description about what the newsgroup is about.

Ng_mails

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

mail_address*: This string field holds the mail addresses of the users who want to receive e-mails from the specified newsgroup.

period*: Period is a string and it is chosen by the user among some specified periods by us. These periods may be ‘weekly’, ‘daily’, etc. If the user doesn’t want to receive mails when the article is posted, he/she, for example chooses ‘weekly’ as a period. Then the articles will be sent to the user weekly.

Ng_access_levels

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

ng_id*: This field is the id specifies the newsgroups uniquely. This is a foreign key for this relation referencing *ng_id* attribute of *Newsgroups* entity. ng_id, itself, is the primary key of this table, since each newsgroup will be stored once in this table.

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

Subscription

This table specifies a relation among users and newsgroups. Users can be subscribed to newsgroups. Required information about this subscription is held in this table.

user_id*: This field is the id of the user who subscribed to the newsgroup. This is a foreign key for this relation referencing *user_id* attribute of *Users* entity. This field is a subset of primary key.

ng_id*: This field is the id of the newsgroup which is subscribed by the user. This is a foreign key for this relation referencing *ng_id* attribute of *Newsgroups* entity. This field is also a subset of primary key.

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

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

Login_Log

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

user_id*: This number specifies each user uniquely. This is a foreign key referencing to the Users table.

login_datetime*: This timestamp attribute stores the date and time of the login.
login_IP*: This attribute stores the ip address of the computer that user logged in to the system.

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

Action_Types

This table, in fact, is stored for specifying the configuration actions of users which are stored in configuration_log table. In fact, this table is mostly a static table, since there will be no major change on this table when all action types have already been specified. Only a small number of insertions, deletions and updates may be applied on this table when an action type will be inserted, deleted or updated, respectively.
action_no*: This number specifies each action type uniquely. Action_no is the primary key of this table. It will be auto incremented when a action is inserted to this table.
id_type*: This attribute specifies one of article_no, user_id, ng_id. This id is the specification for on which type of data, the configuration can be done.
action_name*: action_name is just an attribute to specify the name of the action_type. For instance, update of article may be a possible name for an action_name.

Configuration_Log

This table stores all configurations of users on database. When an insertion, deletion or update is done, a row is inserted to the configuration_log table. Like login_log table, this information is significant for security reasons. Storing configuration actions data in the database provide us to control the configurations done on database by each user and when this configuration is done.
log_id*: This field specifies the configuration log uniquely. It is the primary key of the entity and incremented automatically.
user_id*: This attribute specifies the user who does the configuration. This user_id is a foreign key to the Users table.
log_datetime*: This timestamp attribute stores the date and time of the configuration.

action_no*: This integer stores the information of which configuration is done by the user specified by user_id attribute. Since action types table stores all actions can be applied by users, this attribute is a foreign key to action_types table.
id*: id attribute stores the id of the message, newsgroup or user on which configuration is done. Since action_no table is storing whether the configuration is applied on a message, a newsgroup or a user, it is easy to determine the id is related with whether a message, a newsgroup or a user. By using this id and other attributes of this table, a config_log tuple can easily be created.

Online_users

When a user logs in, id of that user is inserted into this table. When that user logs out, the id is deleted from the table. We will show online users in our web module.
user_id*: This is the primary key and references the user_id field in users table.

Private_messages

Users will be able to send messages to other users. For each user, we will store messages that are sent to him/her. User will be able to see the messages when he/she logs in to the system. After reading the message, user can reply to that message. We will display message history to the users.
receiver_id*: This is the user_id of the user to whom the message is sent.
sender_id*: This is the user_id of the user who sent the message.
date_time*: When the message is sent, the date and time of the message will be hold in date_time field in date format.
content: This attribute holds the content of the private message.

Entity Descriptions for Archive Database

We are supposed not to delete old articles. As a result of this, after a period, there will be a great deal of articles and the database will begin to be congested. In such a situation, database access and retrievals will be slow. For this reason, we came up with a decision of archiving old articles. We have an archiving criterion based on article load. For each newsgroup, when a specific article load is exceeded, we archive some amount of articles for that group. This criterion can differ for different newsgroups. We will keep an archive database and store the archived articles there. If the user wants to retrieve an archived article, the content of the article will be retrieved from archive database. However, retrieving an archived article will be a rare operation and most of the operations will access our main database which will be faster after archiving mechanism.

Articles

This entity contains all necessary information about archived articles which are posted to the news server. This information is the ones that are kept in ng_mails table in main database, plus the content of the message.

Newsgroups

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

In_ng

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

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

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

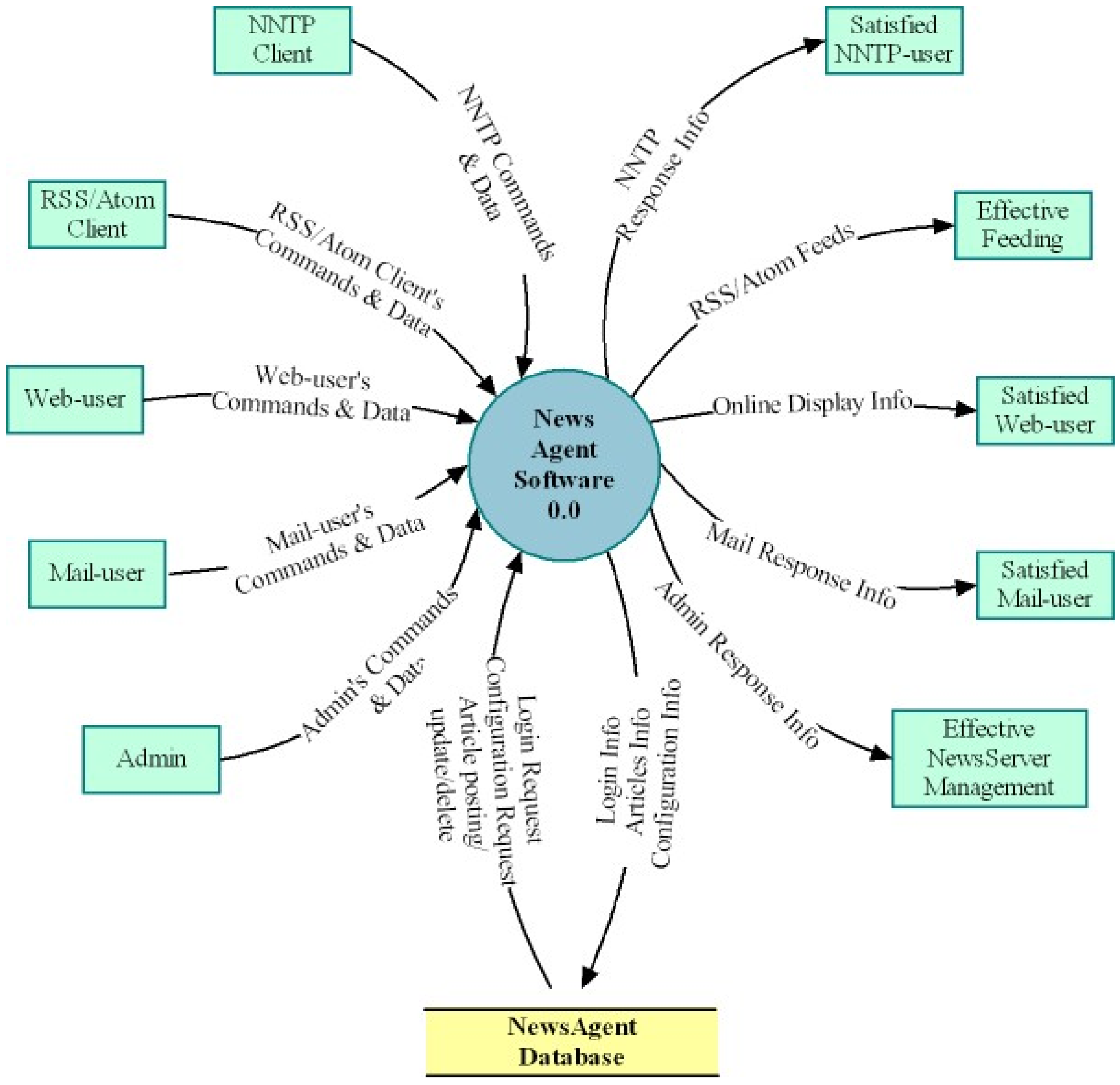
ng_id*: This field is a foreign key for this relation referencing *ng_id* of *Newsgroups* entity. It defines which newsgroup the message belongs to.

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

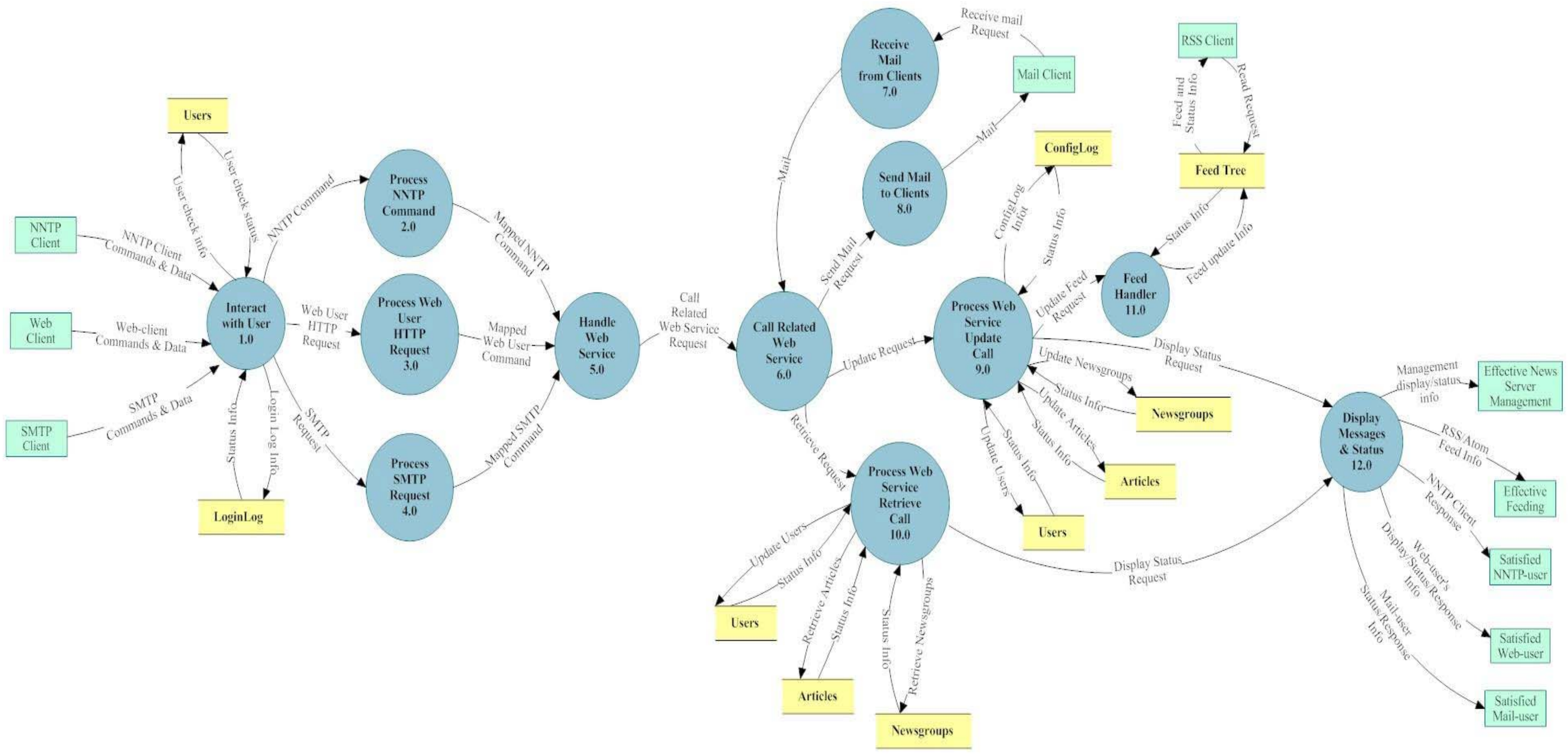
4.2 Functional Modeling

4.2.1 Data Flow Diagrams

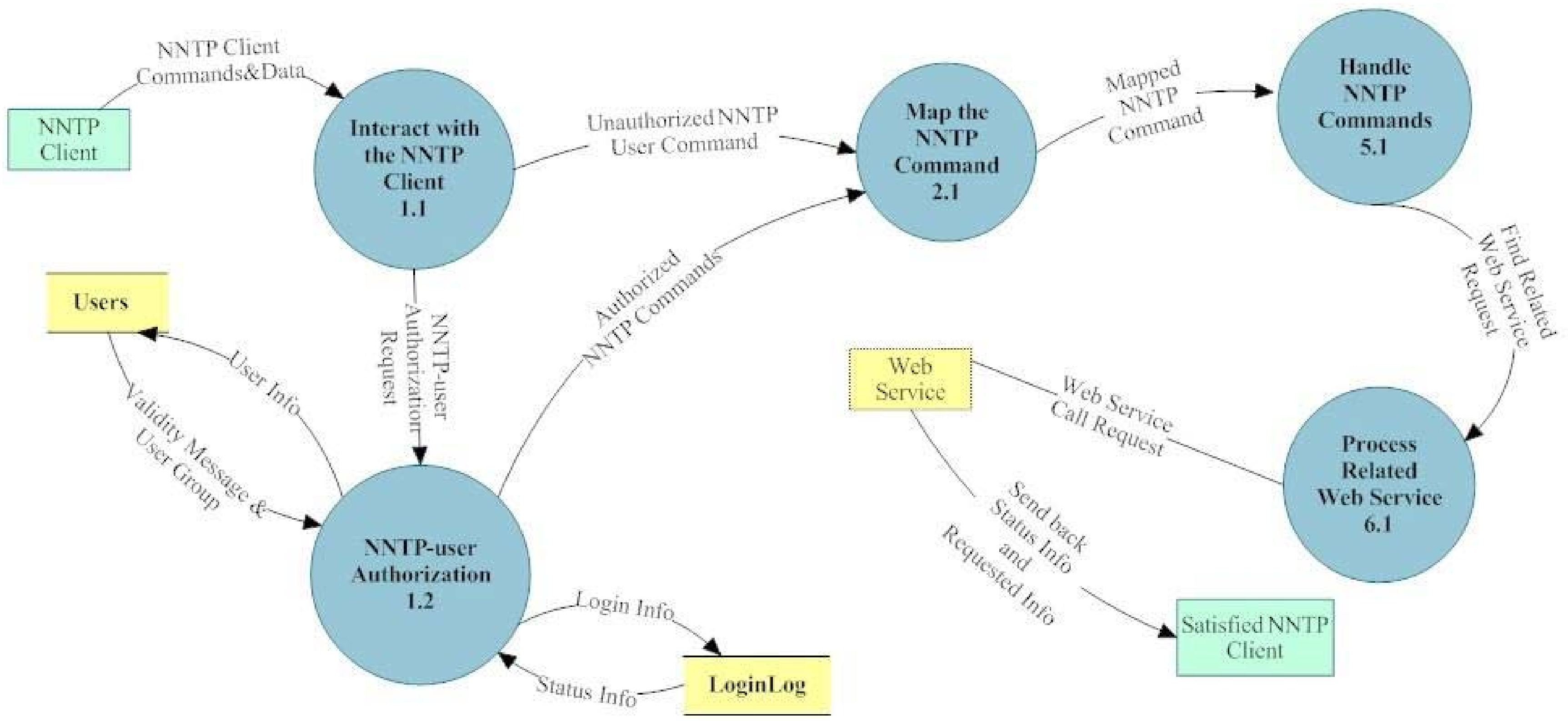
4.2.1.1 LEVEL 0 DATA FLOW DIAGRAM

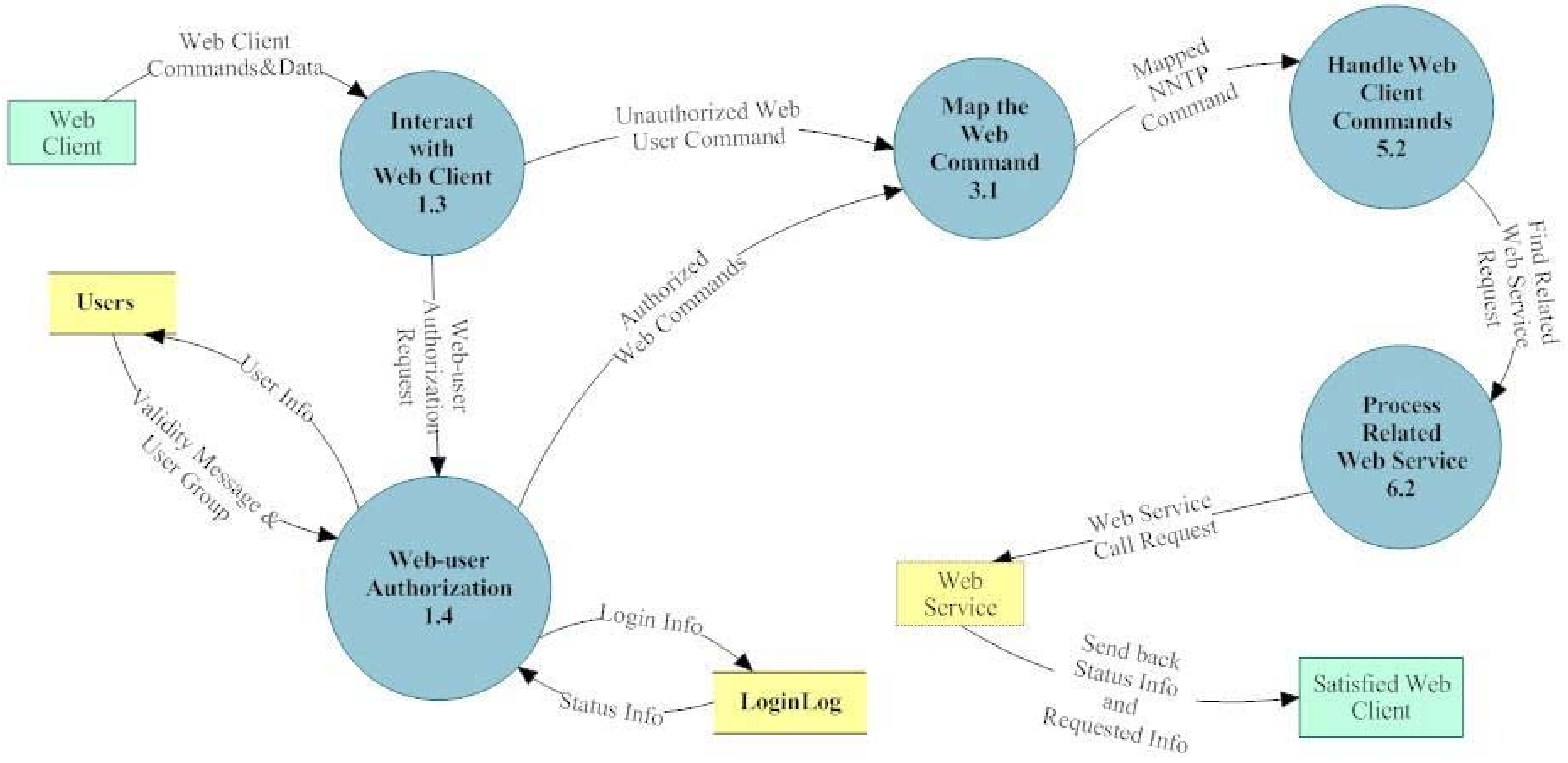


4.2.1.2 LEVEL 1 DATA FLOW DIAGRAM

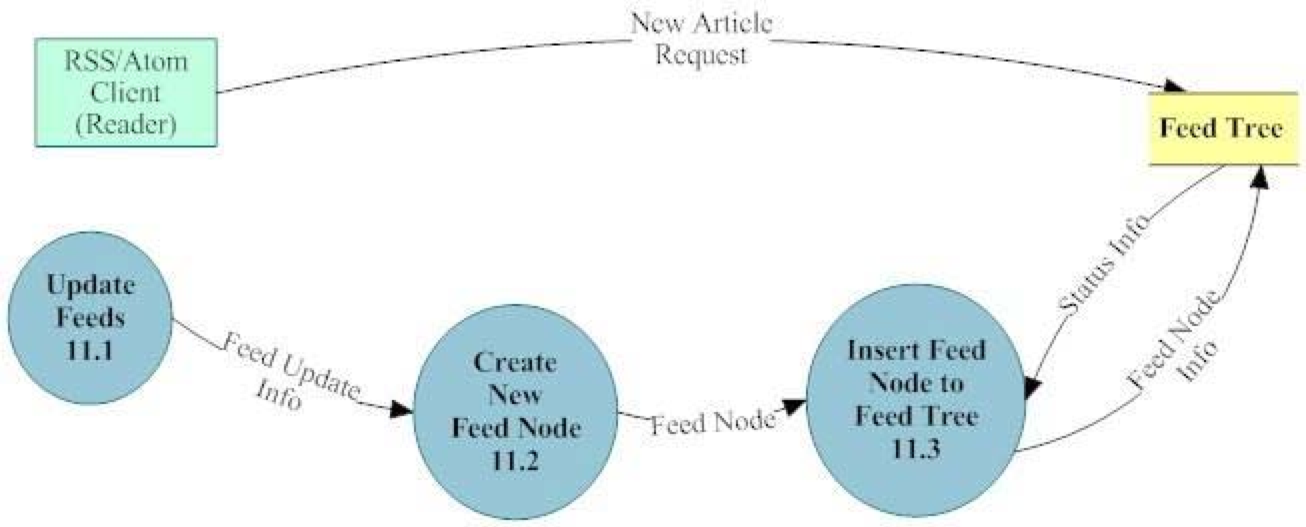


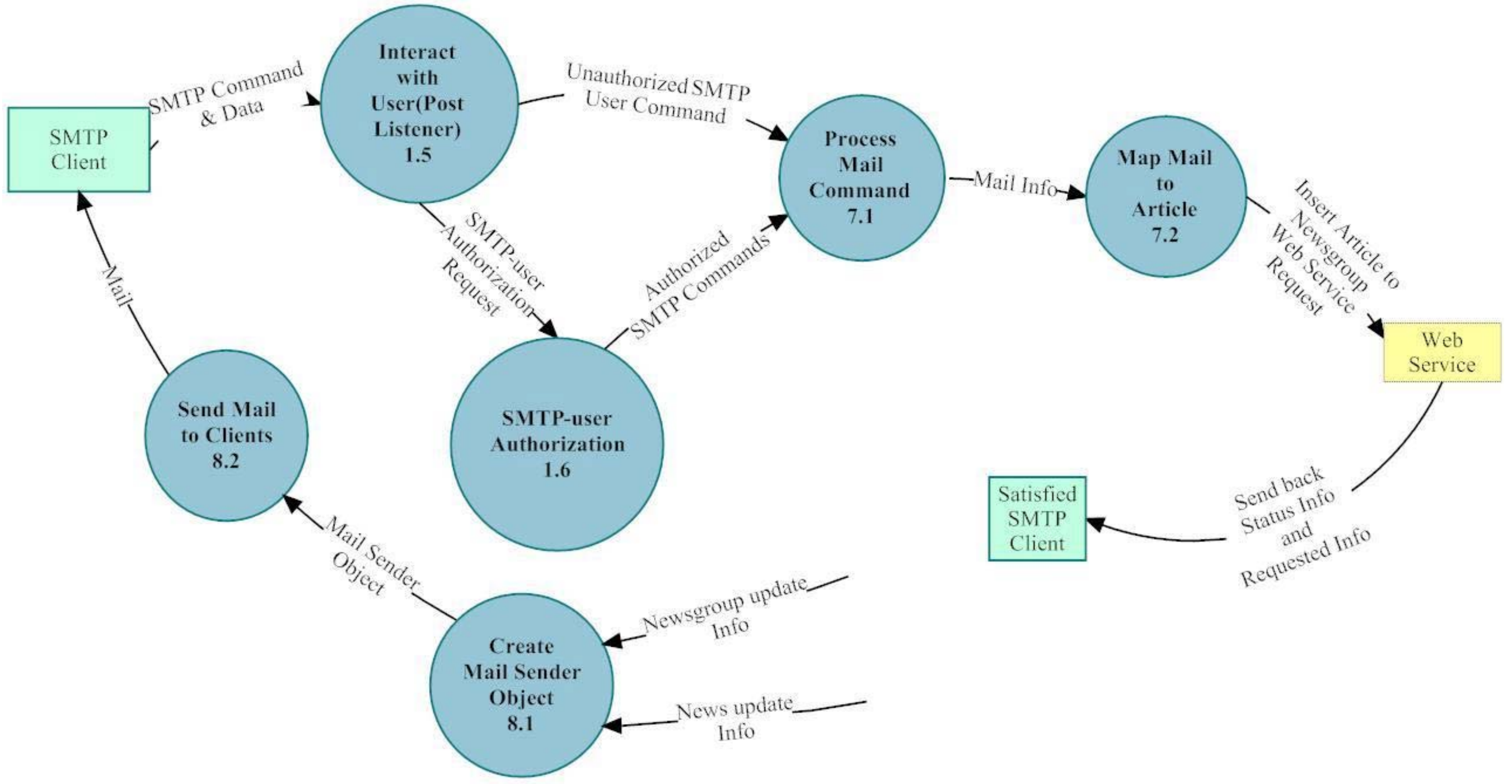
4.2.1.3 LEVEL 2 DATA FLOW DIAGRAMS

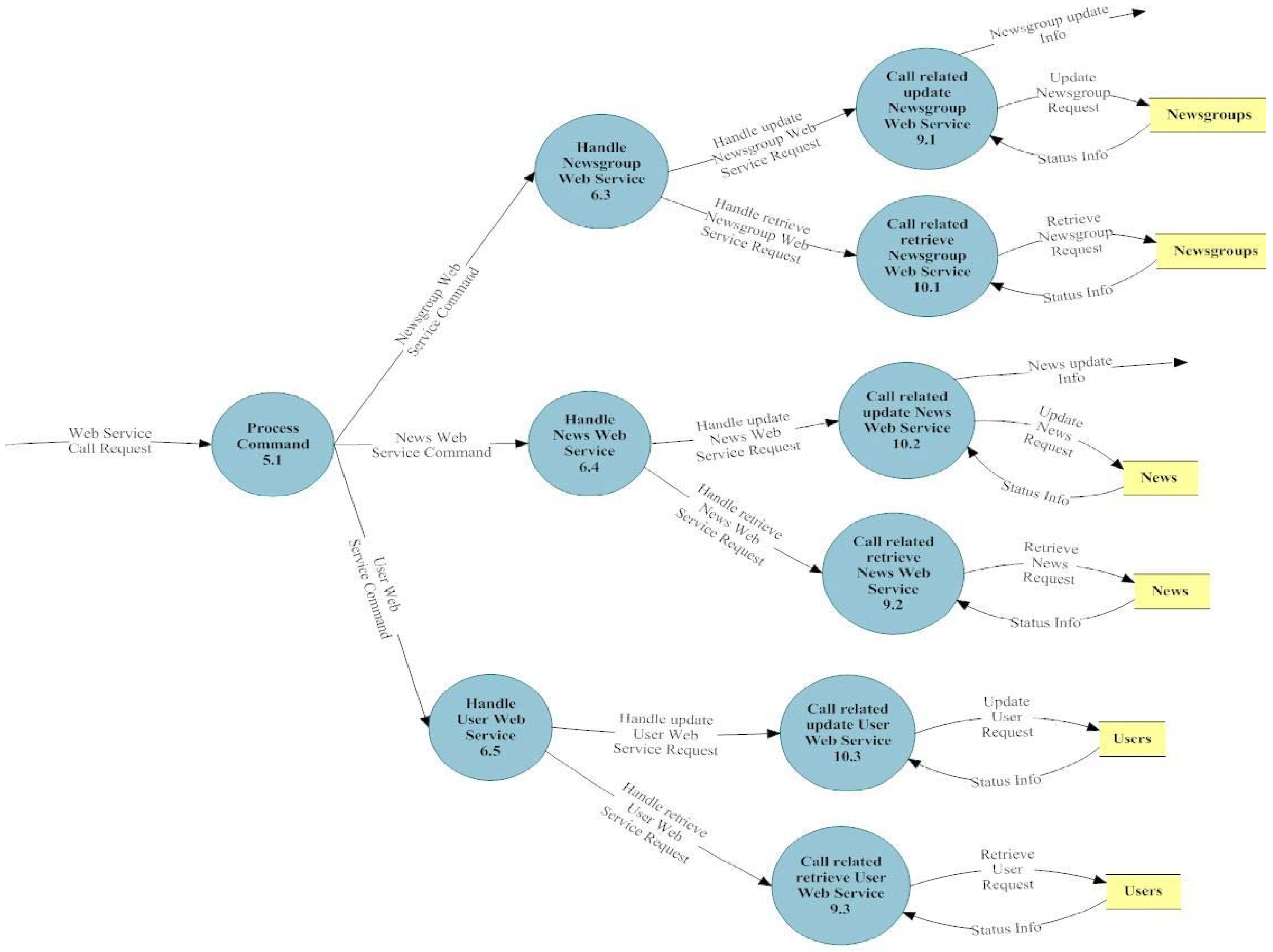

NNTP Client
NNTP Client Commands&Data
Interact with the NNTP Client 1.1
Unauthorized NNTP User Command
Map the NNTP Command 2.1
Mapped NNTP Command
Handle NNTP Commands 5.1
Find Related Web Service Request
Users
User Info
Validity Message & User Group
NNTP-user Authorization Request
Authorized NNTP Commands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



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


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


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


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

4.2.2 Process Specifications (PSEPC)

4.2.2.1 PSPECs for NNTP Module

PSPEC : Interact with NNTP Client

This process controls interaction for users who want to reach articles through NNTP Module. These users are people who uses e-mail and news client software packages. When interacting with NNTP Client, NNTP commands and data will be handled and these data will be sent to the NNTP User Authentication process. However, if user wants to reach articles which can be accessible by unauthenticated users and did not send authentication data, he/she will be able to access newsgroups which have access level providing unauthenticated user accesses and their articles.

PSPEC : NNTP User Authentication

Corresponding to the information sent from Interaction with NNTP Client, in this process database access is handled for a control of username and password to specify user group of the user. After the control, the result action can be authenticated user authentication or rejection of user authentication data. User may send authentication data again if user authentication command was rejected or he/she may act as an unauthenticated (if user wants to reach articles which can be accessible by unauthenticated users and did not send authentication data.) or authenticated user (if user authentication data has been already accepted.). According to the result of authentication process, LoginLog table is updated and its return status is handled. Unauthenticated NNTP users should specify their names and passwords as *anonymous*; however that is not the case for web users.

PSPEC : Map NNTP Command

NNTP User Commands are sent from authenticated or unauthenticated users and these commands are mapped to predefined NNTP commands. For instance, when user wants to post an article to a newsgroup, its mapped command will be sent to Handle NNTP Commands process. By having a Map NNTP Command process, a modular design is established for handling NNTP Commands.

PSPEC : Handle NNTP Command

Mapped NNTP Commands are handled by this process. According to the mapped command retrieved, related web service is called.

PSPEC : Process Related Web Service

Since mapped NNTP Command has already been determined by Map NNTP Command process and its related web service has already been determined by Handle NNTP Command

process, it is not a big deal to processing related web services. Detailed explanations about Web Service processes are in Web Service processes part. In short, Web Service processes handle each web service and by this way, the core of NewsAgent is accessed via web services. Modularity is the main point for having such a Web Service processes.
Processing related web services (corresponding to the commands of NNTP user) ends NNTP Module with a satisfied NNTP User.

4.2.2.2 PSPECs for Web Module

PSPEC : Interact with Web Client

This process controls interaction for users who want to reach articles through Web Module. These users are people who uses NewsAgent web user interface. When interacting with Web Client, user commands and data will be handled and these data will be sent to the Web User Authentication process. However, if user wants to reach articles without any authentication process, he/she will be able access some newsgroups specified as accessible without authentication (in fact, access levels of newsgroups are specifications).

PSPEC : Web User Authentication

Corresponding to the information sent from Interaction with Web Client, in this process database access is handled for a control of username and password to specify user group of the user. Validity message or invalid user data will be returned from database access. If the validity message is returned from database access, user access level is also returned to specify to which groups will be accessible for user. After the control (according to the validity message), the result action can be authenticated user authentication or rejection of user authentication data. User may send authentication data again if user authentication command was rejected or he/she may act as an unauthenticated (if user wants to reach articles which can be accessible by unauthenticated users and did not send authentication data.) or authenticated user (if user authentication data has been already accepted.). According to the result of authentication process, LoginLog table is updated and its return status is handled.

PSPEC : Map Web Command

Web User Commands are sent from authenticated or unauthenticated web users. Users will send their commands by using the web interface of NewsAgent. For instance, when user wants to list articles of a newsgroup, he/she should click on the name of newsgroup from the list of all newsgroups. After the specification of web user command, its mapped command will be sent to Handle Web Client Commands process. By having Map Web Command process, a modular design is established for handling Web Client Commands.

PSPEC : Handle Web Client Command

Mapped Web Client Commands are handled by this process. According to the mapped command retrieved, related web service is called.

PSPEC : Process Related Web Service

Since mapped Web Client Command has already been determined by Map Web Command process and its related web service has already been determined by Handle Web Client Command process, it is not a big deal to processing related web services. Detailed explanations about Web Service processes are in Web Service processes part. In short, Web Service processes handle each web service and by this way, the core of NewsAgent is accessed via web services. Modularity is the main point for having such a Web Service processes.

Processing related web services (corresponding to the commands of NNTP user) ends NNTP Module with a satisfied NNTP User.

4.2.2.3 PSPECs for RSS/ATOM Module

Feed Updates are handled by this module. As mentioned earlier, NewsAgent will have feed trees for each newsgroup and users will be able to subscribe each of them according to their user groups. After updated RSS/ATOM readers will be able to retrieve updated article or newsgroup information.

PSPEC : Update Feeds

Update feeds is the start process for updating feed trees. When there is a post, delete or in general term an update on a newsgroup or article, update feeds process is started and necessary update information is supplied to this process.

PSPEC : Create New Feed Node

When necessary information for an update is supplied by Update Feeds process to Create New Feed Node, it creates a new free (not bound to any feed tree) feed node for insertions to feed trees of different newsgroups.

PSPEC : Insert Feed Node to Feed Tree

After the creation of a new free feed node by Create New Feed Node process, feed node is ready to be inserted to feed trees of newsgroups. Insert feed node process establishes a connection to feed trees for newsgroups to which new feed node will be inserted. For each feed tree that the new feed node will be inserted to, this process sends all data related with the created free feed node and the newsgroup specification (for specifying to which feed tree the feed node will be inserted to). After the insertion of the new feed node, status information is

handled again by this process. This design of Insert Feed Node to Feed Tree process is, in fact, so useful to handle cross-posting.
After all processes of RSS/ATOM module, when a user requests the feed of any newsgroup, he/she will be able to get an updated version of feeds by the help of an RSS/ATOM reader.

4.2.2.4 PSPECs for SMTP Module

In fact, this module consists of two sub-modules, one for sending mails to mail-users and one for receiving mails from mail-users. By using this module mail users will be able to post an article to newsgroups and receive articles that are posted to newsgroups via e-mail from NewsAgent server.

PSPEC : Interact with User (Port Listener)

This process interacts with mail user and when there is a new e-mail sent to any newsgroup of NewsAgent, port listener will handle it. In fact, since NewsAgent will use James SMTP Server, this will be handled by it.

PSPEC : SMTP User Authentication

SMTP User Authentication process gets SMTP User Authentication request from James SMTP Server and sends a new e-mail to the sender to verify whether the sender is correct or not. After the verification of the sender, mail can be posted to the related newsgroups as articles. By this way, spams will not be posted as articles to newsgroups and this will be a significant point for security. In fact, verification step makes SMTP User Authentication process different from authentication in other modules.

PSPEC : Process Mail Command

After the authentication of mail-user, commands will be produced for converting the mail to article format and sending it to specified newsgroups. Unauthenticated users will also be able to send mail to any of the newsgroups and again verification step will be handled for them.

PSPEC : Map Mail to Article

After the specification of commands, mail should be mapped to article. By this way, mail will be converted to article format and after that point mail will be sent to newsgroups as if it was simply an article. Since it will be handled as an article related web services will be called to insert the article to specified newsgroups.

PSPEC : Create Mail Sender Object

When necessary information for an update is supplied by Web Service processes to Create Mail Sender Object process, it creates a new Mail Sender object and this object will be passed to Send Mail to Clients process. Coming data from Web Services part are explained in Web Service process part in a detailed manner.

PSPEC : Send Mail to Clients

Creation of a Mail Sender Object is necessary before the application of this process. Since it has already been done by Create Mail Sender Object process, after a control from the database for mail users of the newsgroups to which the new article is sent, by using Mail Sender Object, a mail is created (content of the mail can be retrieved from Mail Sender Object and receipants are retrieved from the database.) and sent. By this way, a mail user will be able to receive posts to newsgroups that he/she has subscribed beforehand.

Since user will be able to send mails to newsgroups of NewsAgent and receive new articles via e-mail, mail-users will have most of the opportunities that Web or NNTP users have.

PSPECs for Web Service Processes

As explained in the process specifications of modules, when NewsAgent core will be accessed, this will be done by the help of web services. This provides modularity in NewsAgent.

PSPEC : Process Command

Web services will be called by processes according to the command that should be processed. Process Command process is gate keeper for accessing web services. According to the request it diverts data and command to related web services. There are three Web service sub-modules; newsgroup, news, user which access newsgroup, article and user data respectively to retrieve, insert or modify specified data in the command and data attached to it.

PSPEC : Handle Newsgroup Web Service

This process handles web services related to newsgroups. When an update or retrieval on/from Newsgroups and its related tables on the database, Handle Newsgroup Web Service will be activated by Process Command process. Depending on whether the data will be retrieved or updated, it diverts command and data to one of the processes named as Call Related Update Newsgroup Web Service and Call Related Retrieve Newsgroup Web Service.

PSPEC : Call Related Update Newsgroup Web Service

Updates on Newsgroups table will be done through this process. For instance, when name of a newsgroup will be changed, this process will handle the connection to the database and will make the specified change on Newsgroups table. In fact, it will be reasonable to update some related data in other tables according to the updates on Newsgroups table such as ConfigLog. In addition to that, when name of a newsgroup is changed, newsgroup name for Ng_articles and Ng_mails will be changed. Also, after an update on a newsgroup, this should be reported to mail-users and RSS users. NewsAgent server will send mails to mail-users of the updated newsgroup (Note that data named as Newsgroup Update info exist also in DFD for SMTP Module). For reporting the update to RSS users, an article will be sent automatically to a specific newsgroup (such as *newsagent.announce.admin*), by this way, users who are subscribed to this newsgroup will be informed about the change.

PSPEC : Call Related Retrieve Newsgroup Web Service

Retrieves from Newsgroups table will be done through this process. For instance, when articles of a newsgroup will be listed, this process will handle the connection to the database and will retrieve the specified data from Newsgroups table. Since retrieval will not modify any data about newsgroups there is no need to handle cases in Call Related Update Newsgroup Web Services.

PSPEC : Handle News Web Service

This process handles web services related to articles. When an update or retrieval on/from Articles and its related tables on the database, Handle News Web Service will be activated by Process Command process. Depending on whether the data will be retrieved or updated, it diverts command and data to one of the processes named as Call Related Update News Web Service and Call Related Retrieve News Web Service.

PSPEC : Call Related Update News Web Service

Updates on Articles table will be done through this process. For instance, when a new article is posted to a newsgroup, this process will handle the connection to the database and will make the specified change on Articles table. In fact, it will be reasonable to update some related data in other tables according to the updates on Articles table such as Configuration_Log. In addition to that, when a new article is posted to any newsgroup, a new tuple should be inserted to table Ng_articles (for related newsgroups only, of course). Also, after a new article is posted, this should be reported to mail-users and RSS users. NewsAgent server will send articles to mail-users via e-mail (Note that data named as News Update info exist also in DFD for SMTP Module). Since article will be added to feed trees of specified newsgroups, RSS users will easily access new posted articles.

PSPEC : Call Related Retrieve News Web Service

Retrieves from News table will be done through this process. For instance, when article content will be retrieved, this process will handle the connection to the database and will retrieve the specified data from Articles table. Since retrieval will not modify any data about articles there is no need to handle cases in Call Related Update News Web Services.

PSPEC : Handle User Web Service

This process handles web services related to users. When an update or retrieval on/from Users and its related tables on the database, Handle Users Web Service will be activated by Process Command process. Depending on whether the data will be retrieved or updated, it diverts command and data to one of the processes named as Call Related Update Users Web Service and Call Related Retrieve Users Web Service.

PSPEC : Call Related Update User Web Service

Updates on Users table will be done through this process. For instance, when a new user is added, this process will handle the connection to the database and will insert data about the user to Users table. In fact, it will be reasonable to update some related data in other tables according to the updates on Users table such as ConfigLog.

PSPEC : Call Related Retrieve User Web Service

Retrieves from Users table will be done through this process. For instance, a user wants to see his/her account information details this process will establish the database connection and retrieval will be performed.

In general, Web Service Processes is the heart of NewsAgent, since it is the only way to access to database. That is why it is accessible from each module. According to the result of any retrieval or modification by any web service, status information will be returned and according to that some other actions will be performed such as sending Newsgroup Update Info to SMTP module.

4.2.3 Data Dictionary

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Name:	News Update Info
Aliases:	None
Where used/how used:	Call Related Update News Web Service 10.2 (Output)

	Satisfied SMTP Client 8.1 (Input)

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

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

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

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

Description:
The electronic mail which is sent to the mail client.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Name:	Handle Update User Web Service Request

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

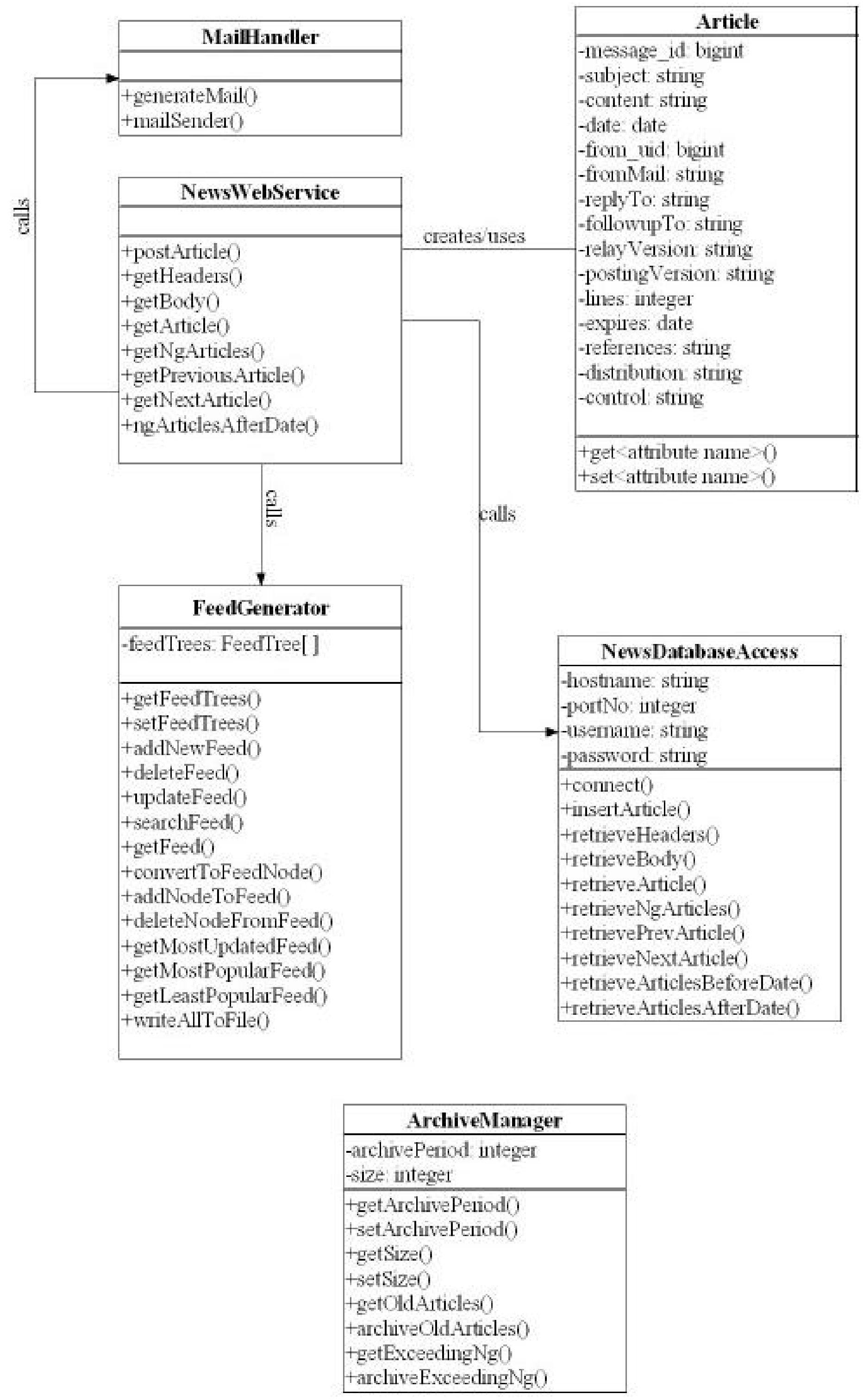
Description:
This data specifies all retrieves from Users table on the database. Any retrieval from Users table will flow through this data. User name retrieval by another user is an instance of such data.

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

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

5 CLASS DIAGRAMS

5.1 Article Management Module



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

Article Class

Attributes

Name	Type	Description
message_id	string	The unique message_id assigned to the article
subject	string	The subject of the article
content	string	The content of the article
date	date	Posted date of the article
from_uid	bigint	Userid of the user who post the article
fromMail	string	Mail address of the user who post the article
replyTo	string	The message_id of the replied article
followupTo	string	The message id of the article being followup to
relayVersion	string	The relayVersion of the article
postingVersion	string	The postingVersion of the article
lines	integer	Number of lines in the article
expires	date	The date which the article expires
references	string	The message_id of the article being referenced
distribution	string	The distribution of the article
control	string	The control of the article

Methods

Name	Return Type	Parameters	Description
getmessage_id	string	void	Returns the message_id
setmessage_id	void	string message_id	Sets message_id attribute
getsubject	string	void	Returns the subject
setsubject	void	string subject	Sets subject attribute
getcontent	string	void	Returns the content
setcontent	void	string content	Sets content attribute
getdate	date	void	Returns the date
setdate	void	date date	Sets date attribute
getfrom_uid	bigint	void	Returns the userid
setfrom_uid	void	int userid	Sets from_uid attribute
getfromMail	string	void	Returns the fromMail
setfromMail	void	string mail	Sets fromMail attribute
getreplyTo	string	void	Returns the replyTo
setreplyTo	void	string replyto	Sets replyTo attribute
getfollowupTo	string	void	Returns the followupTo
setfollowupTo	void	string followupto	Sets followupTo attribute
getrelayVersion	string	void	Returns the relayVersion
setrelayVersion	void	string relayVer	Sets relayVersion attribute
getpostingVersion	string	void	Returns the postingVersion
setpostingVersion	void	string postingVer	Sets postingVersion attribute
getlines	integer	void	Returns the number of lines
setlines	void	int lines	Sets lines attribute
getexpires	date	void	Returns the expire date
setexpires	void	date expires	Sets expires attribute
getreferences	string	void	Returns the references
setreferences	void	string references	Sets references attribute
getdistribution	string	void	Returns the distribution
setdistribution	void	string distribution	Sets distribution attribute
getcontrol	string	void	Returns the control
setcontrol	void	string control	Sets control attribute

NewsWebService Class

Methods

Name	Return Type	Parameters	Description
postArticle	void	void	Posts article
getHeaders	String[]	int ng_id	Retrieves headers
getBody	String	String message_id	Retrieves body of the article

getArticle	Article	String message_id	Retrieves article
getNgArticles	Article[]	int ng_id	Retrieves articles in newsgroup
getPreviousArticle	Article	void	Retrieves previous article
getNextArticle	Article	void	Retrieves next article
ngArticlesAfterDate	Article[]	date date	Retrieves articles posted after a given date

MailHandler Class

Methods

Name	Return Type	Parameters	Description
generateMail	String[]	Article article	Generates email from an article
mailSender	void	void	Sends email

FeedGenerator Class

Attributes

Name	Type	Description
feedTrees	FeedTree[]	Holds the feed trees of every newsgroup

Methods

Name	Return Type	Parameters	Description
getFeedTrees	FeedTree[]	void	Retrieves feed trees
setFeedTrees	void	Int[] ng_ids	Sets feed tree contents
addNewFeed	FeedTree[]	Int ng_id	Creates and adds new feed to existing feed tree array
deleteFeed	FeedTree[]	Int ng_id	Delete a newsgroup feed from feed tree array
updateFeed	FeedTree[]	Int ng_id	Update a newsgroup feed in feed tree array
searchFeed	FeedTree	Int ng_id	Search a feed in feed tree array
getFeed	FeedTree	Int ng_id	Retrieves a newsgroup feed
convertToFeedNode	FeedNode	Article article	Converts the article to feed node
addNodeToFeed	FeedTree	FeedNode fn	Appends node to feed tree
deleteNodeFromFeed	FeedTree	FeedNode fn	Deletes node from feed tree
serialize	void	FeedTree ft	Serializes the tree as xml document

NewsDatabaseAccess Class

Attributes

Name	Type	Description
hostname	string	Holds the hostname of the database
portNo	integer	Holds the portNo of the database
username	string	Holds the username of the database
password	string	Holds the password of the database

Methods

Name	Return Type	Parameters	Description
connect	Connection	String constr	Connects to database
insertArticle	Boolean	Article article	Inserts the posted article to the database
deleteArticle	Boolean	String mesage_id	Deletes the specified article
retrieveHeaders	String[]	Int ng_id	Retrieves headers of the newsgroup from database
retrieveBody	String	String mesage_id	Retrieves body of the article
retrieveArticle	Article	String mesage_id	Retrieves the specified article
retrieveNgArticles	Article[]	Int ng_id	Retrieves the articles of the newsgroup
retrievePrevArticle	Article	void	Retrieves previous article
retrieveNextArticle	Article	void	Retrieves next article
retrieveArticlesBeforeDate	Article[]	date date	Retrieves articles posted before a given date
retrieveArticlesAfterDate	Article[]	date date	Retrieves articles posted after a given date

ArchiveManager Class

Attributes

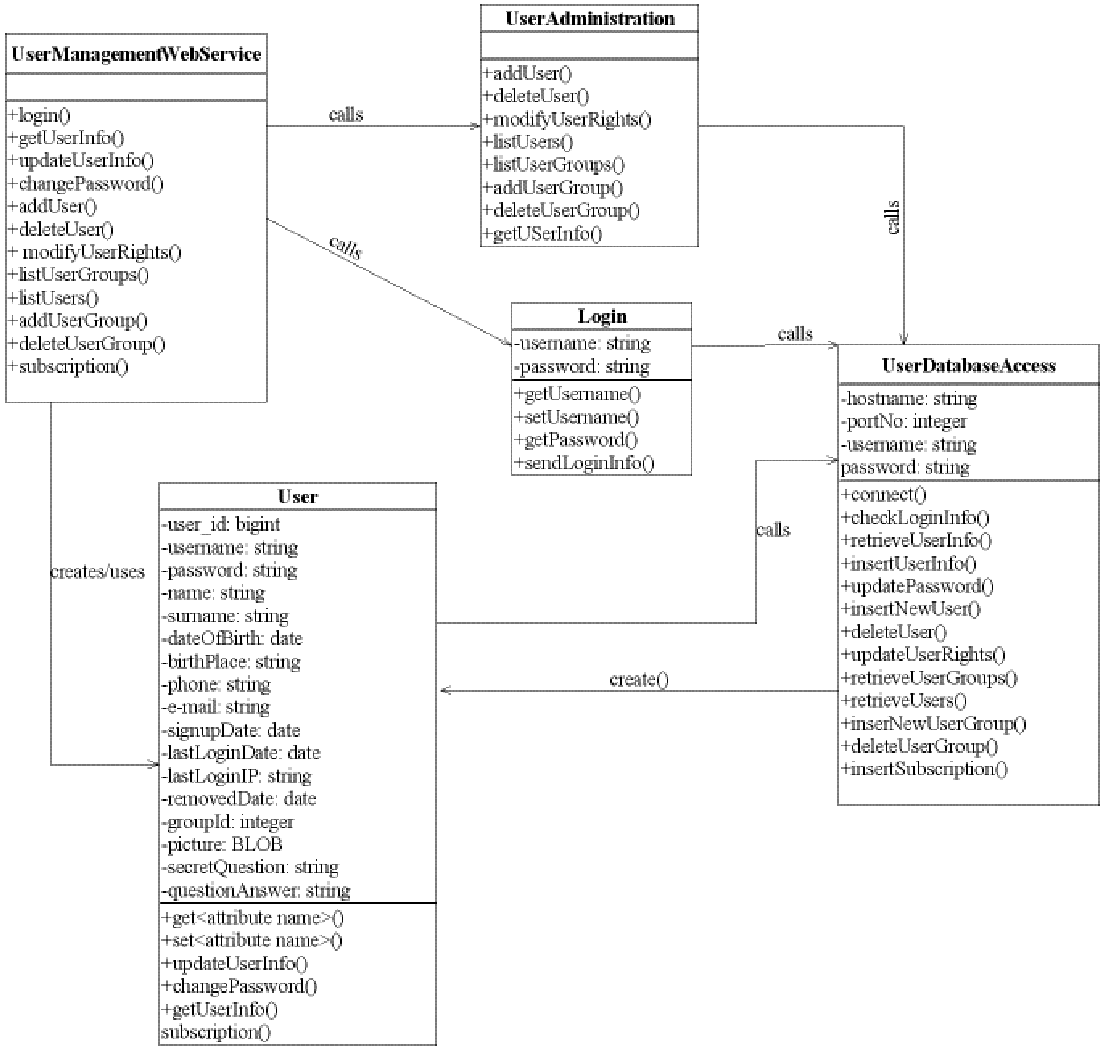
Name	Type	Description
archievePeriod	integer	Holds the archiving period of the articles
size	integer	Holds the archiving size of the article

Methods

Name	Return Type	Parameters	Description
getArchivePeriod	integer	void	Returns archive period
setArchivePeriod	void	Int period	Sets archivePeriod attribute
getSize	integer	void	Returns size

setSize	void	Int size	Sets size attribute
getOldArticles	Article[]	date date	Retrieves articles before a given date
archiveOldArticles	void	void	Inserts the old articles into archive database
getExceedingArticles	Article[]	Int size	Retrieves articles exceeding a given size
archiveExceedingArticles	void	void	Inserts exceeding articles into archive database

5.2 User Management Module



➢ **UserManagementWebService** class is a web service that maintains all methods required for user management. It calls UserAdministration, User and Login classes.

- **UserAdministration** class handles the administrative operations on users. When a user wants to add, delete, modify users and usergroups, the related methods are called and the modifications are reflected to the database. It calls UserDatabaseAccess class.
- **Login** class handles the login operation. It gets username and password and send login info to database in order to be checked. It calls UserDatabase Access class.
- **User** class handles the user related operations of the user management such as update user info, change login info, display user info etc. It calls UserDatabase Access class.
- **UserDatabaseAccess** class establishes connection with the database and creates queries in order to retrieve data from database or insert, delete and modify data into database. Its methods use these queries and do all the work related with users.

User Class

Attributes

Name	Type	Description
user_id	bigint	Holds user_id of the user
username	string	Holds username of the user
password	string	Holds password of the user
name	string	Holds name of the user
surname	string	Holds surname of the user
dateOfBirth	date	Holds dateOfBirth of the user
phone	string	Holds phone of the user
e-mail	string	Holds e-mail of the user
signupDate	date	Holds signupDate of the user
lastLoginDate	date	Holds lastLoginDate of the user
lastLoginIP	string	Holds lastLoginIP of the user
removedDate	date	Holds removedDate of the user
groupId	int	Holds groupId of the user
picture	BLOB	Holds picture of the user
secretQuestion	string	Holds secretQuestion of the user
questionAnswer	string	Holds questionAnswer of the user

Methods

Name	Return Type	Parameters	Description
getuser_id	int	void	Returns the user_id
setuser_id	void	int user_id	Sets user_id attribute
getusername	string	void	Returns the username
setusername	void	string username	Sets username attribute
getpassword	string	void	Returns the password

setpassword	void	string password	Sets password attribute
getdateOfBirth	date	void	Returns the dateOfBirth
setdateOfBirth	void	date date	Sets dateOfBirth attribute
getphone	string	void	Returns the phone
setphone	void	string phone	Sets phone attribute
gete-mail	string	void	Returns the e-mail
sete-mail	void	string e-mail	Sets e-mail attribute
getsignupDate	date	void	Returns the signupDate
setsignupDate	void	date date	Sets signupDate attribute
getlastLoginDate	date	void	Returns the lastLoginDate
setlastLoginDate	void	date date	Sets lastLoginDate attribute
getlastLoginIP	string	void	Returns the lastLoginIP
setlastLoginIP	void	string IP	Sets lastLoginIP attribute
getremovedDate	date	void	Returns the removedDate
setremovedDate	void	date date	Sets removedDate attribute
getgroupId	int	void	Returns the groupId
setgroupId	void	int gr_id	Sets groupId attribute
getpicture	Object	void	Returns the picture
setpicture	void	Object pic	Sets picture attribute
getsecretQuestion	string	void	Returns the secretQuestion
setsecretQuestion	void	string question	Sets secretQuestion attribute
getquestionAnswer	string	void	Returns the questionAnswer
setquestionAnswer	void	string answer	Sets questionAnswer attribute
getUserInfo	User	int user_id	Retrieves user info
updateUserInfo	void	int user_id	Updates user info
changePassword	void	String password	Changes password
subscription	void	void	Starts subscription process

UserManagementWebService Class

<u>Methods</u>

Name	Return Type	Parameters	Description
login	void	String username, String password	Starts the login process
getUserInfo	User	int user_id	Retrieves user info
updateUserInfo	void	int user_id	Updates user info
changePassword	void	String password	Changes password
addUser	void	User user	Adds new user
deleteUser	void	int user_id	Deletes an existing user
modifyUserRights	void	void	Updates the access rights of the usergroups
listUserGroups	String[]	void	Retrieves all usergroups

listUsers	User[]	void	Retrieves all users
addUserGroup	void	String groupname	Adds new user group
deleteUserGroup	void	String groupname	Deletes an existing user group
subscription	void	void	Starts subscription process

UserDatabaseAccess Class

Attributes

Name	Type	Description
hostname	string	Holds the hostname of the database
portNo	integer	Holds the portNo of the database
username	string	Holds the username of the database
password	string	Holds the password of the database

Methods

Name	Return Type	Parameters	Description
connect	Connection	String constr	Connects to database
insertNewUser	Boolean	User user	Inserts the user to the database
deleteUser	Boolean	int user_id	Deletes the specified user
retrieveUserInfo	User	Int user_id	Retrieves user info of the user from database
checkLoginInfo	Boolean	String username, String password	Validates the login data
retrieveUsers	User[]	void	Retrieves all users from database
retrieveUserGroups	String[]	void	Retrieves all usergroups from database
insertNewUserGroup	Boolean	String group	Inserts new user groups
deleteUserGroup	Boolean	String group	Deletes specified user group
updatePassword	void	String pwd	Changes the existing password
insertSubscription	Boolean	int user_id, int ng_id	Inserts the subscription info to the database
updateUserRights	void	String group	Updates the user rights of the specified user group

Login Class

Attributes

Name	Type	Description
username	string	Holds the username of the articles

password	string	Holds the password size of the article

Methods

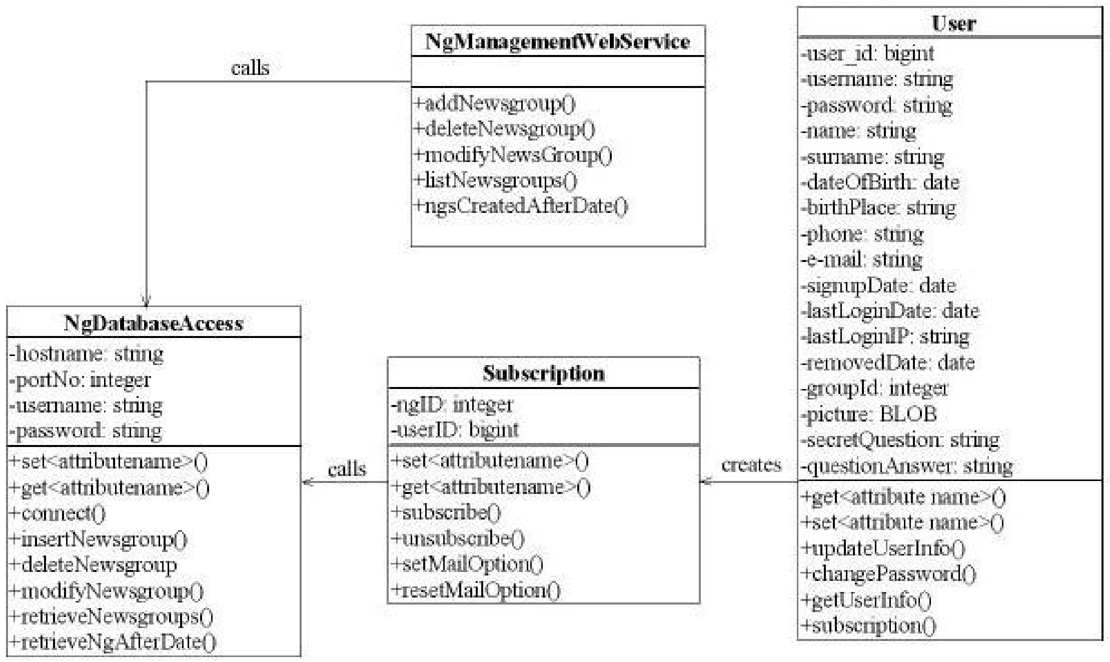
Name	Return Type	Parameters	Description
getUsername	string	void	Returns username
setUsername	void	String username	Sets username attribute
getPassword	string	void	Returns password
setPassword	void	String password	Sets password attribute
sendLoginInfo	void	String username, String password	Sends login data to the database in order to be controlled

UserAdministration Class

Methods

Name	Return Type	Parameters	Description
getUserInfo	User	int user_id	Retrieves user info
updateUserInfo	void	int user_id	Updates user info
changePassword	void	String password	Changes password
addUser	void	User user	Adds new user
deleteUser	void	int user_id	Deletes an existing user
modifyUserRights	void	void	Updates the access rights of the usergroups
listUserGroups	String[]	void	Retrieves all usergroups
listUsers	User[]	void	Retrieves all users
addUserGroup	void	String groupname	Adds new user group
deleteUserGroup	void	String groupname	Deletes an existing user group
subscription	void	void	Starts subscription process

5.3 Newsgroup Management Module



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

NgManagementWebService Class

Methods

Name	Return Type	Parameters	Description
addNewsgroup	Boolean	String gr_name	Adds new newsgroup with specified name
deleteNewsgroup	Boolean	String gr_name	Deletes the existing newsgroup with the specified name

modifyNewsgroup	void	String gr_name	Modifies the newsgroup with the specified name
listNewsgroups	String[]	void	Lists all newsgroups
ngsCreatedAfterDate	String[]	date date	Lists newsgroups created after the given date.

NgDatabaseAccess Class

Attributes

Name	Type	Description
hostname	string	Holds the hostname of the database
portNo	integer	Holds the portNo of the database
username	string	Holds the username of the database
password	string	Holds the password of the database

Methods

Name	Return Type	Parameters	Description
connect	Connection	String constr	Connects to database
insertNewsGroup	Boolean	String gr_name	Inserts newsgroup to the database
deleteNewsGroup	Boolean	int id	Deletes the specified newsgroup
retrieveNewsgroups	String[]	void	Retrieves newsgroups from database
modifyNewsgroup	void	int id	Retrieves body of the article
validateNG	Boolean	int id	Returns if the specified newsgroup is valid or not
retrieveNewsgroupAfterDate	String[]	date date	Retrieves articles posted after a given date

Subscription Class

Attributes

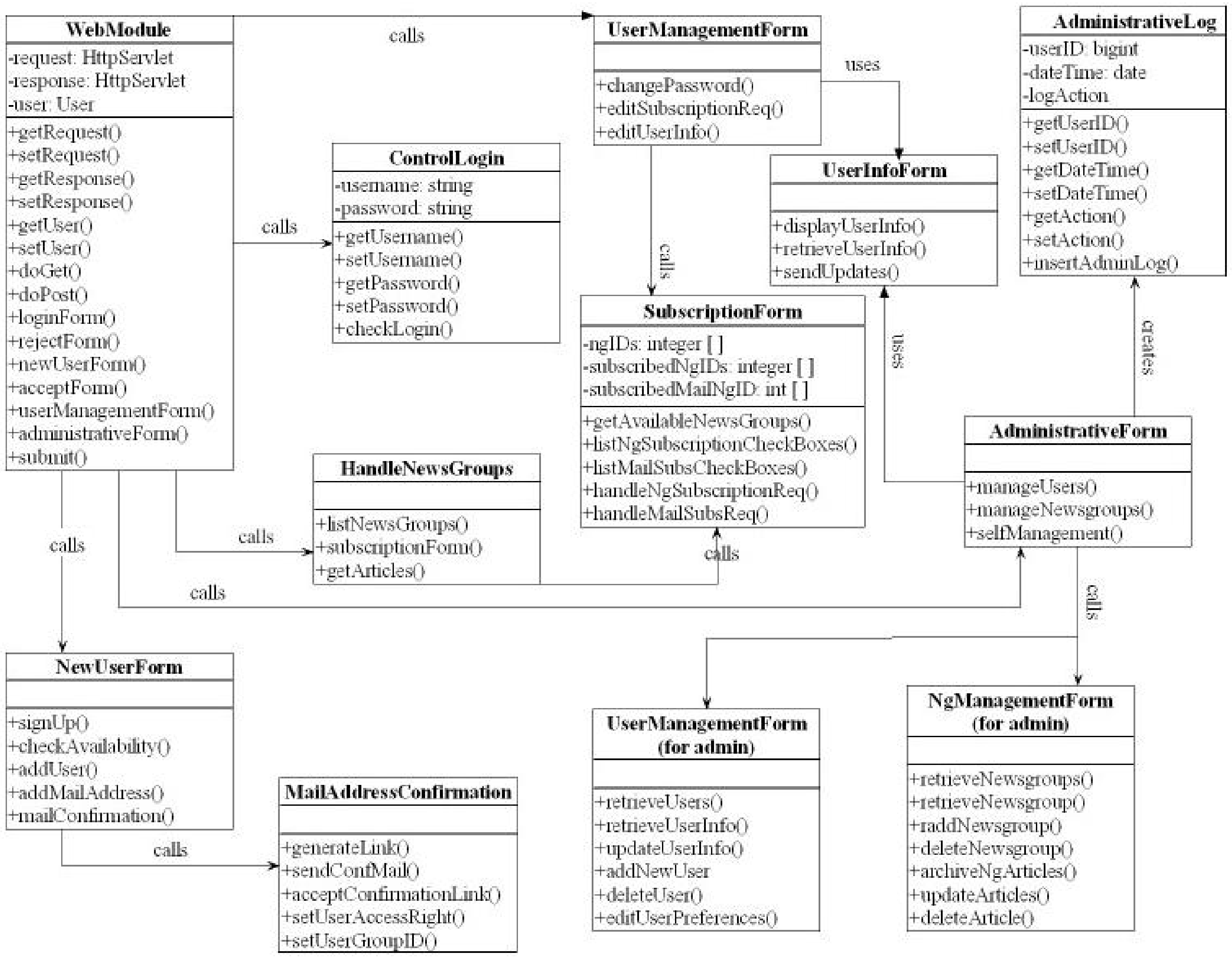
Name	Type	Description
ngID	integer	Holds the newsgroup id to be subscribed
userID	integer	Holds the userid of the user who requests to subscribe

Methods

Name	Return Type	Parameters	Description

getngID	integer	void	Returns ngID
setngID	void	Int ng_id	Sets ngID attribute
getuserID	integer	void	Returns userID
setuserID	void	Int user_id	Sets userID attribute
subscribe	void	Int user_id, Int ng_id	Sets subscription request for the specified user to the specified newsgroup
unsubscribe	void	Int user_id, Int ng_id	Releases the subscription request for the specified user to the specified newsgroup
setMailOption	void	Int user_id, Int ng_id	Sets receiving e-mail option for the specified user and from the specified newsgroup
resetMailOption	void	Int user_id, Int ng_id	Resets receiving e-mail option for the specified user and from the specified newsgroup

5.4 Web Module



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

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

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

WebModule Class

Attributes

Name	Type	Description
request	HttpServletRequest	Holds the request
response	HttpServletResponse	Holds the response
user	User	Holds the user info

Methods

Name	Return Type	Parameters	Description
getRequest	HttpServletRequest	void	Retrieves request
setRequest	void	HttpServletRequest req	Sets request attribute
getResponse	HttpServletResponse	void	Retrieves response
setResponse	void	HttpServletResponse	Sets response attribute
getUser	User	void	Retrieves user info
setUser	void	User user	Sets user attribute
doGet	void	HttpServletRequest req, HttpServletResponse res	Handles the Http Get requests
doPost	void	HttpServletRequest req, HttpServletResponse res	Handles the Http Post requests
loginForm	void	void	Opens login form for the web users for login operation
rejectForm	void	void	Rejects the form and sent information
newUserForm	void	void	Opens a signup form for the candidate users to signup

acceptForm	void	void	Accepts the information sent with the form
userManagement Form	void	void	Opens form for user operations such as update info etc
administrativeFor m	void	void	Opens form for administrative operations

NewUserForm Class

Methods

Name	Return Type	Parameters	Description
signUp	void	User user	Gets the sign up info and saves
checkAvailability	Boolean	User user	Controls if there is missing or invalid info
addUser	Boolean	User user	Adds the user in a different category until confirmation
addMailAddress	void	String addr	Saves mail address for confirmation
mailConfirmation	void	String addr	Sends confirmation mail to the user

UserManagementForm Class

Methods

Name	Return Type	Parameters	Description
changePassword	Boolean	String pwd	Gets new password from the user and changes the password
editSubsReq	void	void	Updates the subscription info for the user
editUserInfo	void	User user	Gets new user info and updates according to changes

HandleNewsGroups Class

Methods

Name	Return Type	Parameters	Description
listNewsGroups	String[]	void	Lists all newsgroups
subscriptionForm	void	void	Opens a subscription formfor the users in order to subscribe and set mailing options
getArticles	Article[]	int ng_id	Retrieves articles for the specified newsgroup

ControlLogin Class

Attributes

Name	Type	Description
username	string	Holds the username
password	string	Holds the password

Methods

Name	Return Type	Parameters	Description
getUsername	String	void	Returns username
setUsername	void	String uname	Sets username attribute
getPassword	String	void	Returns password
setPassword	void	String pwd	Sets password attribute
checkLogin	Boolean	void	Controls whether the username and password is a valid combination

MailConfirmation Class

Methods

Name	Return Type	Parameters	Description
generateLink	void	void	Generates confirmation link
sendConfMail	void	String mail	Sends confirmation mail to the user
acceptConfLink	void	void	Accepts confirmation
setUserAccessRight	void	void	Sets access rights of the newly added user
setUserGroupID	void	int id	Sets usergroup id for the user

AdminUserManagementForm Class

Methods

Name	Return Type	Parameters	Description
retrieveUsers	User[]	void	Retrieves all the users
retrieveUserInfo	User	int user_id	Retrieves the user info of a specified user
updateUserInfo	void	int user_id	Updates user info
addNewUser	Boolean	User user	Adds new user to the system after confirmation

deleteUser	Boolean	int user_id	Deletes user from the system
editUserPreferen ces	void	int user_id	Updates some user preferences

AdminNgManagementForm Class

Methods

Name	Return Type	Parameters	Description
retrieveNewsgroups	String[]	void	Retrieves all newsgroups
retrieveNewsgroup	String	int ng_id	Retrieves the newsgroup specified by the ng_id
addNewsgroup	Boolean	String name	Adds a new newsgroup as a result of admin request
deleteNewsgroup	Boolean	int ng_id	Deletes the newsgroup with id ng_id as a result of admin request
archiveNgArticles	Boolean	int ng_id, String Criteria	Archieve the articles in the specified newsgroup according to the criteria given
updateArticles	Boolean	int ng_id	Updates some properties of articles
deleteArticle	Boolean	String mes_id	Deletes the article whose message id is mes_id

AdministrativeLog Class

Attributes

Name	Type	Description
userID	integer	Holds the userID of the admin
datetime	dateTime	Holds the date and time of the configuration
logAction	integer	Holds the action type

Methods

Name	Return Type	Parameters	Description
getUserID	integer	void	Returns userID
setUserID	void	Int user_id	Sets UserID attribute
getdateTime	dateTime	void	Returns date and time of the action
setdateTime	void	dateTime dt	Sets dateTime attribute
getLogAction	integer	void	Retrieves log action type
setLogAction	void	int act	Sets logAction attribute
insertAdminLogs	Boolean	void	Adds Configuration logs as a result

			of changes

SubscriptionForm Class

Attributes

Name	Type	Description
ngIDs	integer[]	Holds the newsgroup ids
subscribedNgIDs	integer[]	Holds the ids of subscribed newsgroups for the user
subscribedMailNd IDs	integer[]	Holds the ids of subscribed and mail requested newsgroups for the user

Methods

Name	Return Type	Parameters	Description
getAvailableNGs	String[]	void	Retrieves newsgroups
listNGsubsCheckBoxes	void	void	Displays subscription options for the newsgroups
listMailSubsCheckBoxes	void	void	Displays mail receiving options
handleNGsubsReq	Boolean	subscription subs	Gets the subscription request and post the request to database access
handleMailSubsReq	Boolean	subscription subs	Gets the mail option set/reset request and post the request to database access

AdministrativeForm Class

Methods

Name	Return Type	Parameters	Description
manageUsers	void	void	Directs admin to the user mangement form as a result of user mangement request
manageNewsgroups	void	void	Directs admin to the newsgroup mangement form as a result of newsgroup mangement request
selfManagement	void	void	Directs admin to userInfo Form in order to change / update user info

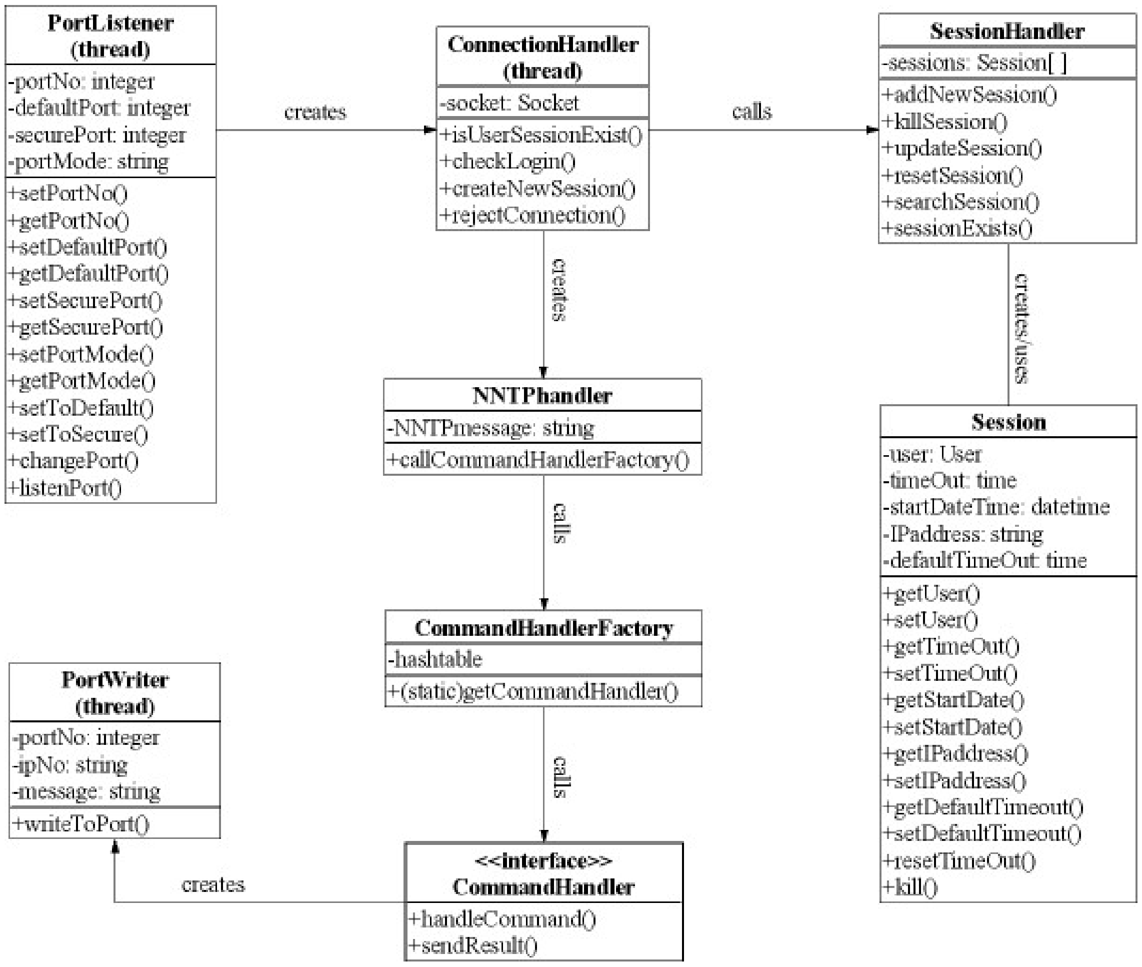
UserInfoForm Class

Methods

Name	Return Type	Parameters	Description
retrieveUserInfo	User	void	Retrieves the user info of the user

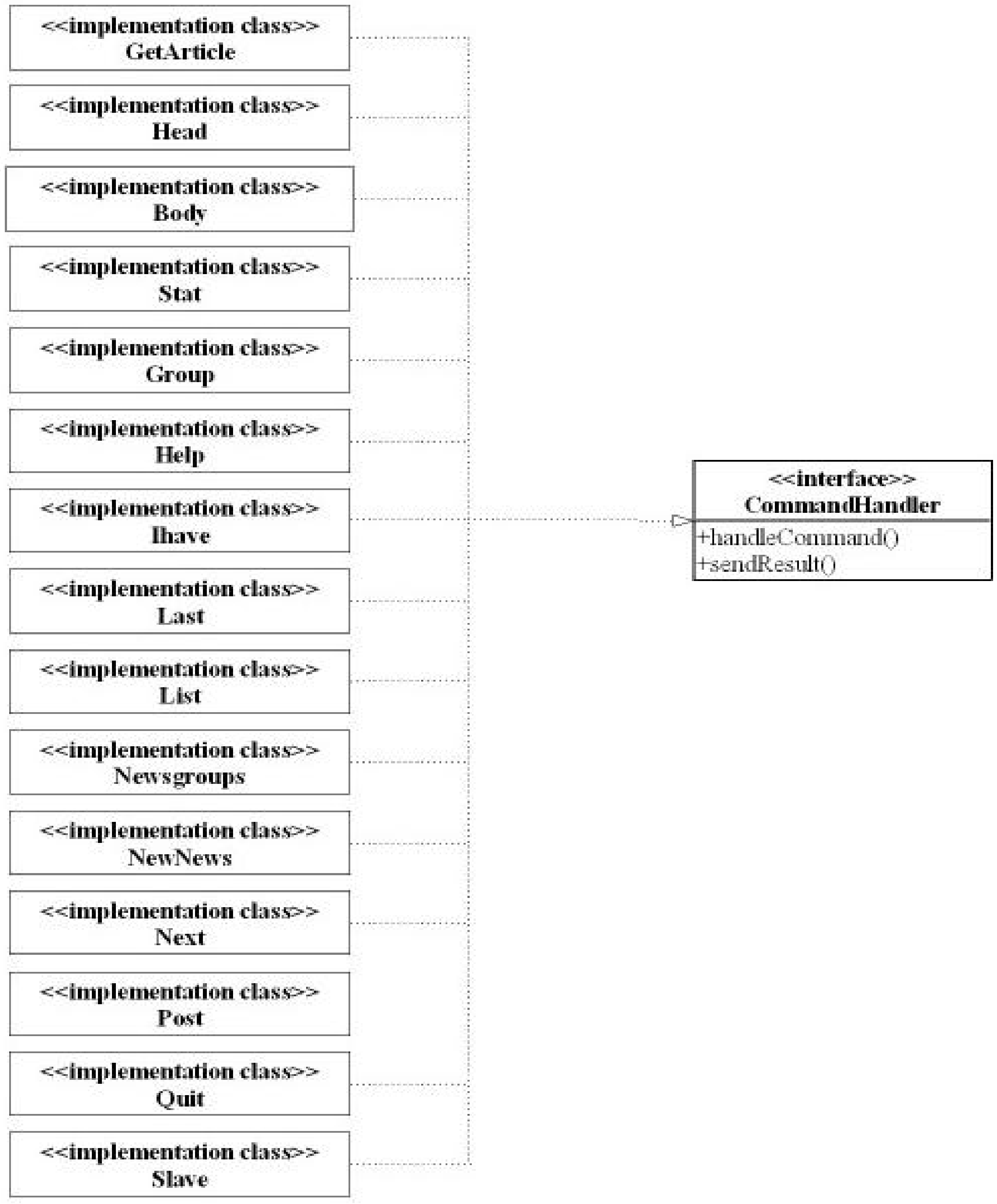
displayUserInfo	void	int user_id	Displays the user info retrieved fro database
sendUpdates	void	User user	As a result of change or update sends the updates to the database

5.5 NNTP Commands Module



- **PortListener** class is a thread and listens the specified port continuously. When a new message arrives, an instance of **ConnectionHandler** class is created. The information about sender of the message also arrives when message is sent. With this information, user’s session info is checked by calling SessionHandler class. If user’s session exists, it is updated. If not, a new session is created after username and password check.
- **SessionHandler** class calls session related methods. If a new session is created, **Session** class is instantiated and returned to SessionHandler and added to the sessions array.

➢ **NNTPhandler** class is created by ConnectionHandler. ConnectionHandler passes the message it received from the socket to NNTPhandler. NNTPhandler calls **CommandHandlerFactory** which has a hashtable including available NNTP commands. According to command, it calls related implementing class of interface **CommandHandler**. CommandHandler creates PortWriter after handling the command. **PortWriter** class receives result of the command and writes it to the port.


<<implementation class>> GetArticle
<<implementation class>> Head
<<implementation class>> Body
<<implementation class>> Stat
<<implementation class>> Group
<<implementation class>> Help
<<implementation class>> Ihave
<<implementation class>> Last
<<implementation class>> List
<<implementation class>> Newsgroups
<<implementation class>> NewNews
<<implementation class>> Next
<<implementation class>> Post
<<implementation class>> Quit
<<implementation class>> Slave
<<interface>> CommandHandler
+handleCommand()
+sendResult()

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

PortListener Class

Attributes

Name	Type	Description
portNo	integer	holds the portNo that the server listens
defaultPort	integer	holds the default port number
securePort	integer	holds the secure port number
portMode	string	holds the port mode

Methods

Name	Return Type	Parameters	Description
getportNo	integer	void	Returns port number
setportNo	void	int potno	Sets portNo attribute
getdefaultPort	integer	void	Returns default port number
setdefaultPort	void	int defport	Sets defaultPort attribute
getsecurePort	integer	void	Returns secure port number
setsecurePort	void	int secport	Sets securePort attribute
getPortMode	string	void	Returns port mode
setPortMode	void	string mode	Sets portMode attribute
setToDefault	void	void	Sets to portNo to defaultPort value
setToSecure	void	void	Sets to portNo to securePort value
changePort	void	int pno	Changes portNo to pno
listenPort	void	void	Listens the port specified with the portNo

Session Class

Attributes

Name	Type	Description
user	User	Holds the user
timeout	Time	Holds the timeout for the session
startDateTime	dateTime	Holds the start date and time of the session
IPAddress	string	Holds the IPAddress
defaultTimeOut	Time	Holds the default time out for the session

Methods

Name	Return Type	Parameters	Description
getUser	User	void	Returns the user that the session is created for
setUser	void	User user	Sets user attribute
getTimeOut	Time	void	Returns the timeout for he session
setTimeOut	void	Time t	Sets timeout attribute
getStartDateTime	DateTime	void	Returns the start date and time of the session
setStartDateTime	void	DateTime dt	Sets the startDateTime attribute
getIPAddress	String	void	Returns the IPAddress that the user connects from
setIPAddress	void	String IP	Sets IPAddress attribute
getDefaultTimeOut	Time	void	Returns default time out fort he session
setDefaultTimeOut	void	Time t	Sets defaultTimeOut attribute
resetTimeOut	void	void	Resets the timeout value of the session
kill	void	void	Kills the session

ConnectionHandler Class

Attributes

Name	Type	Description
socket	Socket	Holds the socket which handles the connection

Methods

Name	Return Type	Parameters	Description
isUserSessionExist	Boolean	int user_id	Returns if the user has a session, has authenticated
checkLogin	Boolean	String uname, String pwd	If the session does not exist, controls whether the login data is valid
createNewSession	Session	User user, String IP	Creates a new session for a valid user
rejectConnection	void	String IP	If the login data is invalid or there is any other problem, it rejects the connection

SessionHandler Class

Attributes

Name	Type	Description
sessions	Session[]	Holds the sessions of the authenticated users

Methods

Name	Return Type	Parameters	Description
addNewSession	void	Session ses	Adds a new session object to the current session array
killSession	void	int id	Kills the session
updateSession	void	int id	Updates the session
resetSession	void	int id	Resets the session
searchSession	Session	int id	Searches for a specific session
sessionExists	boolean	int id	Returns whether the session exists or not

PortWriter Class

Attributes

Name	Type	Description
portNo	integer	Holds the port number
IPno	string	Holds IP
message	string	Holds the message written

Methods

Name	Return Type	Parameters	Description
writeToPort	Boolean	String mes	Writes the message to the socket created on the portNo

NNTPHandler Class

Attributes

Name	Type	Description
NNTPMessage	String	Holds the NNTP command sent by an NNTP client

Methods

Name	Return Type	Parameters	Description
callCommandHandlerFactory	void	String Command	Creates a commandHandlerFactory object in order to map the command

CommandHandlerFactory Class

Attributes

Name	Type	Description
hashtable	HashTable	Holds the hash table of the NNTP commands

Methods

Name	Return Type	Parameters	Description
getCommandHandler	Object	String Command	Maps the command with the right commandHandler and returns the CommandHandler fort he command

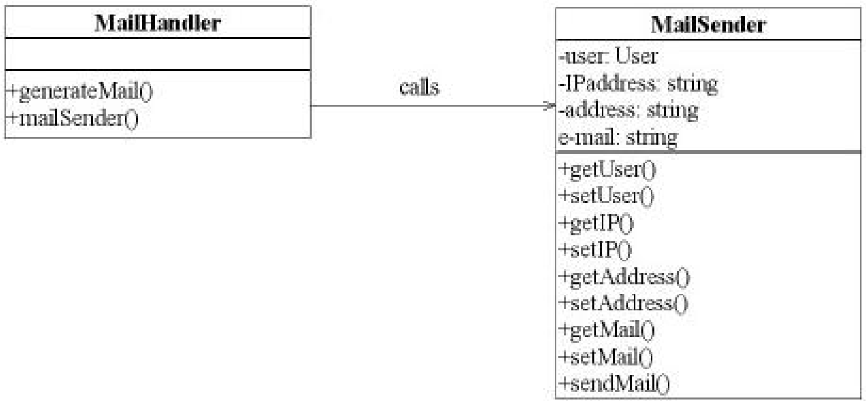
CommandHandler Class

Methods

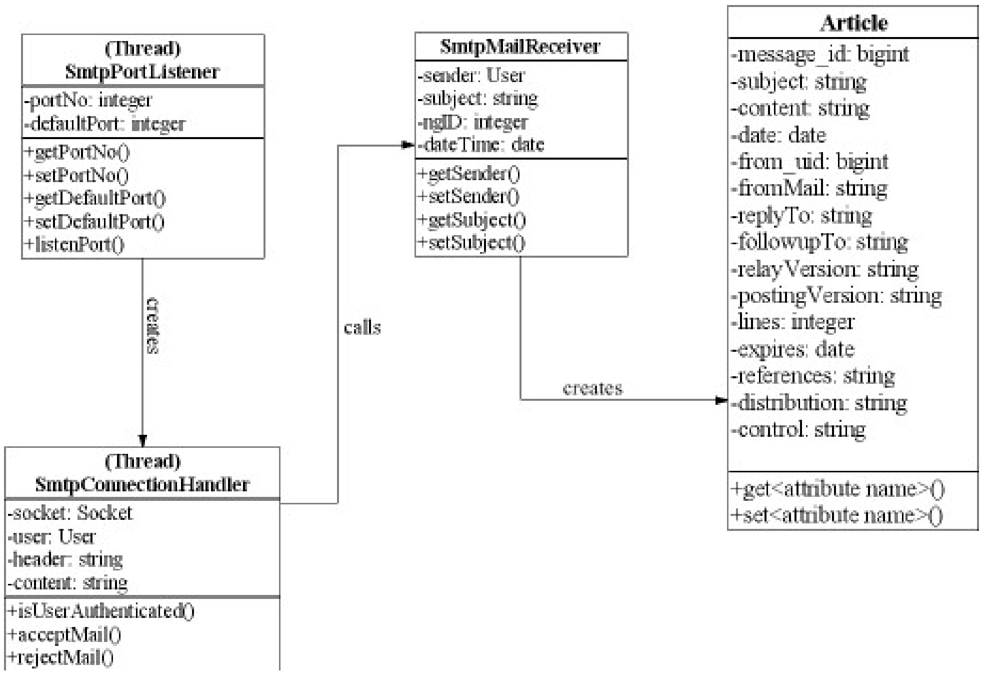
Name	Return Type	Parameters	Description
sendResult	Object	void	Returns the result of the command
handleCommand	void	void	Handles the mapped command. Since this class is interface class, this function will be implemented in child classes.

5.6 Mailing Module

Sending Mail



Receiving Mail



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

For the First part:

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

- **MailSender** class maintains the information about the user and sends e-mail to the user via smtp.

For the Second Part:

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

Mail Sender Class

Attributes

Name	Type	Description
user	User	Holds the sender of the article
IPaddress	string	Holds the IPaddress
address	string	Holds the mail address of the user to be sent
Mail	string	Holds the mail content

Methods

Name	Return Type	Parameters	Description
getUser	User	void	Returns the sender of the article
setUser	void	User user	Sets user attribute
getIPaddress	string	void	Returns the IPaddress
setIPaddress	void	string IP	Sets IPaddress attribute
getAddress	string	void	Returns the mailaddress
setAddress	void	string address	Sets mailaddress attribute
getMail	string	void	Returns generated Mail
setMail	void	String Mail	Sets Mail attribute
sendMail	void	void	Sends e-mail to the users

SMTPPortListener Class

Attributes

Name	Type	Description
portNo	int	Holds the port number that the server listens
defaultPort	int	Holds the default port number

Methods

Name	Return Type	Parameters	Description
getPortNo	int	void	Returns the port number that the server listens
setPortNo	void	int portno	Sets the portNo attribute
getdefaultPort	int	void	Returns the default port number
setdefaultPort	void	int defPortNo	Sets defaultPort attribute
listenPort	void	void	Listens to the port in order to serve the client requests

SMTPMailReceiver Class

Attributes

Name	Type	Description
sender	User	Holds the sender of the e-mail received
subject	string	Holds the subject of the e-mail
content	string	Holds the content of the e-mail
ngID	int	Holds the newsgroup id of the newsgroup that the mail is sent to
date	dateTime	Holds the sent date and time of the mail

Methods

Name	Return Type	Parameters	Description
getSender	User	void	Returns the sender of the e-mail
setSender	void	User sender	Sets the sender attribute
getSubject	string	void	Returns the subject of the e-mail
setSubject	void	String subject	Sets subject attribute
getNgID	int	void	Returns ngID of the newsgroup that the mail is sent to
setNgID	void	int id	Sets the ngID attribute
getDateTime	dateTime	void	Returns the date and time that the mail is sent
setDateTime	void	dateTime date	Sets dateTime attribute

SMTPConnectionHandler Class

Attributes

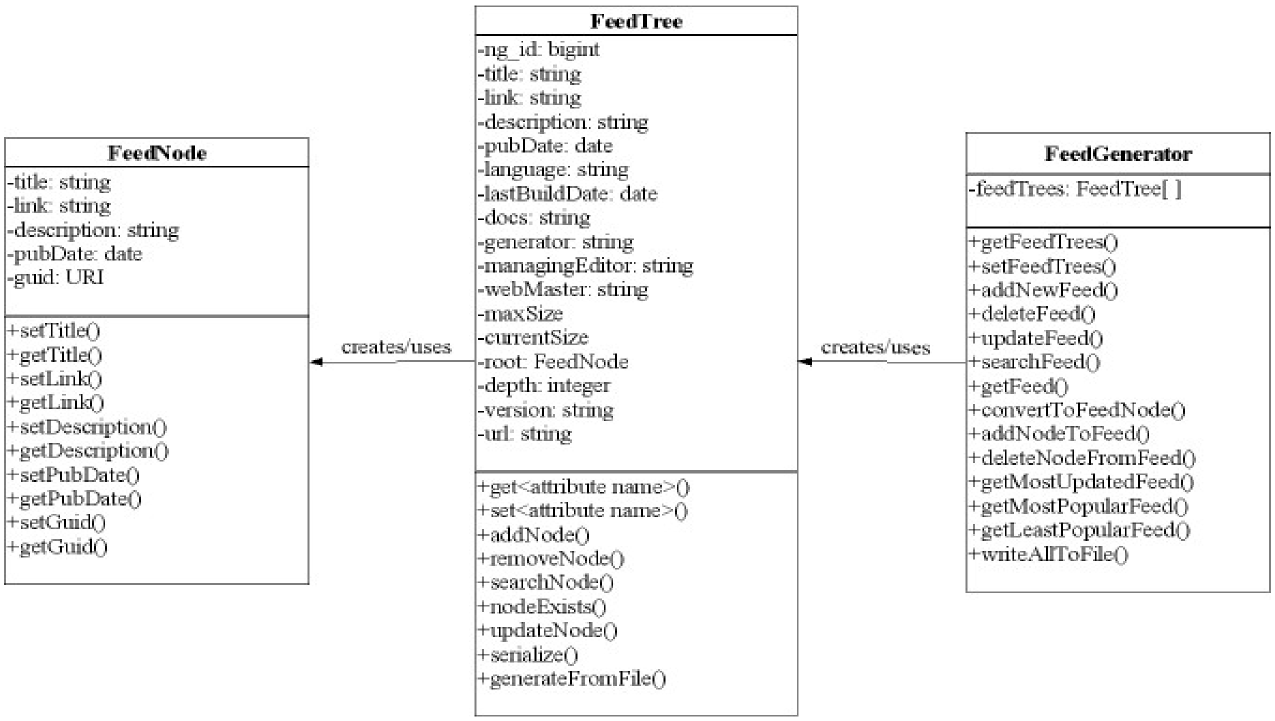
Name	Type	Description

user	User	Holds the user who connects an sends mail
socket	Socket	Holds the socket that the connection is established through

Methods

Name	Return Type	Parameters	Description
isUserAuthenticated	Boolean	void	Returns if the sender is a valid user or not
acceptMail	void	String mail	Accepts the e-mail in order to add as an article and generates an article
rejectMail	void	void	Rejects e-mail and does not generate an article

5.7 RSS Module



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

- **FeedGenerator** class is called when a new article is posted and inserted into the database. As shown in article management module, article management web service calls this class. FeedGenerator class has an array of feed trees which are instances of

FeedTree class. Each newsgroup has its own feed tree, since a user may subscribe to any of them individually. For example, if the web service for inserting an article is invoked, it generates a request to the FeedGenerator after confirming the insertion of the article to the database. This request is to add a new entry for the specified newsgroup tree associated with the newly added article. FeedTree finds the corresponding feed tree and calls the method to add the article to the tree. Update and delete operations follows the same steps as in adding a new article.

- **FeedTree** class is a tree of feed nodes. It is a logical representation of the xml document. The listed methods above maintain the tree. Each tree has a maximum size. When the tree exceeds this size, the oldest entry of the tree is deleted to maintain the size. After each change operation to the tree, it serializes the tree to the file path specified by "url" attribute of the class. Now, any feed aggregators realize the changes when it checks out the feed for new news.
- **FeedNode** is a logical representation of the xml of a single article. It is appended to the related feed tree when a new post is inserted into database.

FeedNode Class

Attributes

Name	Type	Description
title	string	The title of the article(node)
link	string	The link of the article(node)
description	string	The description of the article
pubDate	date	Publish date of the article
guid	URI	URI of the article

Methods

Name	Return Type	Parameters	Description
getTitle	string	void	Returns the title
setTitle	void	string title	Sets title attribute
getLink	string	void	Returns the link
setLink	void	string link	Sets link attribute
getDescription	string	void	Returns the description
setDescription	void	string description	Sets description attribute
getPubDate	date	void	Returns the pubDate
setPubDate	void	date date	Sets pubDate attribute
getGuid	URI	void	Returns the guid

setGuid	void	URI uri	Sets guid attribute

FeedTree Class

Attributes

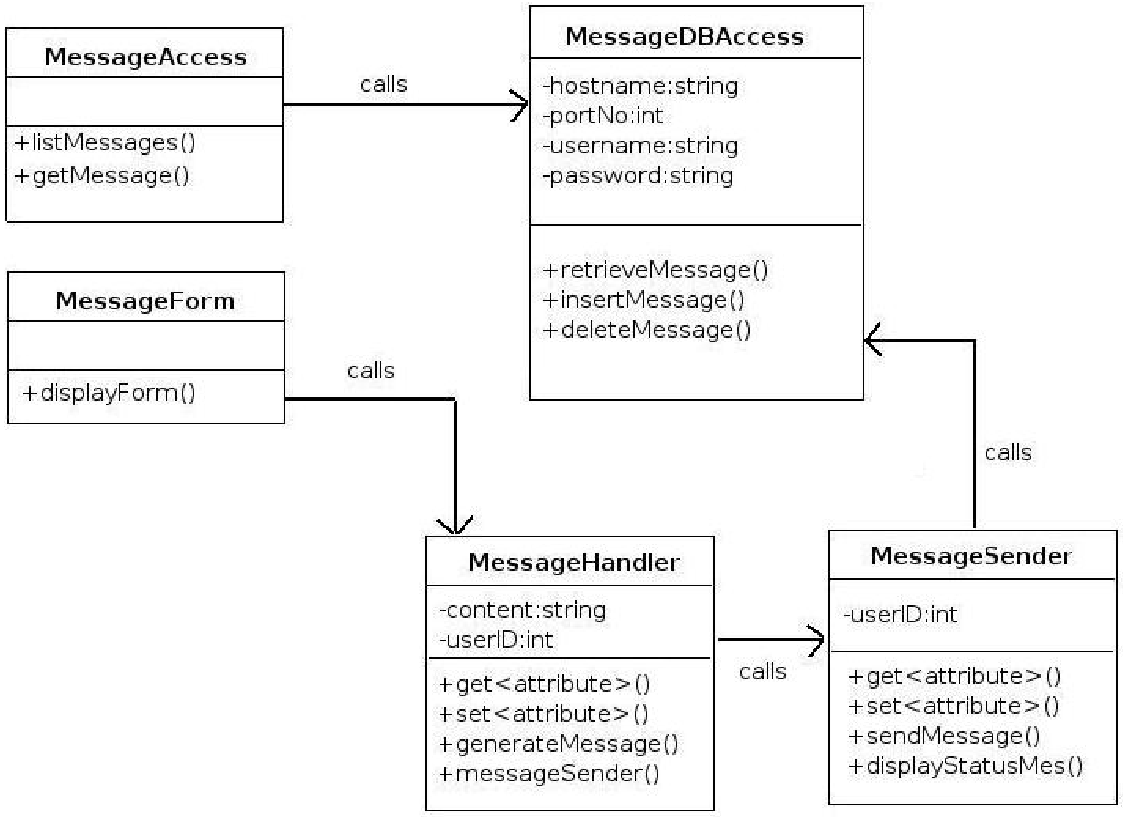
Name	Type	Description
ng_id	bigint	The newsgroup id
title	string	The title of the tree
link	string	The link of the tree
description	string	Description of the tree
pubDate	date	PublishDate of the tree
language	string	
lastBuildDate	date	
docs	string	
generator	string	
managingEditor	string	
webMaster	string	
maxSize	int	
currentSize	int	
root	FeedNode	
depth	int	
version	string	
url	string	

Methods

Name	Return Type	Parameters	Description
get<attributename>	Attribute type	void	Returns the attributes
set<attributename>	void	Type var	Sets attributes
addNode	FeedTree	FeedNode nd, FeedTree tr	Adds feed node to feed tree
removeNode	FeedTree	FeedNode nd, FeedTree tr	Removes feed node from tree
searchNode	FeedNode	FeedNode nd, FeedTree tr	Searches feed node in the tree
nodeExists	Boolean	FeedNode nd, FeedTree tr	Returns if the node exists or not
updateNode	void	FeedNode nd, FeedTree tr	Updates feed node in the tree
serialize	void	void	Serializes the tree as xml

			document
generateFromFile	void	string filename	Generates tree from xml document

5.8 Messaging Module



MessageSender Class

Attributes

Name	Type	Description
userID	integer	Holds the id of the user that the message to be sent.

Methods

Name	Return Type	Parameters	Description
getUserID	int	void	Returns the id of the user that the message is to be sent
setUserID	void	int id	Sets userID attribute
sendMessage	Boolean	String Message	Sends message to the user
displayStatusMes	void	Boolean status	Sends a status message to the sender

			of the message denoting the success or failure of sending operation.

MessageDBAccess Class

Attributes

Name	Type	Description
hostname	string	Holds the hostname of the database
portNo	integer	Holds the portNo of the database
username	string	Holds the username of the database
password	string	Holds the password of the database

Methods

Name	Return Type	Parameters	Description
retrieveMessage	string	int id	Retrieves body of the specified message.
insertMessage	Boolean	string content	Inserts a sent message to the database.
deleteMessage	Boolean	int id	Deletes a specified message from database.

MessageHandler Class

Attributes

Name	Type	Description
userID	int	Holds the id of the receiver.
content	string	Holds the content of the message

Methods

Name	Return Type	Parameters	Description
getUserID	int	void	Returns the id of the receiver
setUserID	void	int id	Sets the userID attribute
getContent	string	void	Returns the content of the message
setContent	void	String content	Sets content attribute
generateMessage	string	void	Forms a message from the content.
messageSender	void	void	Sends the message

MessageAccess Class

Methods

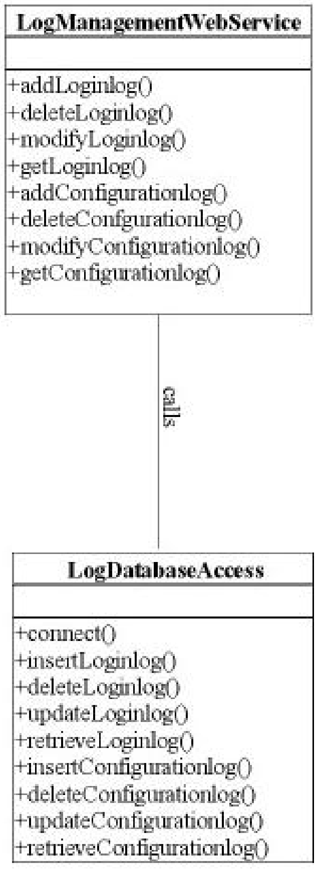
Name	Return Type	Parameters	Description
listMessages	string[]	int id	Returns the messages sent to the user
getMessage	string	int mes_id	Retrieves the specified message

MessageForm Class

Methods

Name	Return Type	Parameters	Description
displayForm	void	void	Displays the message form according to a user request.

5.9 Log Module



- LogManagementWebService class is a web service that maintains required methods for login log and configuration log operations. This class calls LogDatabaseAccess class to reflect the modifications into the database.

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

LogManagementWebService Class

Methods

Name	Return Type	Parameters	Description
addLoginlog	Boolean	int user_id, date date, string IP	Adds login logs after every login operation
deleteLoginlog	Boolean	int log_id	Deletes login log when the admin requests
modifyLoginlog	Boolean	int log_id	Modifies login log when the admin requests
getLoginlog	String[]	void	Retrieves all login logs
addConfigurationlog	Boolean	int user_id, date date, string IP, int type	Adds configuration logs after every system operation
deleteConfigurationlog	Boolean	int log_id	Deletes configuration log when the admin requests
modifyConfigurationlog	Boolean	int log_id	Modifies configuration log when the admin requests
getConfigurationlog	String[]	void	Retrieves all configuration logs

LogDatabaseAccess Class

Attributes

Name	Type	Description
hostname	string	Holds the hostname of the database
portNo	integer	Holds the portNo of the database
username	string	Holds the username of the database
password	string	Holds the password of the database

Methods

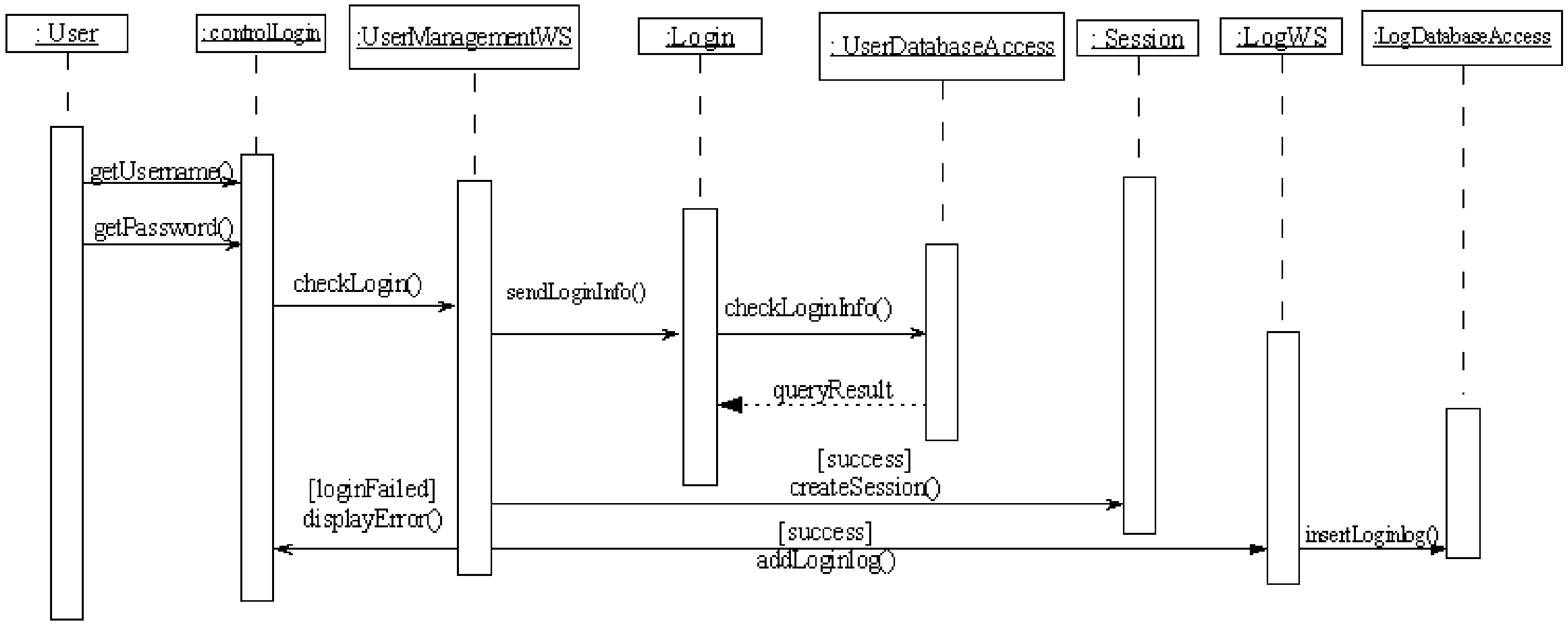
Name	Return Type	Parameters	Description
insertLoginlog	Boolean	String query	Inserts login logs after every login operation into the login log table in the database

deleteLoginlog	Boolean	String query	Deletes login log when the admin requests from the database
updateLoginlog	Boolean	String query	Updates login log when the admin requests
retrieveLoginlog	String[]	String query	Retrieves all login logs from database
insertConfigurationlog	Boolean	String query	Adds configuration logs after every system operation into the configuration log table
deleteConfigurationlog	Boolean	String query	Deletes configuration log when the admin requests from the database
updateConfigurationlog	Boolean	String query	Updates configuration log when the admin requests in the database
retrieveConfigurationlog	String[]	String query	Retrieves all configuration logs in the database

6 SEQUENCE DIAGRAMS & SEQUENCE OF EVENTS

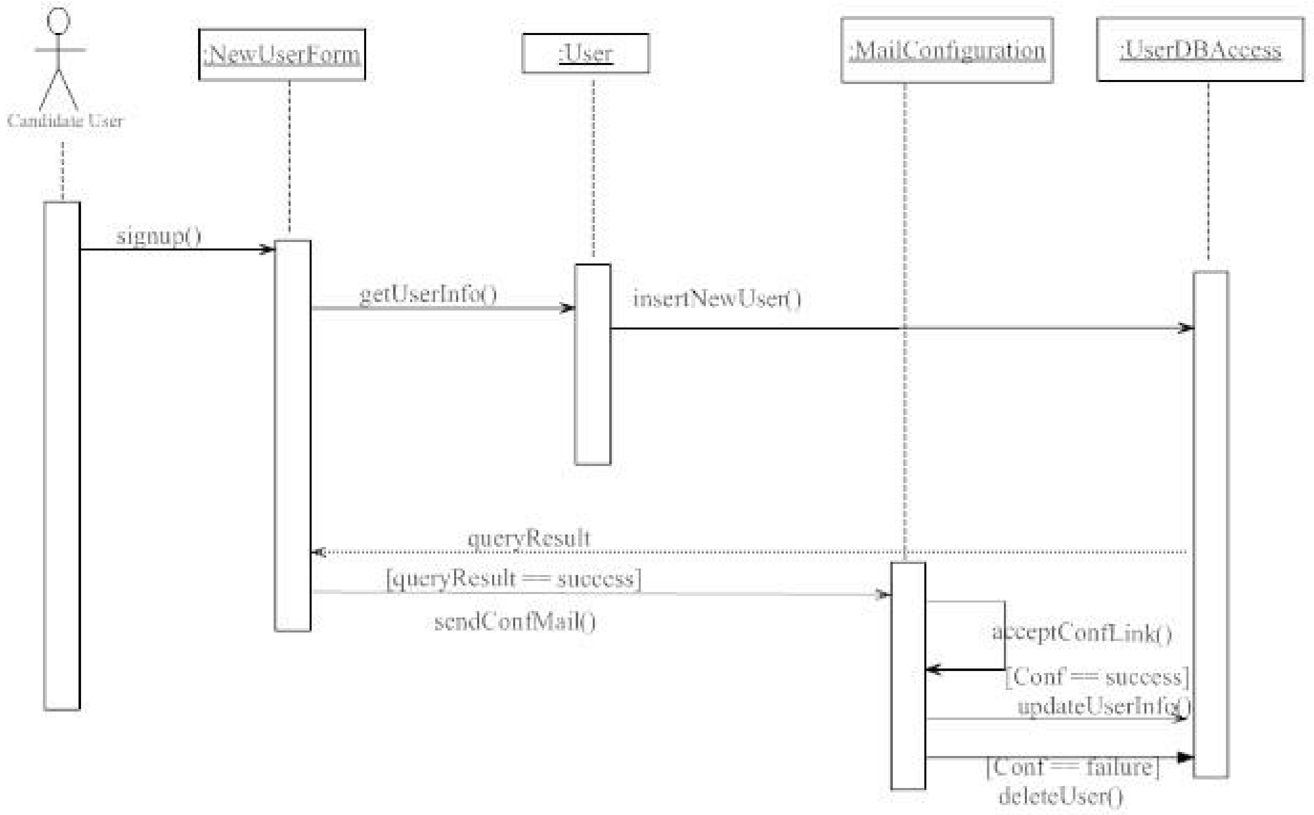
6.1 Sequence Diagrams

6.1.1 Login and Authentication

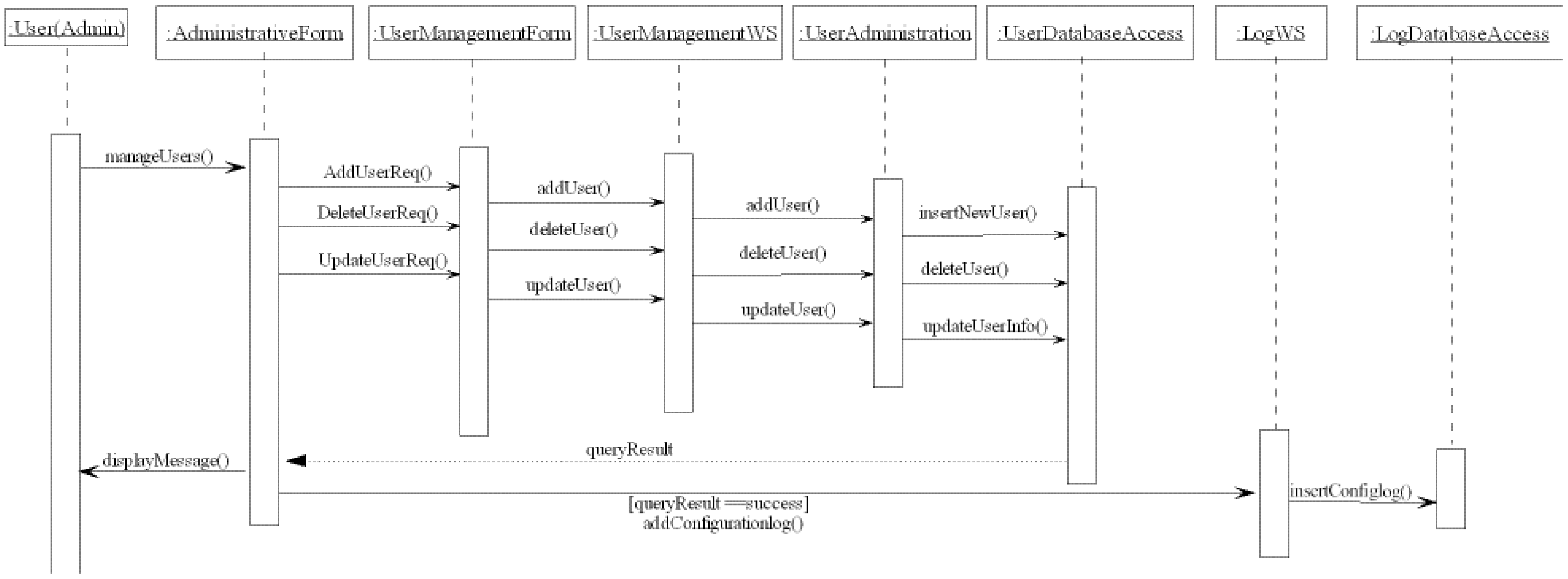


: User
:controlLogin
:UserManagementWS
:Login
: UserDatabaseAccess
: Session
:LogWS
:LogDatabaseAccess
getUsername()
getPassword()
checkLogin()
sendLoginInfo()
checkLoginInfo()
queryResult
[success]
createSession()
[loginFailed]
displayError()
[success]
addLoginlog()
insertLoginlog()

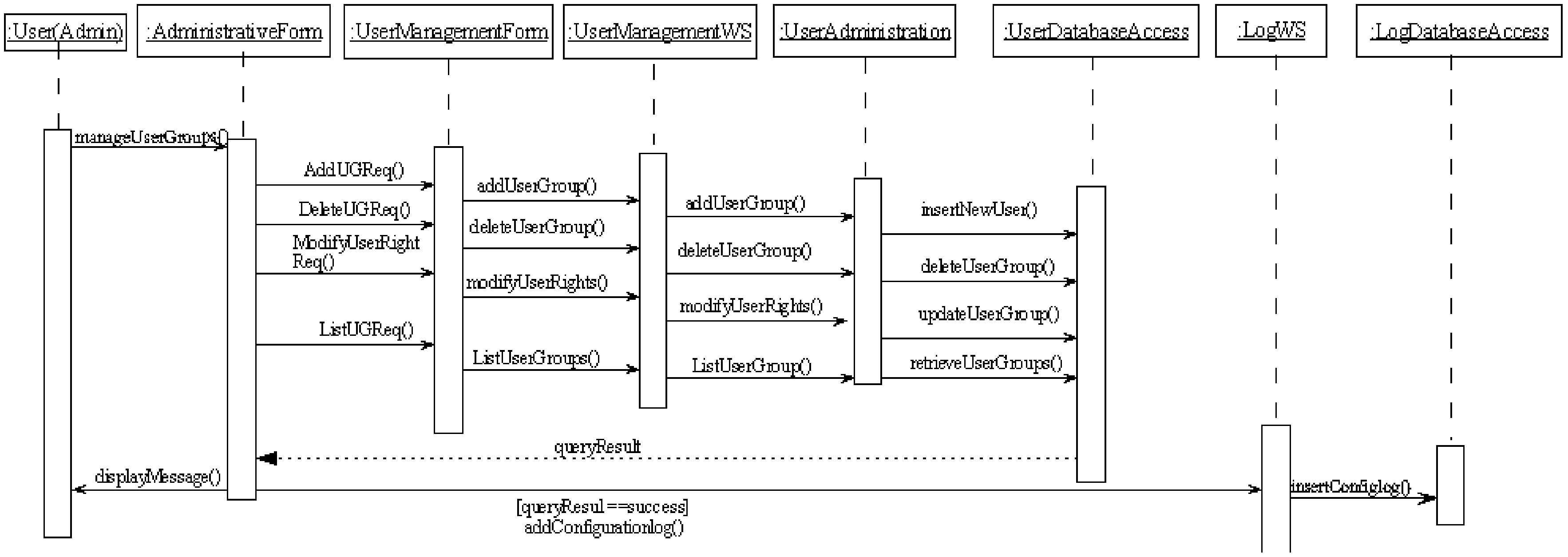
6.1.2 Signup


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

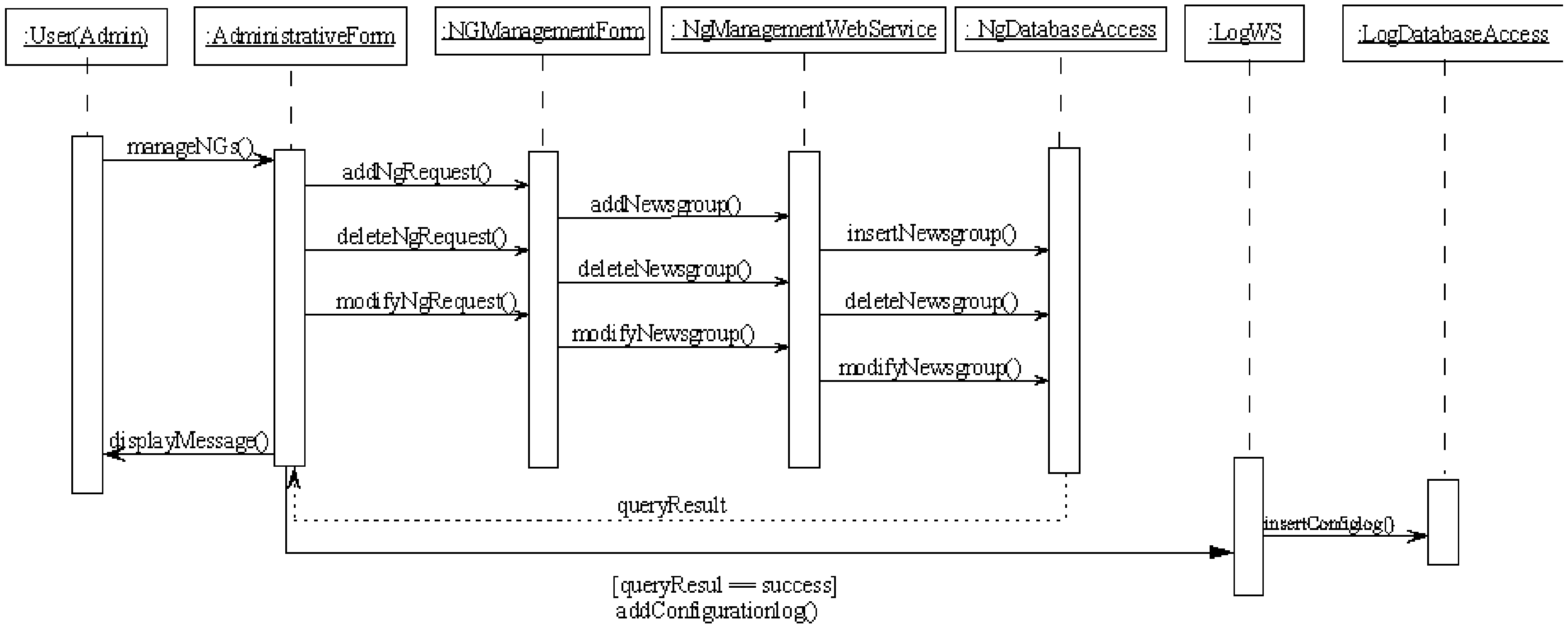
6.1.3 User Management



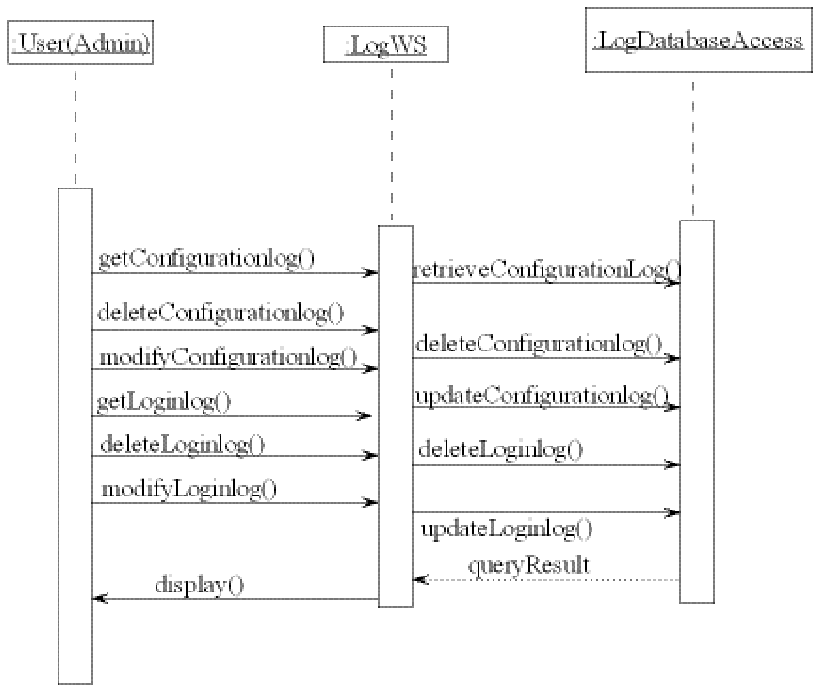
6.1.4 Usergroup Management



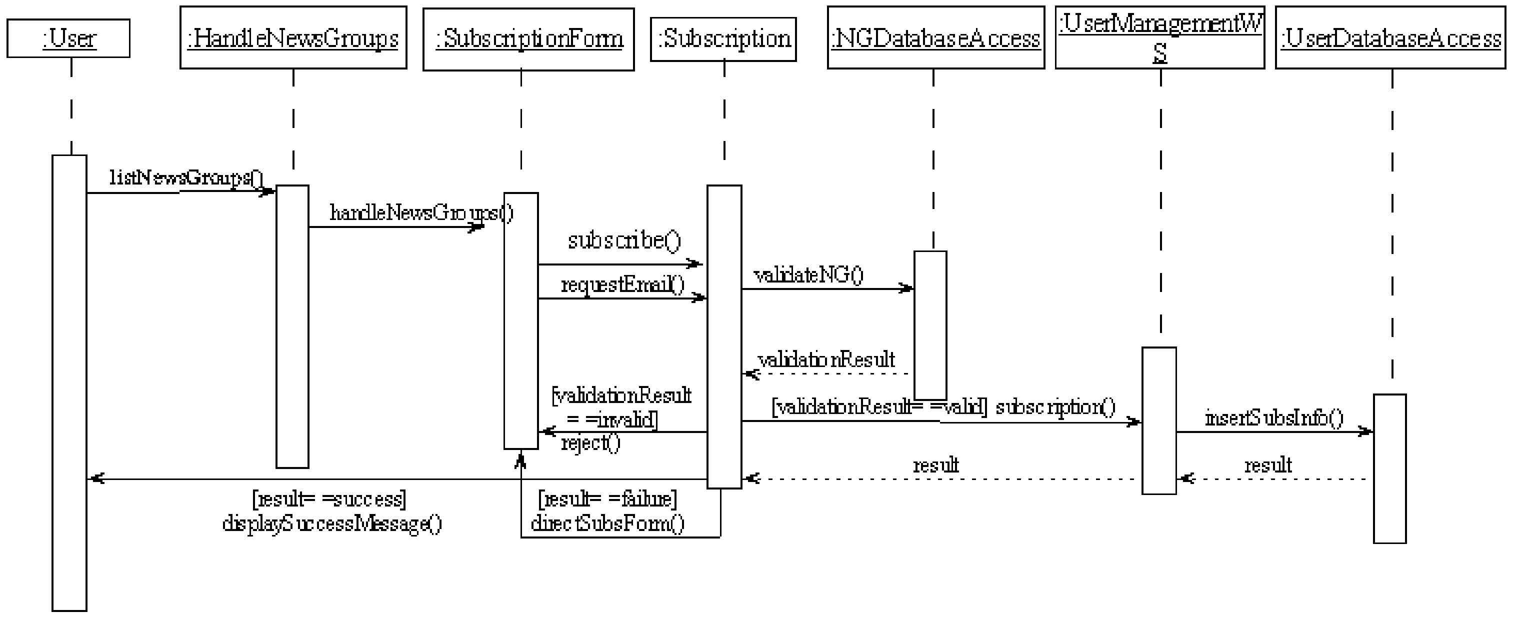
6.1.5 Newsgroup Management



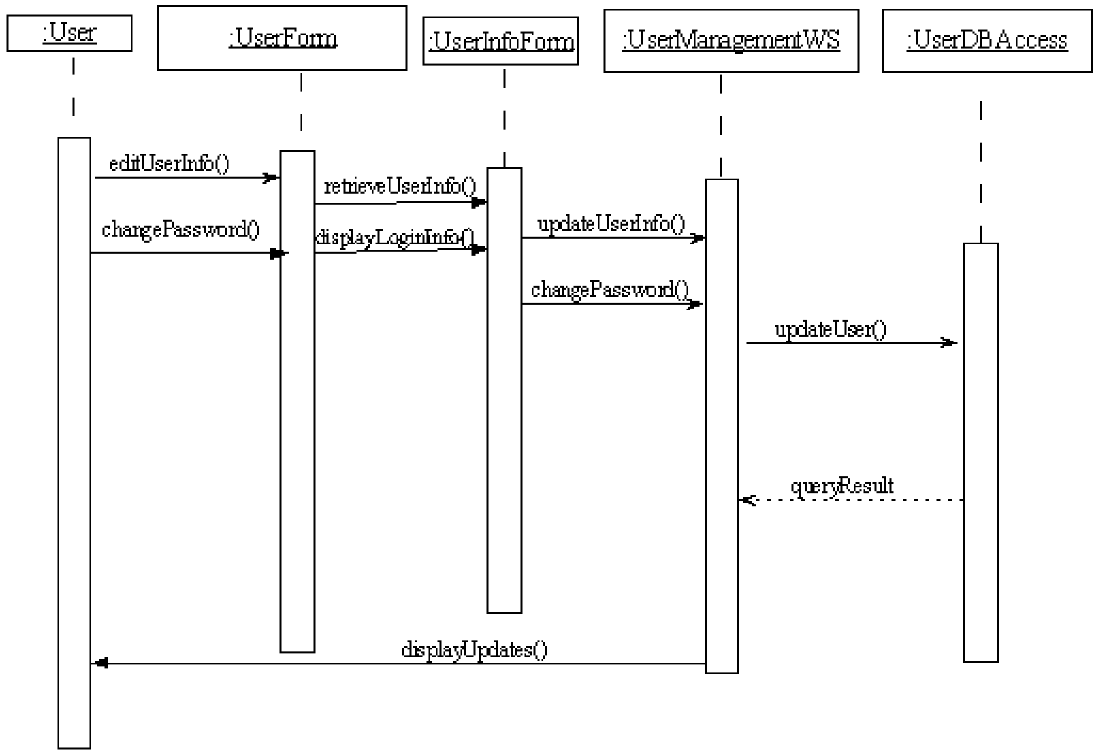
6.1.6 Admin Log Control



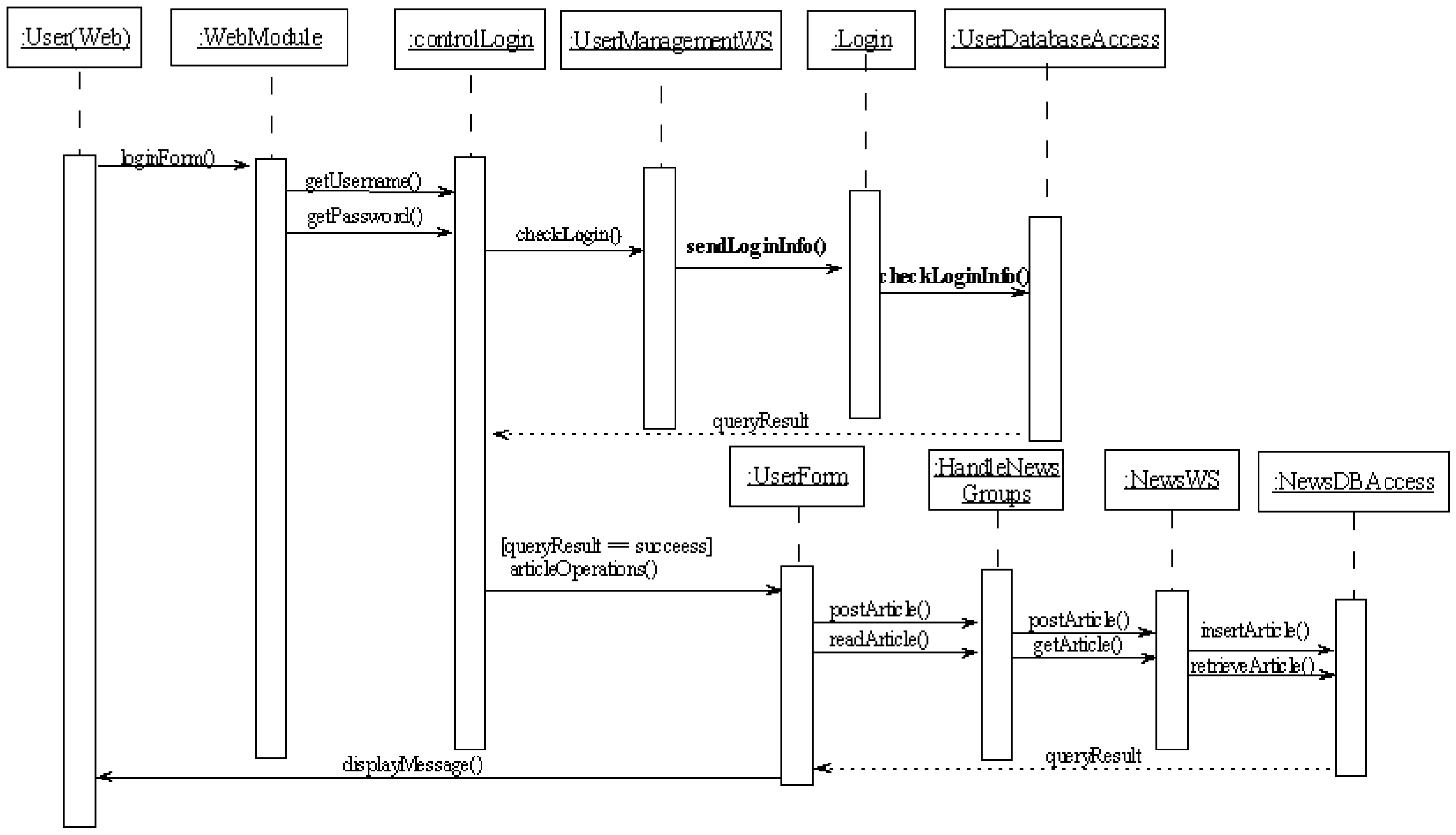
6.1.7 Subscription



6.1.8 Update User Info

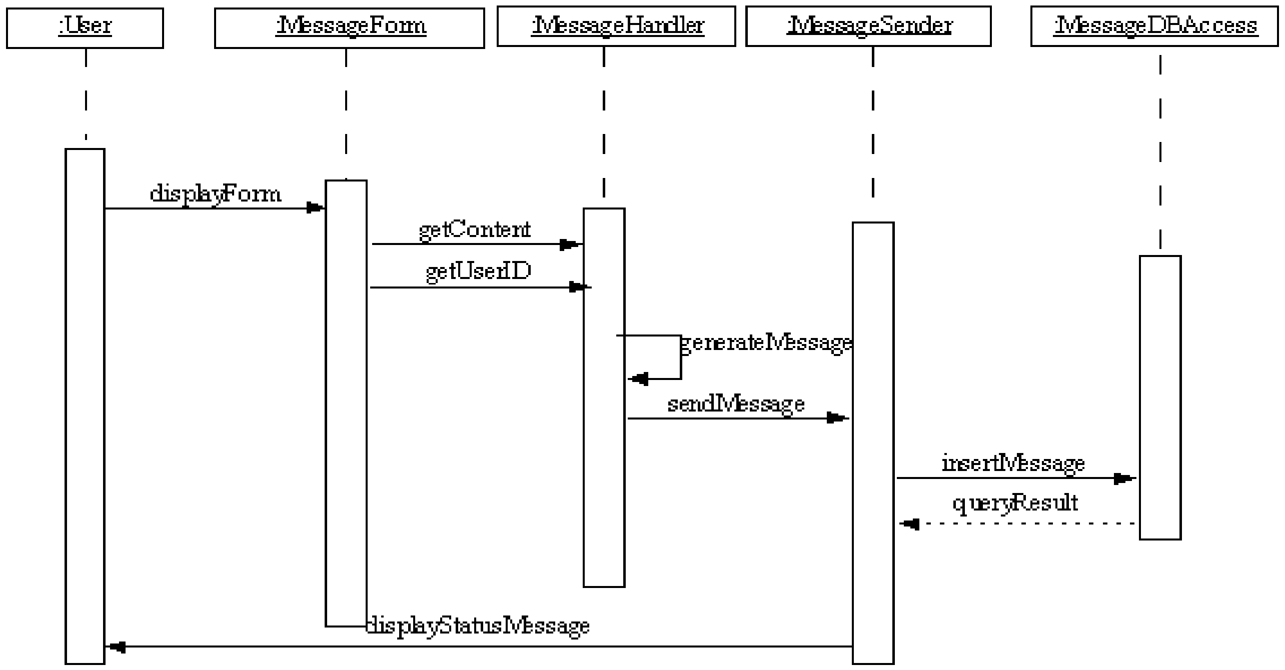


6.1.9 Web User Operations

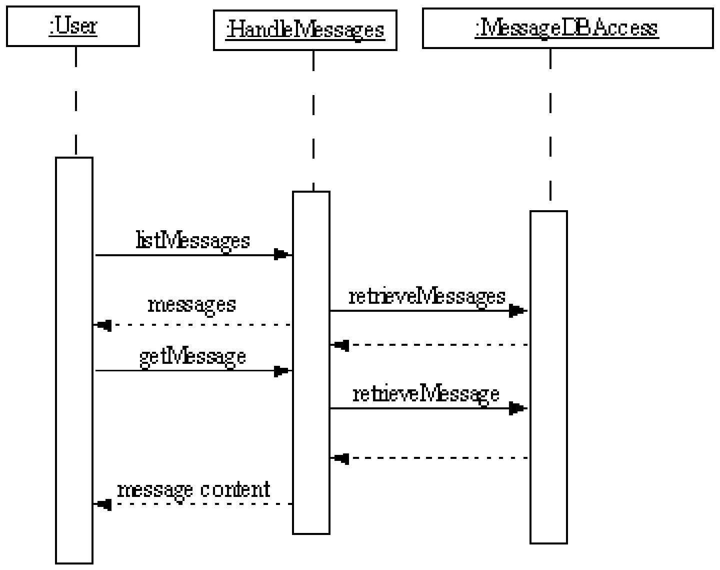


For unauthenticated web users, the user does not login to the system and can request only a small set of article operations.

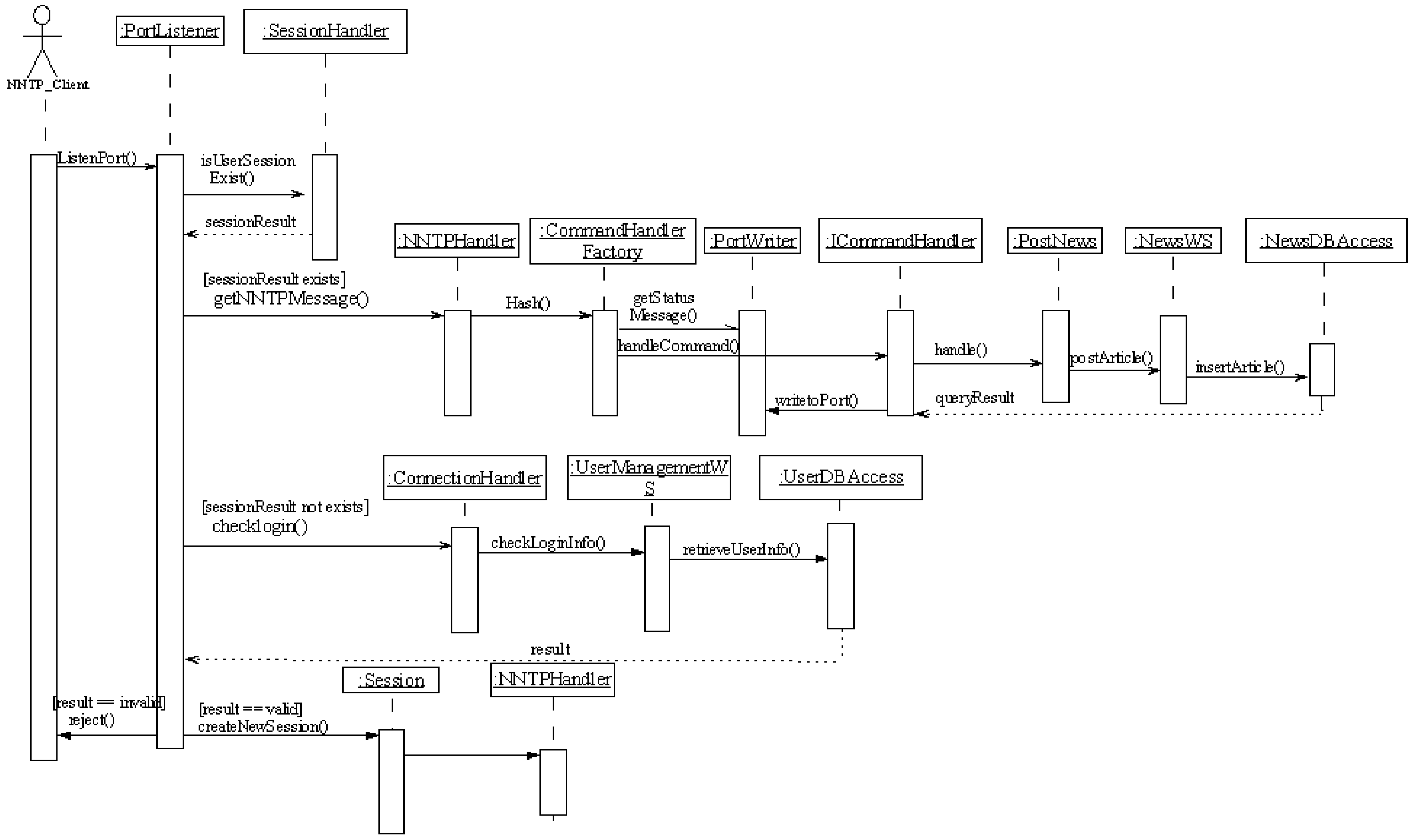
6.1.10 Sending Messages



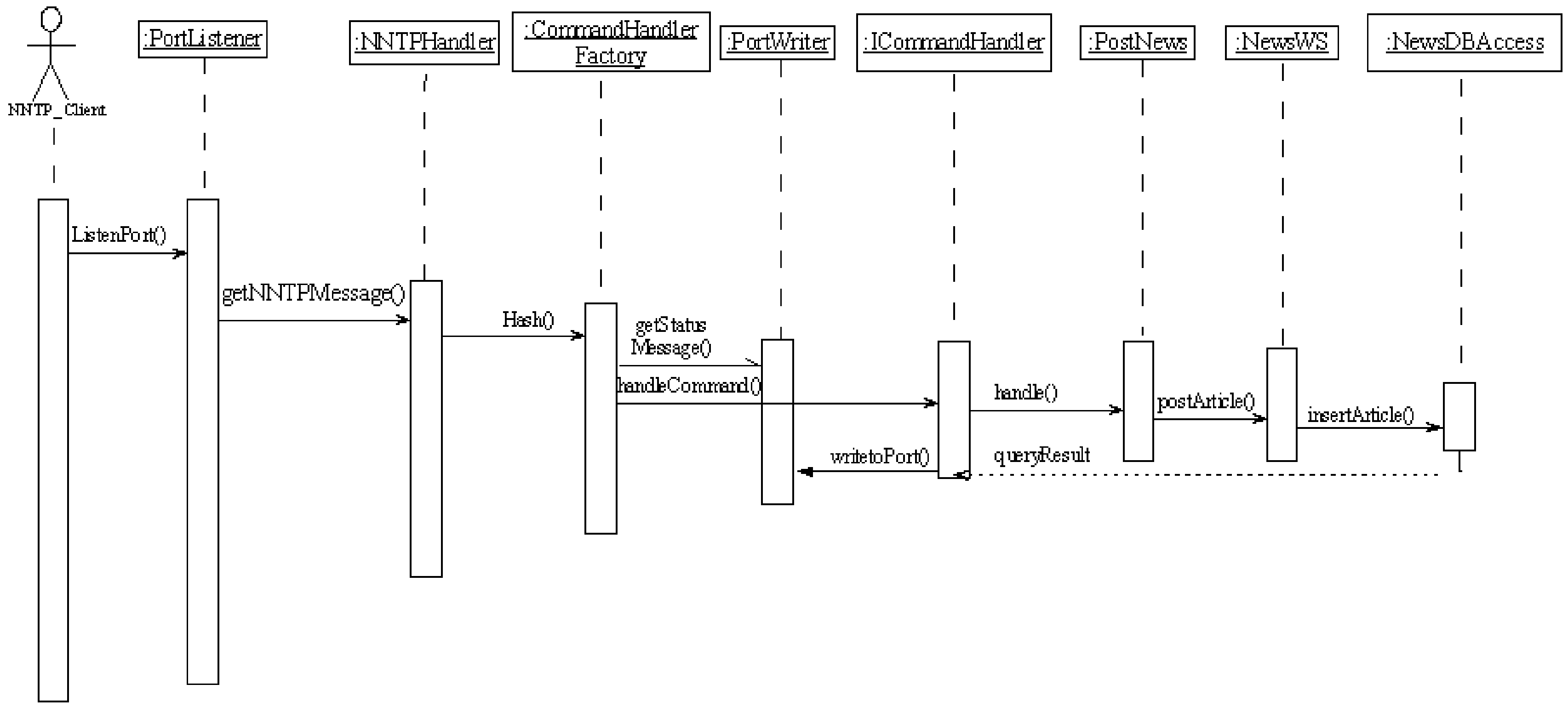
6.1.11 Reading Messages



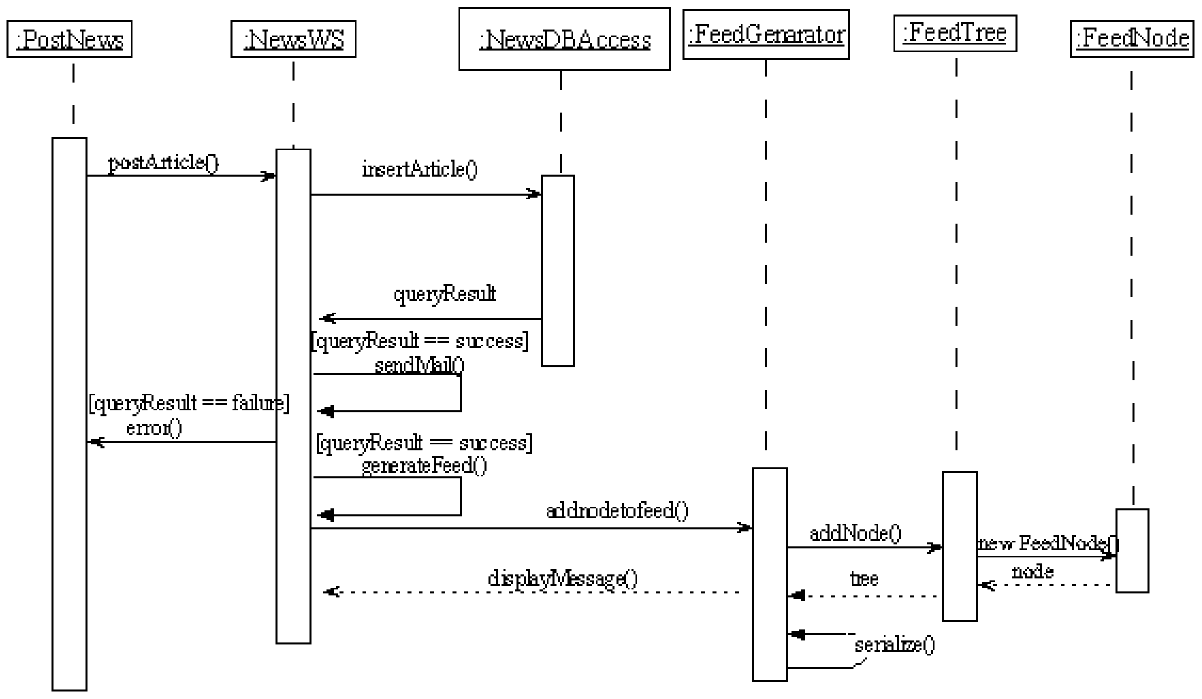
6.1.12 Authenticated NNTP User Operations



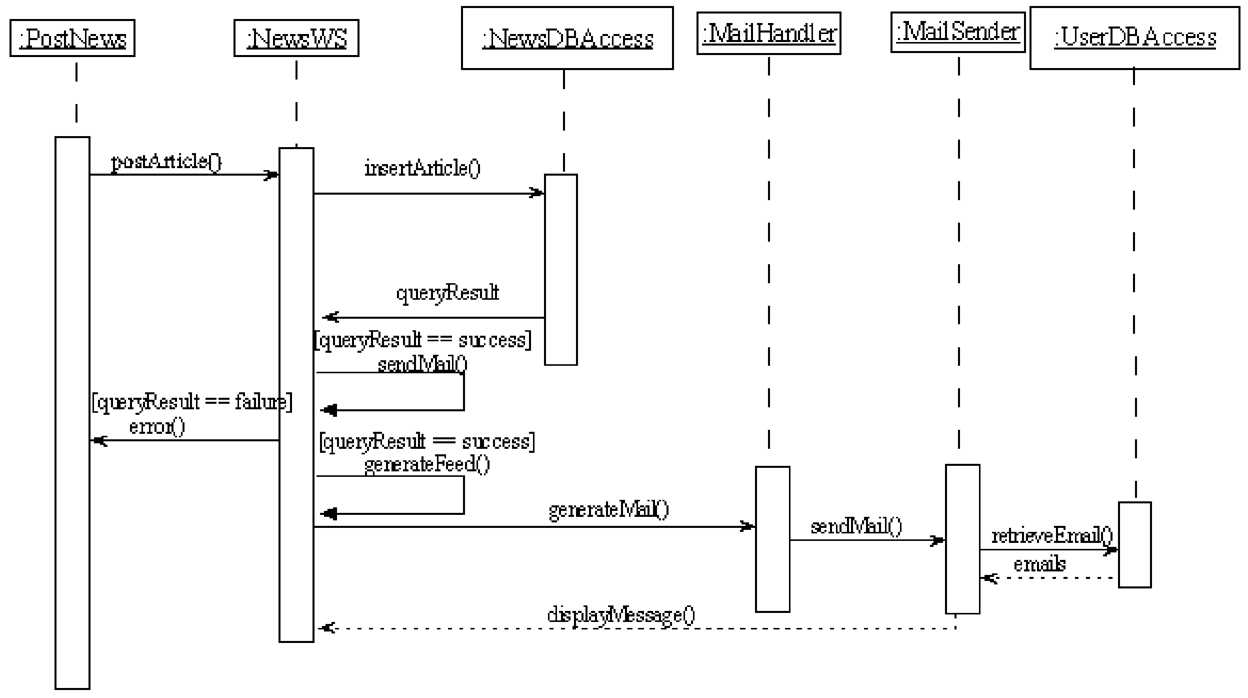
6.1.13 Unauthenticated NNTP User Operations



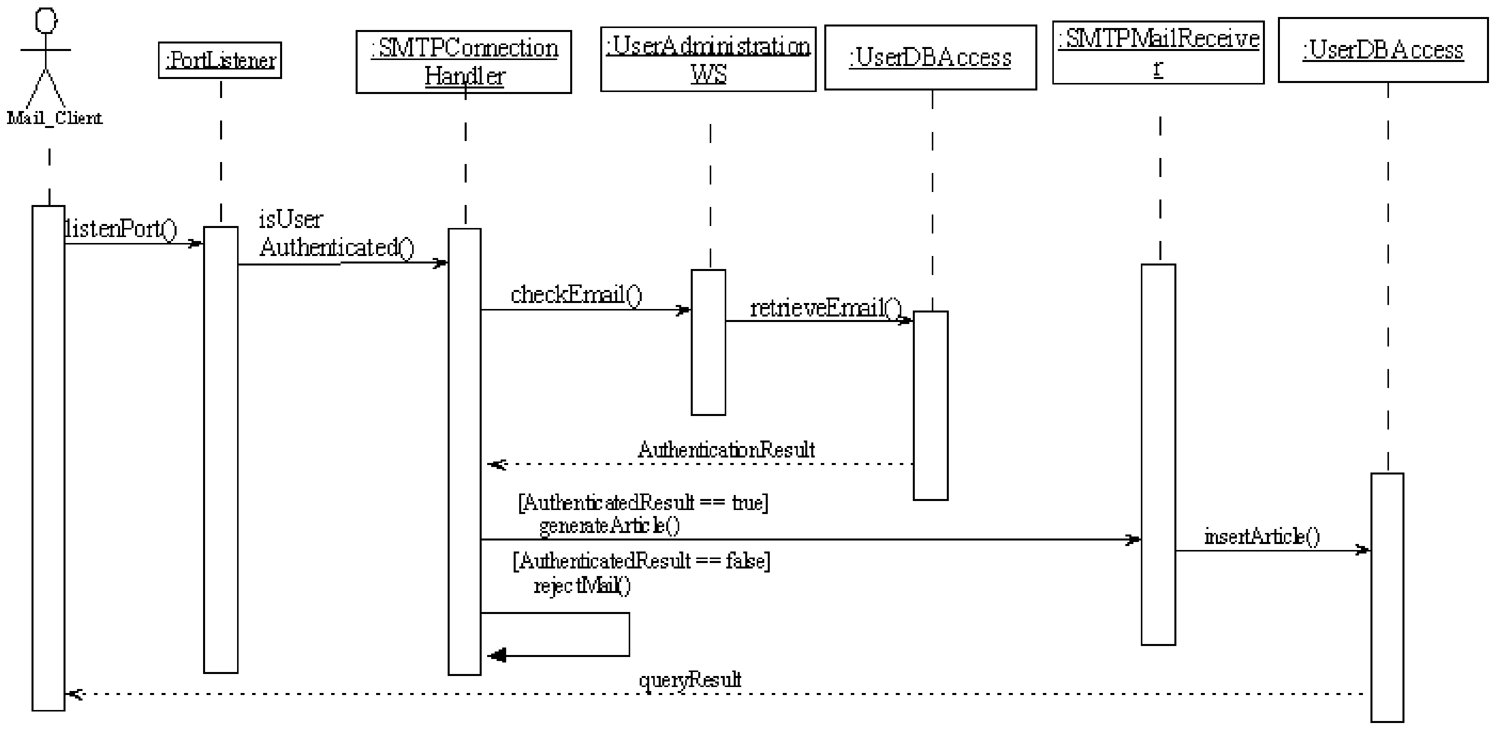
6.1.14 Feed Generation



6.1.15 Sending E-mails



6.1.16 Receiving E-mails



6.2 Sequence of Events

Login and Authentication

Main Sequence

1. The user sends his/her username and password to the controlLogin unit.
2. ControlLogin gets username and password and invokes the checkLogin() method of UserManagementWS.
3. UserManagementWS calls sendLoginInfo() method of Login in order to send username and password to the database.
4. UserDatabaseAccess is activated in order to check username and password with the database.
5. After checking the login data, according to the query result if the query result is failure displayError() method is called in order to inform the user about unsuccessful login. If the query result is success, createSession() method of Session unit is called and addLoginLog() is invoked.
6. LogWS activates the LogDatabaseAccess in order to insert login log to the database.

Sign up

Main Sequence

1. A candidate user requests to sign up to the system.
2. NewUserForm unit gets this request and display a user form.
3. getUserInfo() method of User is called and user info is stored in a User object.
4. User info is sent to the database by activating insertNewUser() method of UserDatabaseAccess.
5. According to the queryResult returned, if the user is added successfully, configuration mail is sent.
6. If it is accepted, inserted user info is updated. User group and access rights are determined. If not, the user info is deleted.

User Management

Main Sequence

1. Administrator sends a user management request in the AdministratorForm unit.
2. Administrator is directed to UserManagementForm unit.
3. Administrator requests to add a new user.

4. addUser() method of the UserManagementWS is invoked.
5. UserManagementWS calls the addUser() method of UserAdministration.
6. In order to insert new user, UserDatabaseAccess is activated with the insertNewUser() method.

Alternative Sequence

3. Administrator requests to delete a user.
4. deleteUser() method of the UserManagementWS is invoked.
5. UserManagementWS calls the deleteUser() method of UserAdministration.
6. In order to delete user, UserDatabaseAccess is activated with the deleteUser() method.

Alternative Sequence

3. Administrator requests to update a user.
4. updateUser() method of the UserManagementWS is invoked.
5. UserManagementWS calls the updateUser() method of UserAdministration.
6. In order to update user, UserDatabaseAccess is activated with the updateUserInfo() method.
7. According to the queryResult returned, if the operation is successful, addConfigurationLog() method of the LogWS is invoked.
8. insertConfigLog() method of LogDatabaseAccess is called.
9. A message is displayed to the admin denoting the success of the operation.

User Group Management

Main Sequence

1. Administrator sends a user group management request in AdministratorForm unit.
2. Administrator is directed to UserManagementForm unit.
3. Administrator requests to add a new user group.
4. addUserGroup() method of the UserManagementWS is invoked.
5. UserManagementWS calls the addUserGroup() method of UserAdministration.
6. In order to insert new user group, UserDatabaseAccess is activated with the insertNewUserGroup() method.

Alternative Sequence

3. Administrator requests to delete a user group.
4. deleteUserGroup() method of the UserManagementWS is invoked.

5. UserManagementWS calls the deleteUserGroup() method of UserAdministration.
6. In order to delete user group, UserDatabaseAccess is activated with the deleteUserGroup() method.

Alternative Sequence

3. Administrator requests to modify user rights.
4. modifyUserRights() method of the UserManagementWS is invoked.
5. UserManagementWS calls the modifyUserRights() method of UserAdministration.
6. UserDatabaseAccess is activated with the updateUserGroup() method.

Alternative Sequence

3. Administrator requests to list user groups.
4. ListUserGroups() method of the UserManagementWS is invoked.
5. UserManagementWS calls the ListUserGroups() method of UserAdministration.
6. UserDatabaseAccess is activated with the retrieveUserGroups() method.
7. According to the queryResult returned, if the operation is successful, addConfigurationLog() method of the LogWS is invoked.
8. insertConfigLog() method of LogDatabaseAccess is called.
9. A message is displayed to the admin denoting the success of the operation.

NewsGroup Management

Main Sequence

1. Administrator sends a newsgroup management request in AdministratorForm unit.
2. Administrator is directed to NGManagementForm unit.
3. Administrator requests to add a new newsgroup.
4. addNewsgroup() method of the NgManagementWS is invoked.
5. In order to insert new newsgroup, NgDatabaseAccess is activated with the insertNewsgroup() method.

Alternative Sequence

3. Administrator requests to delete a newsgroup.
4. deleteNewsgroup() method of the NgManagementWS is invoked.
5. In order to delete newsgroup, NgDatabaseAccess is activated with the deleteNewsGroup() method.

Alternative Sequence

3. Administrator requests to modify newsgroup.
4. modifyNewsgroup() method of the NgManagementWS is invoked.

5. NgDatabaseAccess is activated with the modifyNewsroup() method.

Alternative Sequence

3. Administrator requests to list newsgroups.
4. ListNewsgroups() method of the NgManagementWS is invoked.
5. NgDatabaseAccess is activated with the retrieveNewsgroups() method.
6. According to the queryResult returned, if the operation is successful, addConfigurationLog() method of the LogWS is invoked.
7. insertConfigLog() method of LogDatabaseAccess is called.
8. A message is displayed to the admin denoting the success of the operation.

Admin Log Control

Main Sequence

1. Admin requests to control login and configuration logs.
2. getConfigurationLog() / getLoginLog() method of LogWS is invoked.
3. LogDatabaseAccess is activated with retrieveConfigurationLog / retrieveLoginLog() method.

Alternative Sequence

2. deleteConfigurationLog() / deleteLoginLog() method of LogWS is invoked.
3. LogDatabaseAccess is activated with deleteConfigurationLog / deleteLoginLog() method.

Alternative Sequence

2. modifyConfigurationLog() / modifyLoginLog() method of LogWS is invoked.
3. LogDatabaseAccess is activated with updateConfigurationLog / updateLoginLog() method.
4. According to the query result returned, a message is displayed to the admin denoting the success of the operation.

Subscription

Main Sequence

1. User lists newsgroups.
2. handleNewsgroups() method is called and the subscriptionForm is also displayed to the user.
3. The user requests to subscribe or set/reset mailing option by activating subscribe() or requestEmail() methods of Subscription.

4. NGDatabaseAccess is activated with the validateNG() method in order to control the access rights of the newsgroup and the user etc.
5. A validationResult is returned and according to the validationResult if it is invalid, the operation is rejected. If it is valid, subscription() method of the UserManagementWS is invoked.
6. UserDatabaseAccess is activated with the insertSubsInfo() method.
7. A result denoting the success of the query is returned.
8. If the result is a failure, the user is directed to the subscription form. If it is success, a message is displayed to the user.

Update User Info

Main Sequence

1. User requests to update user info or change password.
2. editUserInfo() or changePassword() method of UserForm is called.
3. Current user info or login data is retrieved and displayed to the user by the help of UserInfoForm.
4. updateUserInfo() or changePassword() method of UserManagementWS is invoked
5. UserDatabaseAccess is activated with updateUser method and the changes are saved to the database.
6. A queryResult is returned denoting the success of the query.
7. Finally, changes and updates are displayed to the user.

Web User Operations

Main Sequence

1. The user sends his/her username and password to the controlLogin unit.
2. ControlLogin gets username and password and invokes the checkLogin() method of UserManagementWS.
3. UserManagementWS calls sendLoginInfo() method of Login in order to send username and password to the database.
4. UserDatabaseAccess is activated in order to check username and password with the database.
5. After checking the login data, a queryResult is returned denoting the success of the login.

6. If the queryResult is success, the user will have the right to realize web user operations. For example, the user may request to realize article operations.
7. Article operations such as read article, post article etc.is activated by the help of postArticle() / readArticle() methods of HandleNewsGroups.
8. postArticle() / readArticle() methods of NewsWS are invoked for these operations.
9. NewsDatabaseAccess is activated with the insertArticle() / retrieveArticle() methods in order to insert to posted article to the database or retrieve the requested article from database.
10. A result is returned.
11. According to the returned result, a message denoting the success or failure of the operation or the article retrieved is displayed to the user.

Sending Messages

Main Sequence

1. The user requests to send message to either an online or an offline user.
2. displayForm method of MessageForm is called for a selected user to send a message.
3. MessageHandler retrieves the content of the message and the user id of the receiver.
4. It generates a message and activates MessageSender with the method sendMessage.
5. MessageSender interacts with the MessageDBAccess and activates insertMessage method.
6. According to the queryResult, a status message denoting the success or failure of the sending message operation is displayed to the user.

Reading Messages

Main Sequence

1. The user requests to list the messages he/she received.
2. listMessages method of HandleMessage is activated.
3. HandleMessage activates the retrieveMessages method of MessageDBAccess in order to get the messages of that user.
4. Messages are returned to the user with an overview of displaying subject sender and date etc.
5. If the user requests to read a message by clicking on it, getMessage method of HandleMessage is activated.

6. HandleMessages activates the retrieveMessage method of MessageDBAccess in order to get the message content.
7. Message Content is displayed to the user.

Authenticated NNTP User Operations

Main Sequence

1. PortListener listens to the related port in order to serve NNTP client requests.
2. When a request comes, SessionHandler is activated in order to check wether the user session exists or not.
3. A session result is returned and if the session exists for the user, command is passed to the NNTPHandler.
4. NNTPHandler creates CommandHandlerFactory object in order to hash the command.
5. The command is mapped to one of the CommandHandler classes such as PostNews, List, ReadNews etc.
6. This class handles the command and invokes the related web service of NewsWS. For example for the post operation, postArticle() is invoked.
7. NewsDatabaseAccess is activated with the related method for database operation. For example for the post operation, insertArticle() method is called.
8. A queryResult is returned.
9. According to the result, PortWriter is activated with the writetoPort() method and related data is written to the port.

Alternative Sequence

3. If the session does not exist, checkLogin() method of the connectionHandler is called.
4. checkLogin() method of the UserManagementWS is invoked in order to control login data.
5. UserDatabaseAccess is activated with checkLoginInfo() method.
6. A result is returned denoting the success of the login data control.
7. If the result is invalid, the operation is rejected.
8. If the result is valid, a new session is created.
9. After creation of the session, the NNTP command is directed to the NNTPHandler and the same sequence is followed.

UnAuthenticated NNTP User Operations

Unauthenticated NNTP users only realize a small set of operations whch do not require being an authenticated user.

Main Sequence

1. PortListener listens to the related port in order to serve NNTP client requests.
2. When a request comes, SessionHandler is activated in order to check wether the user session exists or not.
3. A session result is returned and if the session exists for the user, command is passed to the NNTPHandler.
4. NNTPHandler creates CommandHandlerFactory object in order to hash the command.
5. The command is mapped to one of the CommandHandler classes such as PostNews, List, ReadNews etc.
6. This class handles the command and invokes the related web service of NewsWS. For example for the post operation, postArticle() is invoked.
7. NewsDatabaseAccess is activated with the related method for database operation. For example for the post operation, insertArticle() method is called.
8. A queryResult is returned.
9. According to the result, PortWriter is activated with the writetoPort() method and related data is written to the port.

Feed Generation

Main Sequence

1. A post article operation is accomplished, by invoking postArticle() method of NewsWS and inserting the article to the database.
2. If the article is successfully inserted, generateFeed() method of NewsWS is invoked in order to generate a new RSS and ATOM feed.
3. NewsWS calls addnodetofeed() method of FeedGenerator in order to add the last posted article to the related feed.
4. FeedGenerator accesses the FeedTree object and calls its addNode() method.
5. addNode() method of the FeedTree creates a new FeedNode and appends this new node to the current feed tree and returns this tree.
6. FeedGenerator gets the updated tree and serialize its content.

Sending Emails

Main Sequence

1. A post article operation is accomplished, by invoking postArticle() method of NewsWS and inserting the article to the database.
2. If the article is successfully inserted, sendMail() method of NewsWS is invoked in order to send email to the users who requests to receive email from that newsgroup simultaneously.
3. NewsWS calls generateMail() method of MailHandler in order to form an email from the related article.
4. After generating the email, sendMail() method of the MailSender is called.
5. In order to get the email addresses of the users who request to receive email from that newsgroup, UserDatabaseAccess is activated with retrieveEmail() method.
6. Email addresses of the related users are retrieved from database and emails are sent to these addresses.

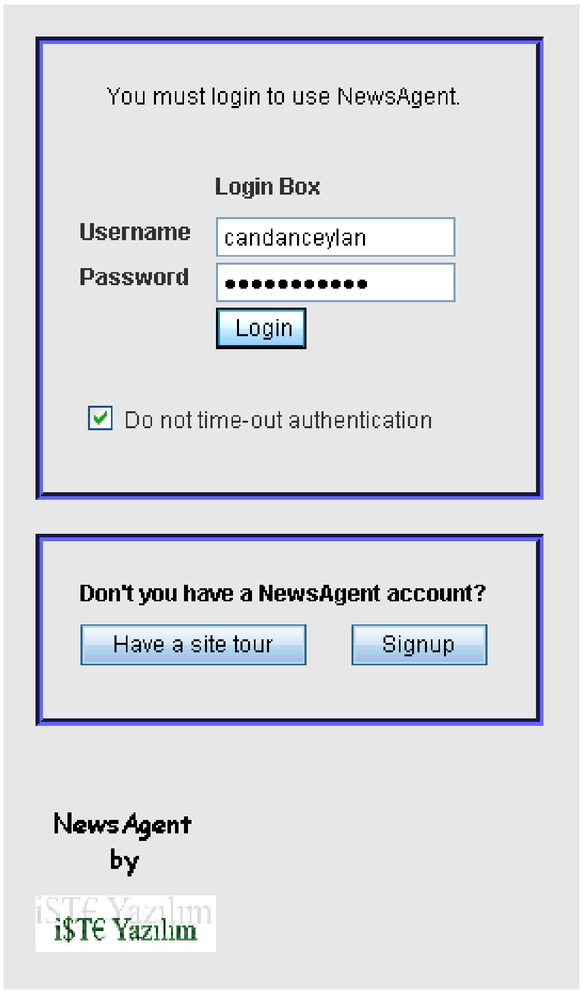
Receiving Emails

Main Sequence

1. Our PortListener listens port related to the incoming emails.
2. When an email is received, SMTPConnectionHandler is activated in order to check whether the mail client is registered or not.
3. checkEmail() method of the UserManagementWS is invoked in order to control the email address of the user.
4. UserDatabaseAccess is activated with retrieveEmail() method.
5. A result is returned and according to the result, if such an email address is registered, generateArticle() method of SMTPMailReceiver is called in order to generate an article from the received email.
6. NewsDatabaseAccess is activated in order to insert the generated article to the database.
7. If such an email address is not registered, email is rejected.

7 NewsAgent INTERFACE

Login Interface


You must login to use NewsAgent.

Login Box

Username candanceylan

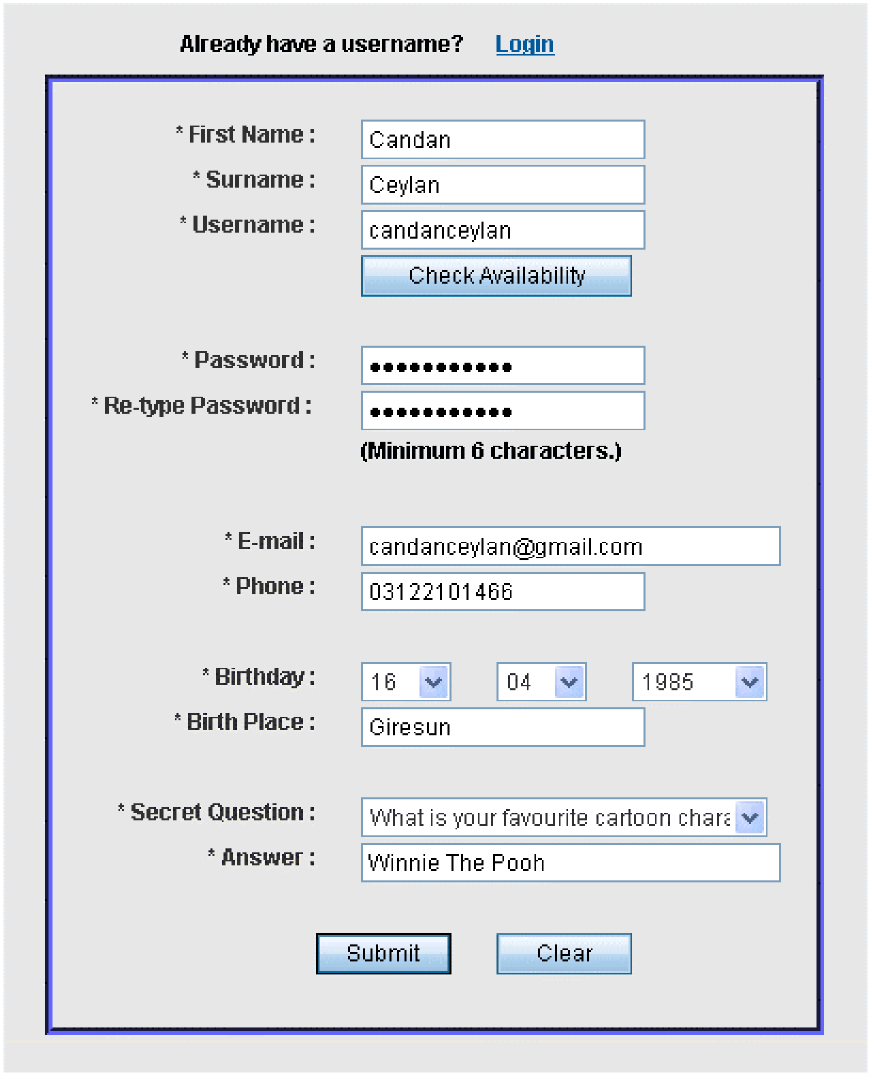
Password ●●●●●●●●●●●●

[✓] Do not time-out authentication

Don't you have a NewsAgent account?

In our login interface, we included two different panels. The first one is for user who have already signed-up to NewsAgent, that is the users who have a username and password. As usual, username and password fiels are expected to be filled with a valid username and password tuple. If the login data is correct, than the user is directed to the main page according to the user type. Administrators will be directed to admin page. If the user checks the check-box which lies under "Login" button, user's session will not time-out. Otherwise, when a specified time (1 hour, for example) passes without any user action, the session will time-out. In other panel, if the people who do not have an account click "Signup" button, he/she will be directed to "Signup Interface". If "Have a site tour" is clicked, since the user is not authenticated, he/she will be directed to a general page including the newsgroups that do not require authentication if there are any.

Signup Interface



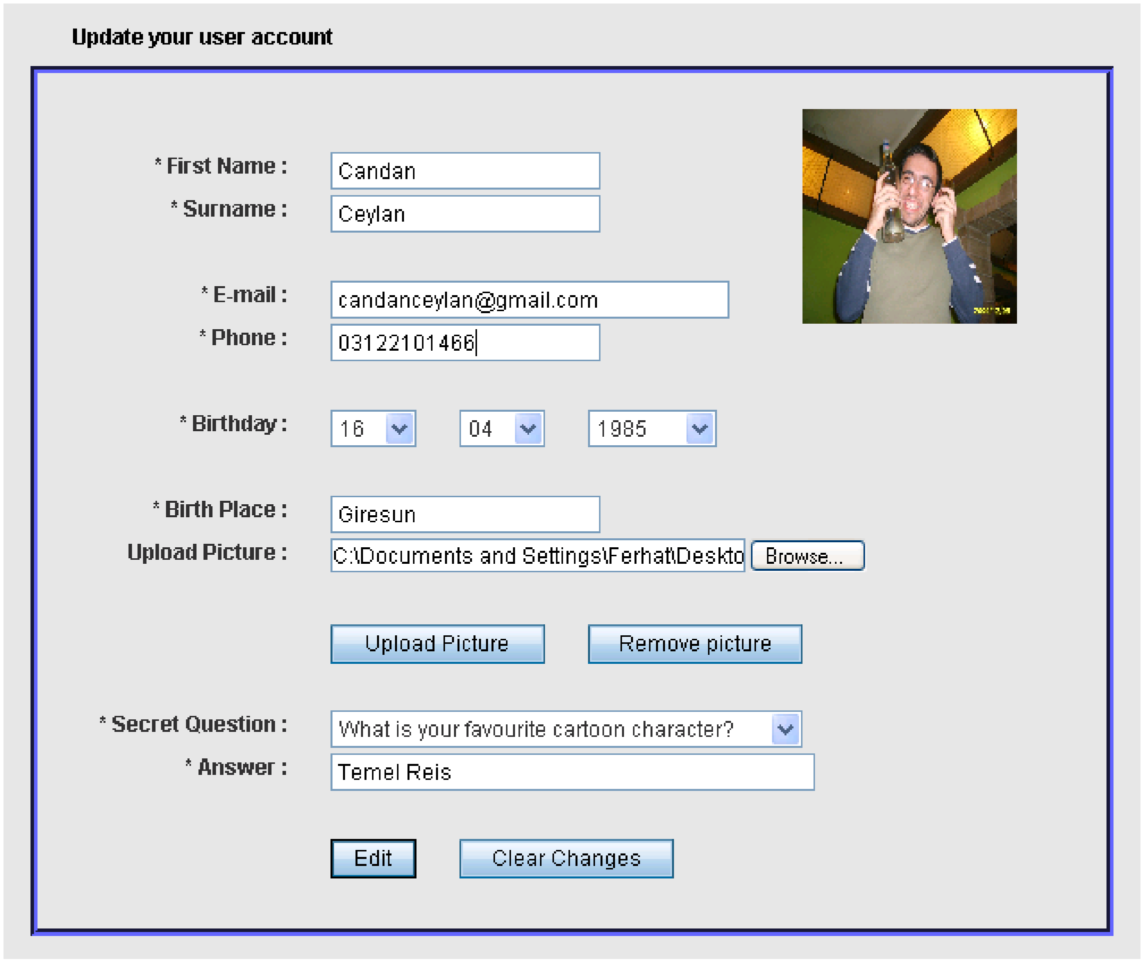
Already have a username? Login

* First Name :	Candan
* Surname :	Ceylan
* Username :	candanceylan
	Check Availability
* Password :	●●●●●●●●●●●
* Re-type Password :	●●●●●●●●●●●
	(Minimum 6 characters.)
* E-mail :	candanceylan@gmail.com
* Phone :	03122101466
* Birthday :	16 / 04 / 1985
* Birth Place :	Giresun
* Secret Question :	What is your favourite cartoon chara
* Answer :	Winnie The Pooh

In signup interface, we have the fields which are required to be filled in order to add the candidate user as a system user. If anyone who is already a user presses "Signup" button in login screen accidentally, he/she can return using the hyperlink "Login" here. Firstname and surname are required. Username is selected by the user, however, since it is unique in the system, user can check the availability of the username pressing "Check Availability" button. If it is already used, user has to choose another username. User has to choose a password which is minimum 8 characters long and has to retype it in order to verify. E-mail and phone number are also required. Invalid e-mails will not be accepted. This is accomplished by a confirmation link which is sent to this mail address by the administrators. The user's account will be activated when he/she follows this link. Birthday and birth place are also required.

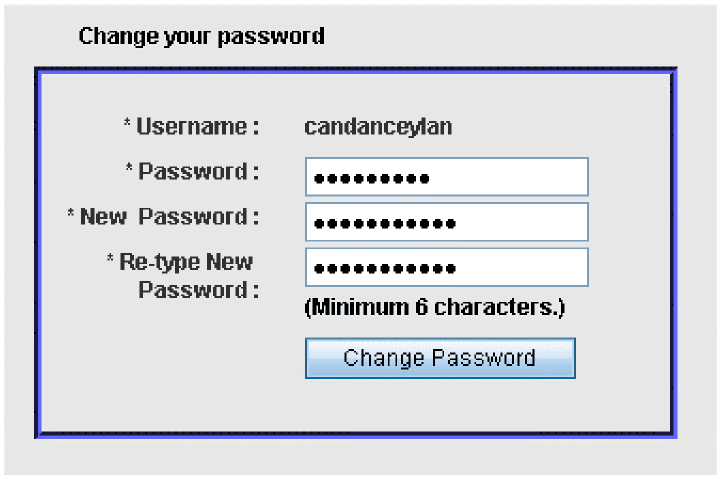
Secret question is any question that the user selects among the ones we offered. The question and answer are kept in order to use if the user forgets his/her password. By using "Submit" button the user can send the form to the administrators and waits until the account is activated. By "Clear" button, user can clear the screen.

Update User Info & Change Password Interfaces


Update your user account

* First Name :	Candan
* Surname :	Ceylan
* E-mail :	candanceylan@gmail.com
* Phone :	03122101466
* Birthday :	16 04 1985
* Birth Place :	Giresun
Upload Picture :	C:\Documents and Settings\Ferhat\Deskto
* Secret Question :	What is your favourite cartoon character?
* Answer :	Temel Reis

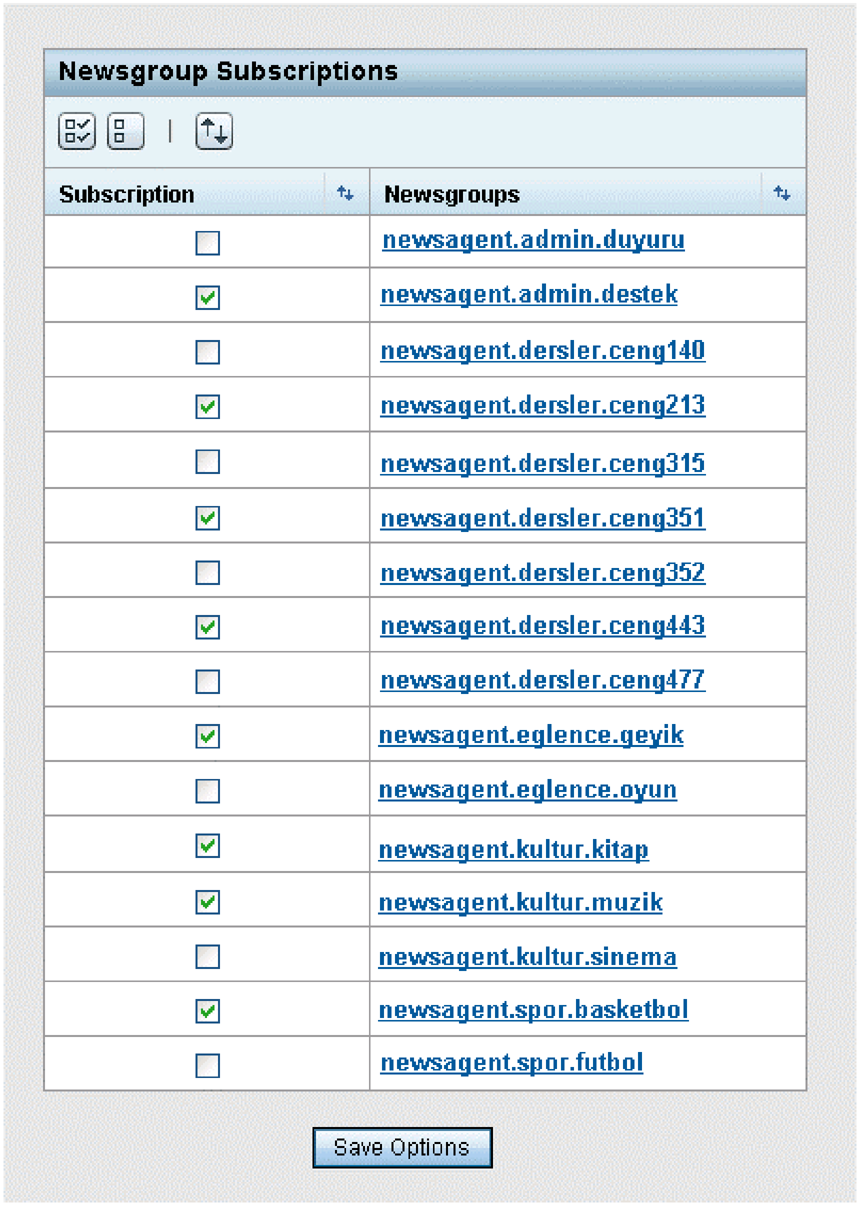
In "Update User Info" screen, first name, last name, email, phone number, birthday, birthplace, secret question and its answer are displayed. These fields will be enabled and user will be able to update these information. "Edit" button saves changes and "Clear Changes" button clears the changes. "Upload Picture" part is optional and by clicking "Browse..." button, user can select a picture from the computer he/she uses. If there is already a picter in the user's account, uploaded one is written on it. If "Remove Picture" button is clicked, existing picture of the user will be removed.


Change your password

* Username :	candanceylan
* Password :	●●●●●●●●●
* New Password :	●●●●●●●●●●●
* Re-type New Password :	●●●●●●●●●●●
	(Minimum 6 characters.)

In “Change Password” screen, username is displayed but the user will not be able to change it. In “Password” field, old password is expected to be entered. And new password is expected to be entered 2 times in order to verify it. Then the password is changed by pressing “Change Password” button.

Interface for Newsgroup Subscriptions

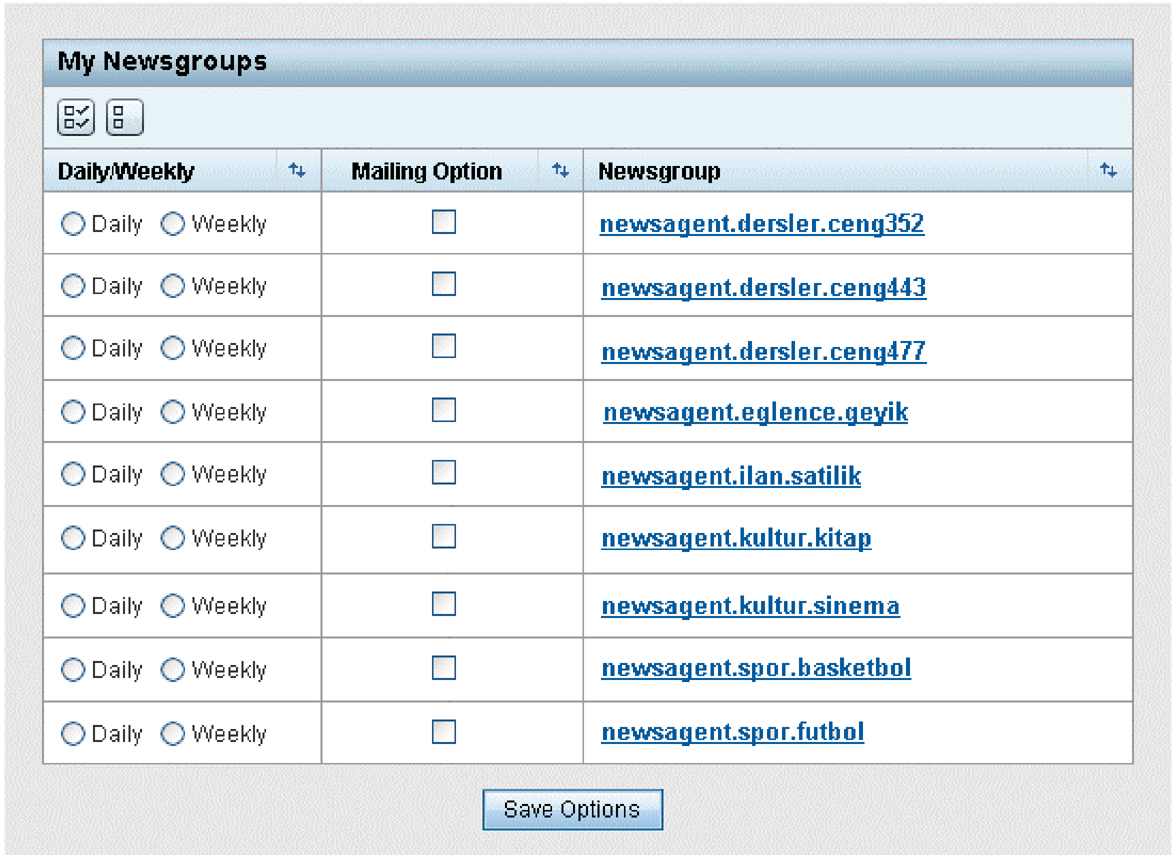


Newsgroup Subscriptions

Subscription	Newsgroups
☐	newsagent.admin.duyuru
✓	newsagent.admin.destek
☐	newsagent.dersler.ceng140
✓	newsagent.dersler.ceng213
☐	newsagent.dersler.ceng315
✓	newsagent.dersler.ceng351
☐	newsagent.dersler.ceng352
✓	newsagent.dersler.ceng443
☐	newsagent.dersler.ceng477
✓	newsagent.eglence.geyik
☐	newsagent.eglence.oyun
✓	newsagent.kultur.kitap
✓	newsagent.kultur.muzik
☐	newsagent.kultur.sinema
✓	newsagent.spor.basketbol
☐	newsagent.spor.futbol

This interface displays all available newsgroups for the user. If the user is subscribed to a newsgroup, the check-box for that newsgroup is displayed checked. If the user wants to subscribe or unsubscribe to a newsgroup, he/she checks or unchecks the check-box and click "Save Options" button.

“My Newsgroups” Interface

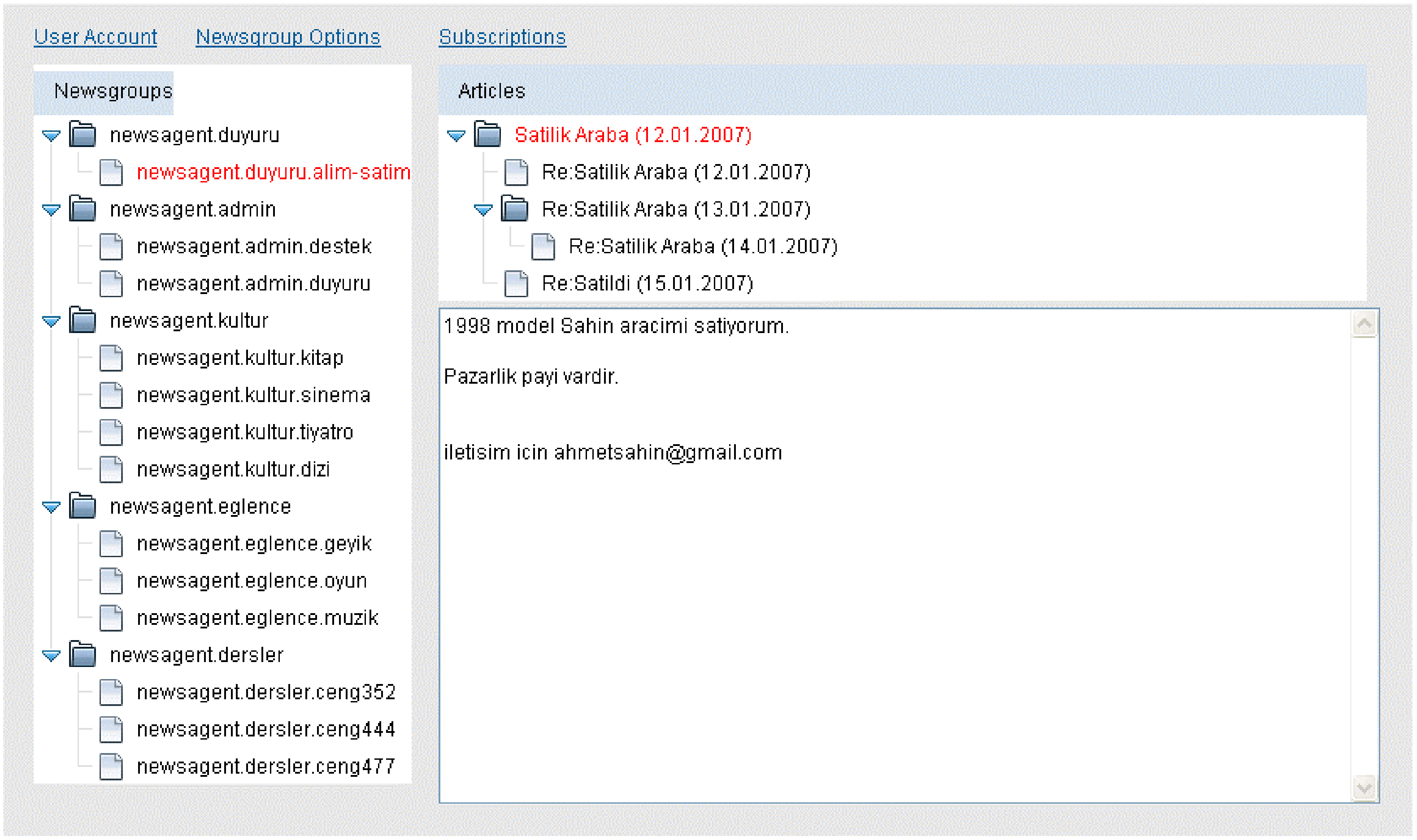


My Newsgroups

Daily/Weekly	Mailing Option	Newsgroup
○ Daily ○ Weekly	☐	newsagent.dersler.ceng352
○ Daily ○ Weekly	☐	newsagent.dersler.ceng443
○ Daily ○ Weekly	☐	newsagent.dersler.ceng477
○ Daily ○ Weekly	☐	newsagent.eglence.geyik
○ Daily ○ Weekly	☐	newsagent.ilan.satilik
○ Daily ○ Weekly	☐	newsagent.kultur.kitap
○ Daily ○ Weekly	☐	newsagent.kultur.sinema
○ Daily ○ Weekly	☐	newsagent.spor.basketbol
○ Daily ○ Weekly	☐	newsagent.spor.futbol

In this interface, we display all newsgroups that the user is subscribed to. For these newsgroups, the user can check mailing option and choose one of daily and weekly in order to receive articles as e-mails from that newsgroup. If daily is checked, articles will be sent to the users daily and if weekly is checked, articles will be sent to the user weekly. “Save options” button saves the changes mae on mail receiving options.

Interface For Reading Articles



1998 model Sahin aracimi satiyorum.

Pazarlik payi vardir.

iletisim icin ahmetsahin@gmail.com

In this interface, newsgroups are listed on the left side, with indicating parent-child relations. The user can select any newsgroup from left, and the headers are displayed on the right. When a header is clicked, the content of the article is displayed below the headers.

8 TESTING PLAN AND PROCEDURES

8.1 Testing Plan

Our aim is to find errors and make a good test that has a high probability of finding an error. We also want to make sure that there are no defects in the product.

After we have generated the source code, we are going to test our program to identify the errors and remove them before delivery to the customer. Our goal is to correct as many errors as possible early in our software development cycle. In order to acquire this we have to design a series of test cases that have a high likelihood of finding errors.

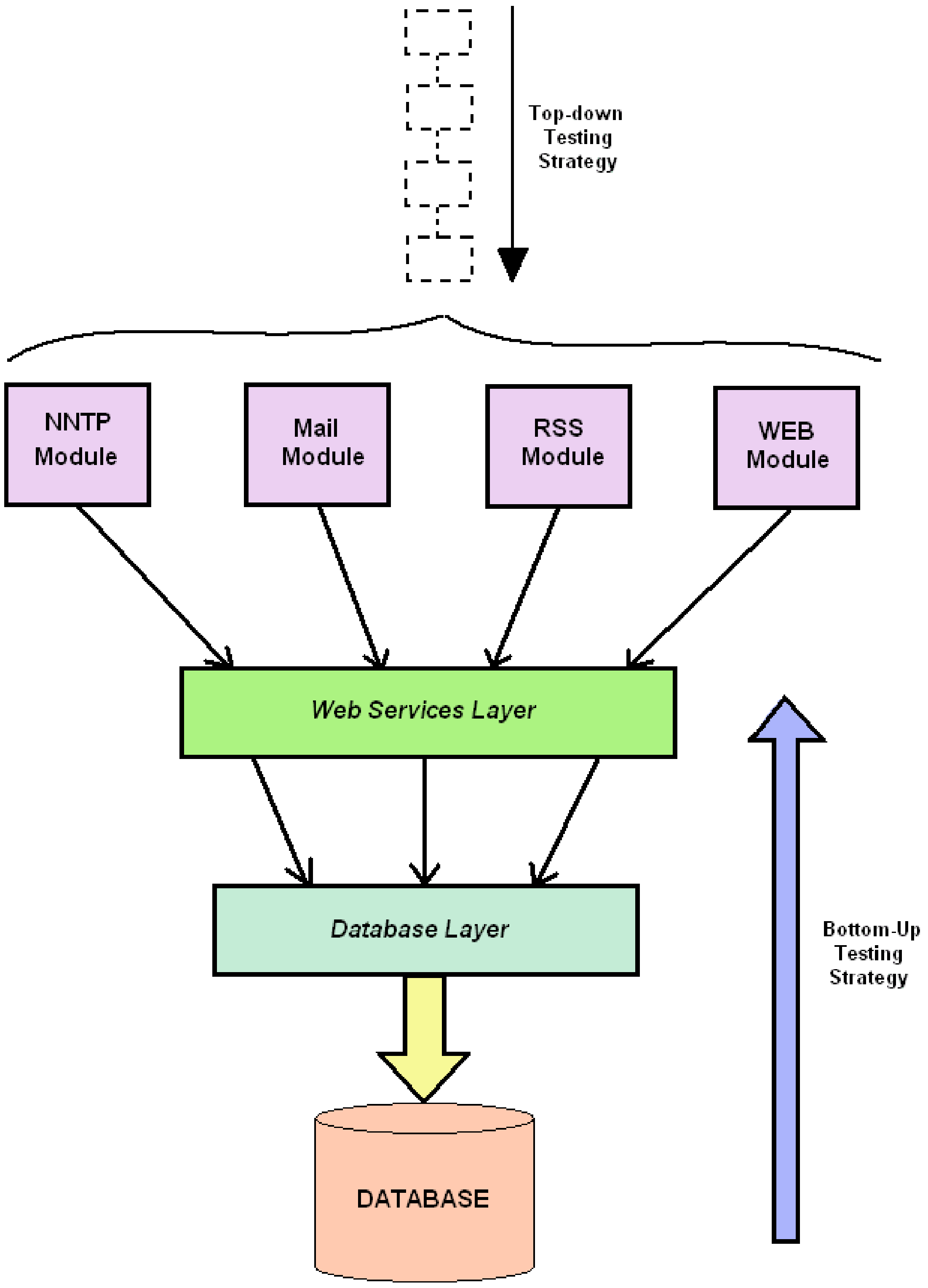
8.2 Testing Strategy

Since NewsAgent has different layers and modules, testing strategy will differ for each subpart of the product. We present a testing schema below, which will briefly explain our testing strategy.

In general, we will follow a bottom-up strategy for testing. Therefore, we will start from database layer as shown in the schema. For this layer, we will apply unit tests in order to check performance and correctness of our database queries. We will test our retrievals, insertions and modifications. Testing of this part is very important since each web service and its methods use the data returned from database layer and insert data into database through this layer. Any mistaken coding error in this layer can cause many problems in above layers.

After testing database layer, we will pass to web services layer. Any operation in NewsAgent will be handled by web services. So testing this part is another important issue in testing the product. For testing our web services, we will deploy each of them separately and invoke related methods. We will check whether each web service works correctly.

Then we will test our modules; NNTP Module, Mail Module, RSS Module and Web Module. While testing these modules, we will follow a different strategy which is top-down testing strategy.



Testing Strategy of NewsAgent

8.3 Testing Procedure

8.3.1 Unit Testing

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

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

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

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

8.3.2 Integration Testing

In this testing period we will be looking for any signs of the collision between our software components and those of the clients. We want to make sure there is no confusion among the application on the network when they are running simultaneously.

As we know, integration testing is testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications, client and server applications on a network, etc. And this type of testing is especially relevant to client/server and distributed systems. We will be carefully looking for any sort of collision between several different applications.

8.3.3 Security Testing

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

9 SYNTAX SPECIFICATION

Coding standards occupy large amounts for big projects which have multiple developers and coders. These standards are so important that some big companies, military services and governmental services only rely on the products which have been produced through a very strictly specified line. This line is determined by the rules. Every developer included in the project must obey these rules.

Not being a big company, even not a company, we can also benefit some rules to simplify the understandability and readability of the codes. As a team we will develop the system together, but most of the time we will work on the code at different time slots. So, with the help of the CVS and a predefined specification rules will prevent us the get in conflicts and doing wrong things.

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

9.1 Naming the Classes and Files

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

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

9.2 Naming Functions

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

9.3 Naming Variables

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

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

- "l" --> local variable. Indicating that the variable is defined locally.

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

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

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

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

9.4 Comment Conventions

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

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

10 IMPLEMENTATION PLAN

10.1 System Overview

System Description

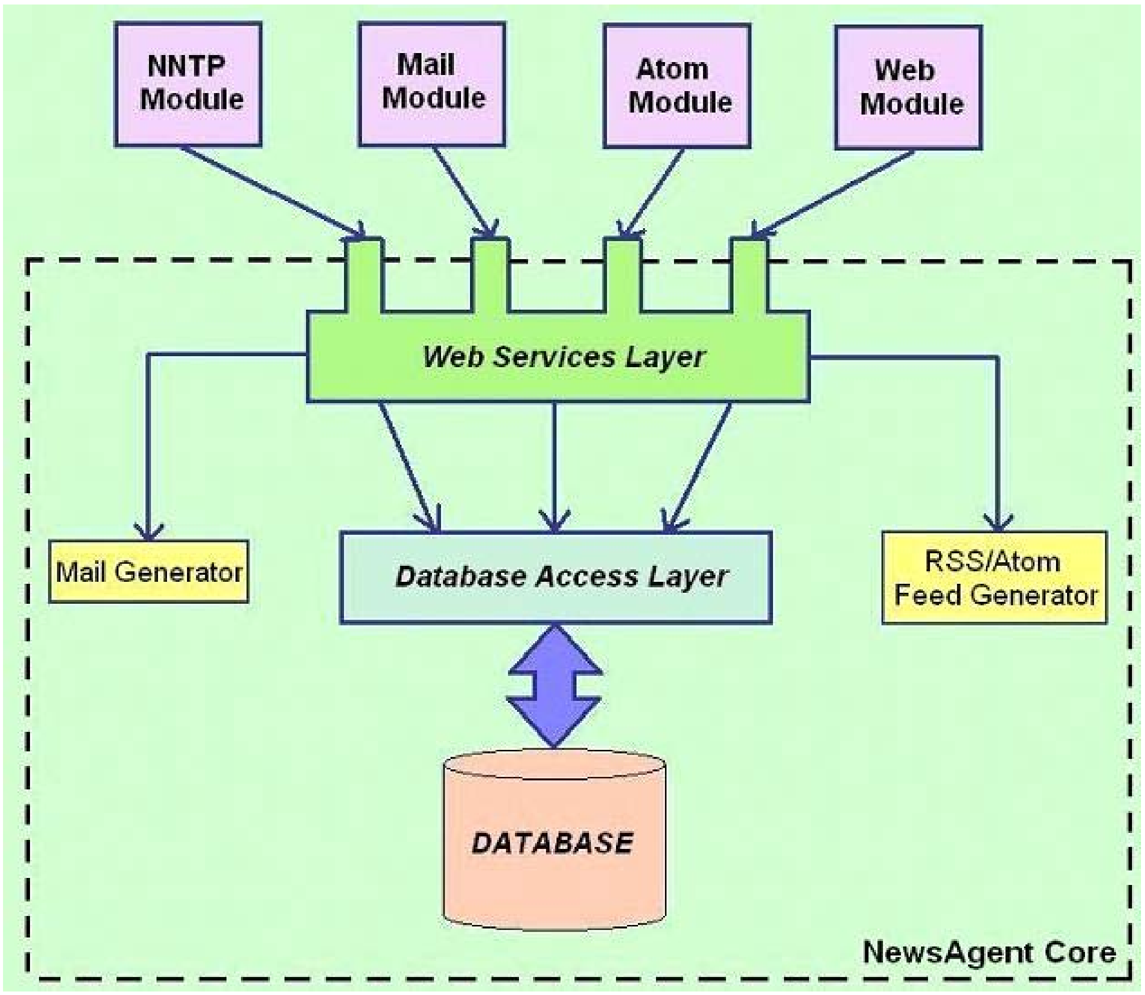
NewsAgent is mainly a pull based news server except the e-mail module because all e-mail protocols operates on push based architectures, supporting many features and standards. *NewsAgent* includes a core which operates on the data and identity management. Articles, user information and all related data is stored in a database, and the archived information is stored in another database. *NewsAgent* core is in charge of management of these databases.

NewsAgent core is in connection to the outside world only with the ports of its web services. All other modules and functioning parts reach the required data through these xml web services. This great feature encapsulates the core of the system and makes it a standalone engine. Interoperability is highly achieved by means of the web services because any other operating system, any other software implemented in any other programming language and platform can connect to the core and operate on the data consistently by the help of xml web services.

External modules of the *NewsAgent* are Web Module, NNTP Module, RSS/Atom Module and E-Mail Module. Web Module interact with the internet users through the web browsers and is the more effective and functional module of the *NewsAgent*. All user account functionalities, admin facilities and news server operations can be done through this module. NNTP Module acts as a mapping engine of the USENET NNTP commands to the appropriate web service operations and returns the required data and reply codes to the news reader. RSS/Atom Module handles the syndication operations which is very popular among the internet users nowadays. Lastly the E-Mail module gives the system the ability to communicate through the e-mail protocol to send or retrieve the articles of the newsgroups.

System Organization

Organization of the system is described in the figure below.



Web service invocations connect the modules to the core and abstract it as a standalone engine. In the core, only access way is through the access layer of the system. And if change occurs in the database which requires notification it creates the required e-mails and appends the required RSS/Atom feeds.

10.2 System Requirements

Hardware Requirements

- For Developer
  A minimum of 512 MB DDRAM
  A minimum of 5 GB free space on hard disk, for database storage and server applications
  A Pentium IV or equivalent AMD processor
  Internet Connection

Network Card

- For Server Applications
  A minimum of 1 GB DDRAM
  A minimum of 50 GB free space on hard disk, for huge database storage and large number of server applications
  A Pentium IV or equivalent AMD processor
  Internet Connection

Software Requirements

- Java as a programming language. JDK 1.5.X
- Eclipse as development environment
- Apache 2.2 HTTP server
- Apache Tomcat 5.5 for Servlet Container
- Apache Axis 1.4 for XML Web Services
- Apache WSDL2Java Tool
- TCP-Mon Tool
- PostgreSQL 8.2 Database Management System
- Hibernate for Object-Relational Database Management.
- Java Studio Creator 2 1.0

10.3 Objectives & Tasks

Although we are not a commercial company, even not a company, we will try to do our best and we will get proud of it if somebody uses benefit of out product, *NewsAgent*. So, we have determined on some objectives for this purpose.

Objectives

- Implementing the *NewsAgent* core as a standalone server and make it interoperable as much as possible.
- Implementing all the modules of the system.
- Getting the feedback from the end users. According to the feedbacks, implementing new modules and meet the rapidly changing internet technology needs.

Completed Tasks

New tasks are assigned to the team members after the date of completion of the final design report. Until this day, as Iste Team, we have worked on several modules of the *NewsAgent*.

- We have implemented part of NNTP server to handle 2 NNTP commands.
- We have created a RSS feed by software and subscribed it from a reader.
- We have used "JAMES" which includes a SMTP server for our module, we have sent and received e-mails through that program and we have parsed the e-mails.
- We have spent a lot of time on XML Web Services. We have completely deployed some services for practice. We have sent and received Java objects embedded in the SOAP messages which we will use for article and user data transfer between the modules and the *NewsAgent* core.

According to the completed tasks, now Iste Team is ready to design the implementation structure and combine the different architecture and make them work together in a very consistent way.

Major Tasks & Work Packages

Major tasks are arranged under the suitable Work Packages. Strict deadlines are determined for the Work Packages for the next semester.

Work Package 1: Core Implementation

This work package includes the implementation of the *NewsAgent* core. Core is the backbone of the system and it has many implementation details. Core implementation is divided into 3 main parts.

- **Database Layer Implementation**

  Database layer implements all the required functionalities for database access. This layer uses the benefit of the Hibernate tool. By the help of this tool database operations will be easier and more consistent.

  Database Layer operations are also divided into 2 parts, because module implementations will use the operations implemented in the database layer. So they might have been concurrently implemented.

  - **News Server Operations Implementation**

    These operations are the article and newsgroup related functionalities.

    - Article Handlers (Retrieval, Insertion, Deletion, Update)

- Article Parsers/Generators
- Newsgroups Handlers(Retrieval, Insertion, Deletion, Update)
- Newsgroup Access Rights Handlers
- Archiving Decision
- Article Archiving
- Newsgroup Archiving
- Archive Article Handlers
- Archive Newsgroup Handlers
- Article Logging Handlers
- Newsgroup Logging Handlers
- Milestone

➢ **User Operations Implementation**

These operations are user account related operations.

- User Sign-up – New Account Creation
- Password Creators
- Auto-generated Confirmation Links
- Confirmation Handlers
- User Info(Password, Demographic data, E-mail options etc…) Retrieval, Update
- User Deletion
- User Logging Handlers
- User Access Rights Handlers
- Subscription/Unsubscription Manager
- Milestone

➢ **Private Messaging and Chatting Operations Implementation**

These operations are the messaging related operations between the online users of web module of the *NewsAgent*.

- Private Message Handlers (Retrieval, Insertion, Deletion, Update)
- User-Message Handlers
- Chat Log
- Milestone

❖ Web Services Layer Implementation

Web Services Layer implements the XML Web Services and acts as a bridge between the modules and the Database Layer. Also Web Services Layer is responsible for triggering the Mail Generator and RSS/Atom Feed Generator.

Actually this layer includes the web services mapping of the functions listed for Database Layer. The extra implementations are listed as follows.

- WSDL(Web Service Description Language) Implementation
- Skeleton Implementations
- Binding Implementations
- Deployment of Services
- Mail Triggers
- RSS/Atom Triggers
- Integration
- Milestone

❖ Mail Generator Implementation

This part generates e-mail messages and sends them to the appropriate receivers upon the coming trigger from the Web Services Layer.

- Mail Generator
- Article Object Parser
- Receiver Handlers
- JAMES Server Access Layer
- E-Mail Sending
- Logging Handler
- Integration

❖ RSS/Atom Feed Generator Implementation

This part generates RSS/Atom Feeds messages and appends them to the appropriate existing feeds upon the coming trigger from the Web Services Layer.

- Feed Generation
- Feed Selection
- Feed Appending
- Feed Load Handlers
- Logging Handler

- Integration
- Milestone

Work Package 2: NNTP Module Implementation

This work package includes the implementation of the USENET NNTP module of the *NewsAgent*.

- Port Listening
- Connection Handling
- Authentication Manager
- NNTP-Command Handlers
- Security Manager
- SSL/TLS integration
- Session Manager
- Logging Handler
- Integration
- Milestone
- RELEASE: NewsAgent 1.0

Work Package 3: E-Mail Module Implementation

This work package implements the E-Mail module operating embedded in the JAMES SMTP server of Apache. It accepts the e-mails from the subscribed users. And it avoids from the spam mailing by using the confirmation strategy.

- Mail Parser
- Authentication Manager
- E-mail Confirmation Manager
- E-mail Submission Manager
- Logging Handler
- Integration
- Milestone
- RELEASE: NewsAgent 1.1

Work Package 4: Atom Module Implementation

This work package implements the Atom Module of *NewsAgent.* Atom module accepts entries from the Atom users and calls the required web services.

- Entry Manager
- Authentication Manager
- Feed Handlers
- Logging Handler
- Integration
- Milestone
- RELEASE: NewsAgent 1.2

Work Package 5: Web Module Implementation

This work package includes the implementation of the most complex module of *NewsAgent*. At this step, web module will be implemented step by step. To ensure the concurrent and consistent implementation, it is divided into

❖ **Graphical User Interface (GUI) Design**

At this part, user friendly and easy-to-use web pages will be designed.

- Home Page Design
- Sign-in Page Design
- Sign-up Page Design
- Account Information Page Design
- Article Operations Page Design
- Newsgroups Operations Page Design
- Private Messaging Page Design
- Chat Pop-up Page Design
- Integration

❖ **News Server Operations Implementation**

At this part, the designed web pages related to the news server operations such as articles and newsgroups will be converted to the functioning pages by implementing the required servlet classes and JSP pages. These classes are the corresponding servlets of the pages listed in the GUI Design Part.

- Article Operations Page Classes
- Newsgroups Operations Page Classes
- Corresponding Web Service Invocations
- Integration
- Milestone

❖ **User Account Operations Implementation**

At this part, the designed web pages related to the user operations will be converted to the functioning pages by implementing the required servlet classes and JSP pages. These classes are the corresponding servlets of the pages listed n the GUI Design Part.

- Home Page Classes
- Sign-in Page Classes
- Sign-up Page Classes
- Account Information Page Classes
- Corresponding Web Service Invocations
- Integration
- Milestone

❖ **Private Messaging and Chatting Operations Implementation**

At this part, the designed web pages related to the private messaging and chatting will be converted to the functioning pages by implementing the required servlet classes and JSP pages. These classes are the corresponding servlets of the pages listed in the GUI Design Part.

- Private Messaging Page Classes
- Chat Pop-up Page Design
- Chatting Handlers
- Synchronization Handlers
- Corresponding Web Service Invocation
- Integration
- Milestone
- RELEASE: NewsAgent 2.0

Work Package 6: Testing and Debugging

This work package includes the testing and debugging phases of the project period. At this stage it is assumed that all the functionalities are implemented and only testing issues are remained.

- Unit Testing
- Integration Testing
- Security Testing

- Robustness Tests
- Milestone
- RELEASE: Testing Reports

Work Package 7: Documentation

This work package includes the required documentation of the project.

- Installation Manual
- Users Manual
- Developers Manual

Work Package 8: Final Releasing

This work package is the final step of *NewsAgent* project. Packaging of the project and releasing of the entire project is included in this work package. Actually, this "sum up" stage includes hard tasks which include the arrangement of the installation files and release notes.

- RELEASE: NewsAgent 2.1 Final Releases

11 GANTT CHART

Gantt chart of NewsAgent is presented in APPENDIX.

12 CONCLUSION

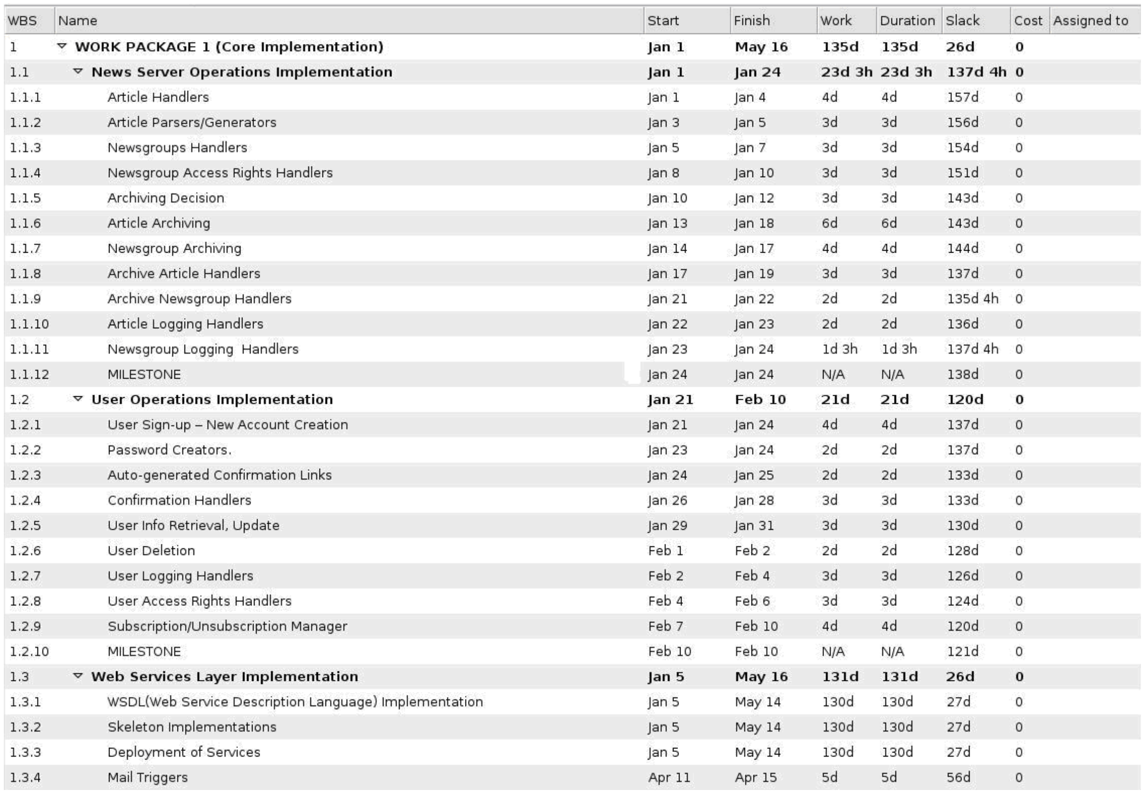
To sum up, throughout this report we presented the detailed design issues and the main structure of the system in a detailed way. Each module of the system is visualized using different diagrams and the concepts and discussions on them were explained clearly. These diagrams and discussions on different aspects of NewsAgent provide it to be handled by using different techniques which will be useful for observing different modules of NewsAgent from different point of views. We believe that we have made benefit of the detailed design report in the sense that design issues and modules of the system became stable in our minds. This design period will guide us in the implementation of the system.

13 REFERENCES

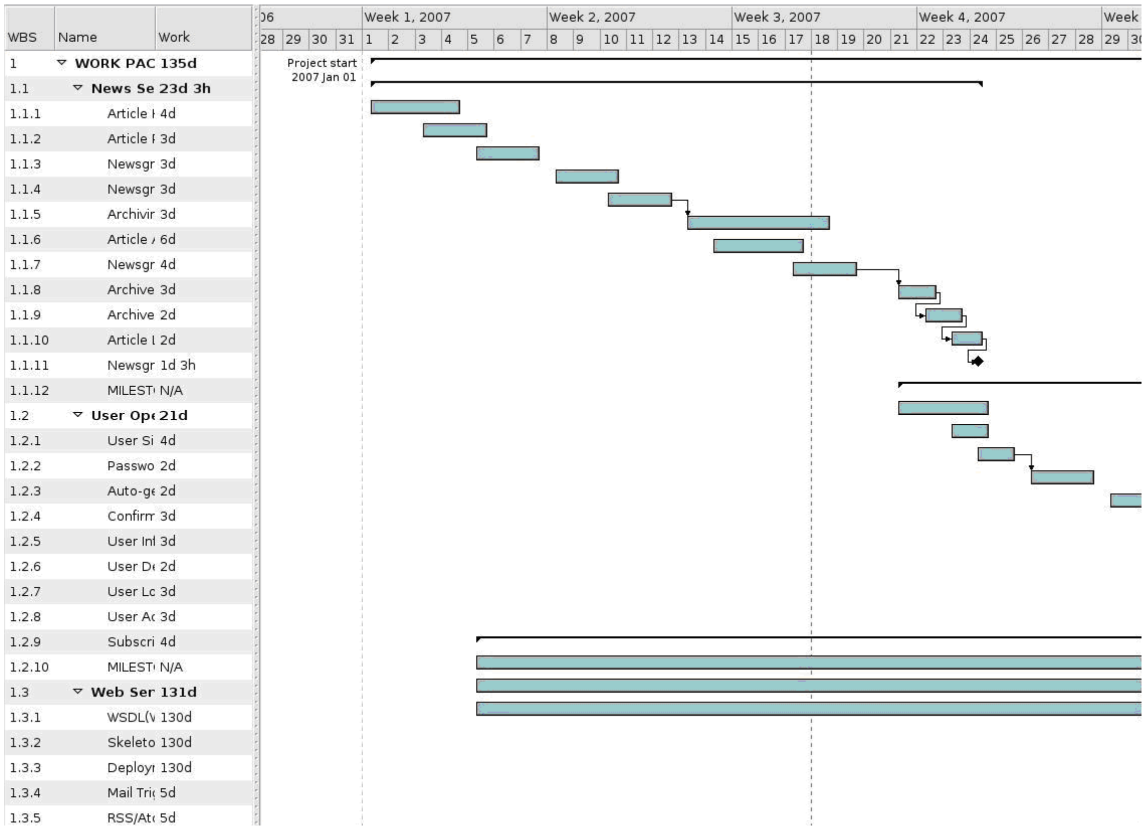
1. http://www.tcpipguide.com
2. http://en.wikipedia.org/wiki/MD5
3. http://www.ietf.org/rfc/rfc0850.txt

4. www.ietf.org/rfc/rfc977.txt

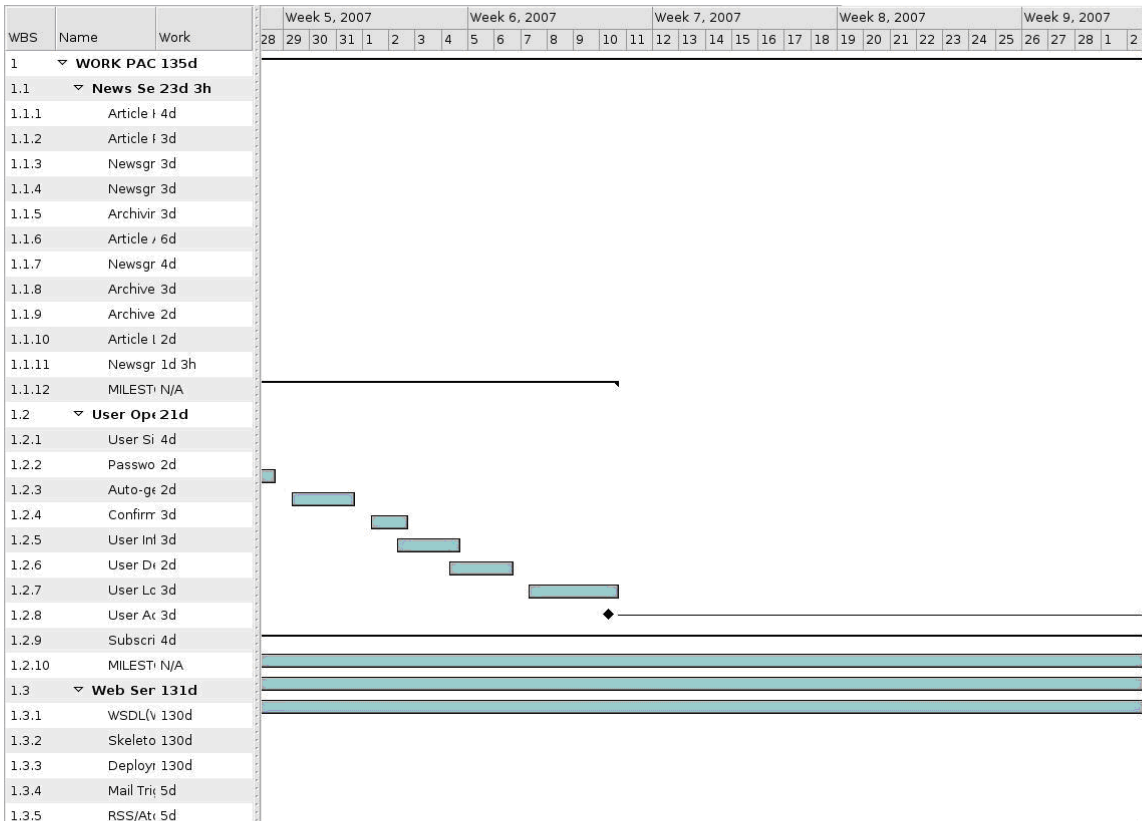
14 APPENDIX



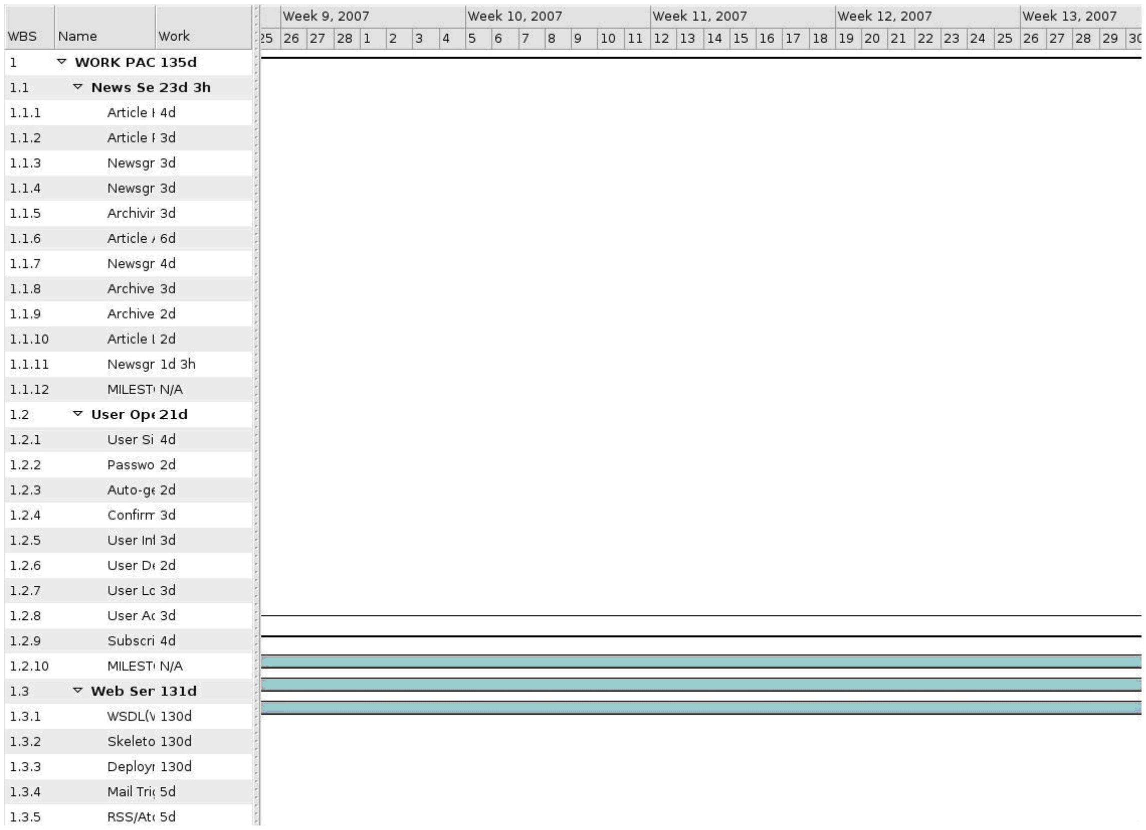
WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
1	**WORK PACKAGE 1 (Core Implementation)**	**Jan 1**	**May 16**	**135d**	**135d**	**26d**	**0**	
1.1	**News Server Operations Implementation**	**Jan 1**	**Jan 24**	**23d 3h**	**23d 3h**	**137d 4h**	**0**	
1.1.1	Article Handlers	Jan 1	Jan 4	4d	4d	157d	0	
1.1.2	Article Parsers/Generators	Jan 3	Jan 5	3d	3d	156d	0	
1.1.3	Newsgroups Handlers	Jan 5	Jan 7	3d	3d	154d	0	
1.1.4	Newsgroup Access Rights Handlers	Jan 8	Jan 10	3d	3d	151d	0	
1.1.5	Archiving Decision	Jan 10	Jan 12	3d	3d	143d	0	
1.1.6	Article Archiving	Jan 13	Jan 18	6d	6d	143d	0	
1.1.7	Newsgroup Archiving	Jan 14	Jan 17	4d	4d	144d	0	
1.1.8	Archive Article Handlers	Jan 17	Jan 19	3d	3d	137d	0	
1.1.9	Archive Newsgroup Handlers	Jan 21	Jan 22	2d	2d	135d 4h	0	
1.1.10	Article Logging Handlers	Jan 22	Jan 23	2d	2d	136d	0	
1.1.11	Newsgroup Logging Handlers	Jan 23	Jan 24	1d 3h	1d 3h	137d 4h	0	
1.1.12	MILESTONE	Jan 24	Jan 24	N/A	N/A	138d	0	
1.2	**User Operations Implementation**	**Jan 21**	**Feb 10**	**21d**	**21d**	**120d**	**0**	
1.2.1	User Sign-up – New Account Creation	Jan 21	Jan 24	4d	4d	137d	0	
1.2.2	Password Creators.	Jan 23	Jan 24	2d	2d	137d	0	
1.2.3	Auto-generated Confirmation Links	Jan 24	Jan 25	2d	2d	133d	0	
1.2.4	Confirmation Handlers	Jan 26	Jan 28	3d	3d	133d	0	
1.2.5	User Info Retrieval, Update	Jan 29	Jan 31	3d	3d	130d	0	
1.2.6	User Deletion	Feb 1	Feb 2	2d	2d	128d	0	
1.2.7	User Logging Handlers	Feb 2	Feb 4	3d	3d	126d	0	
1.2.8	User Access Rights Handlers	Feb 4	Feb 6	3d	3d	124d	0	
1.2.9	Subscription/Unsubscription Manager	Feb 7	Feb 10	4d	4d	120d	0	
1.2.10	MILESTONE	Feb 10	Feb 10	N/A	N/A	121d	0	
1.3	**Web Services Layer Implementation**	**Jan 5**	**May 16**	**131d**	**131d**	**26d**	**0**	
1.3.1	WSDL(Web Service Description Language) Implementation	Jan 5	May 14	130d	130d	27d	0	
1.3.2	Skeleton Implementations	Jan 5	May 14	130d	130d	27d	0	
1.3.3	Deployment of Services	Jan 5	May 14	130d	130d	27d	0	
1.3.4	Mail Triggers	Apr 11	Apr 15	5d	5d	56d	0	



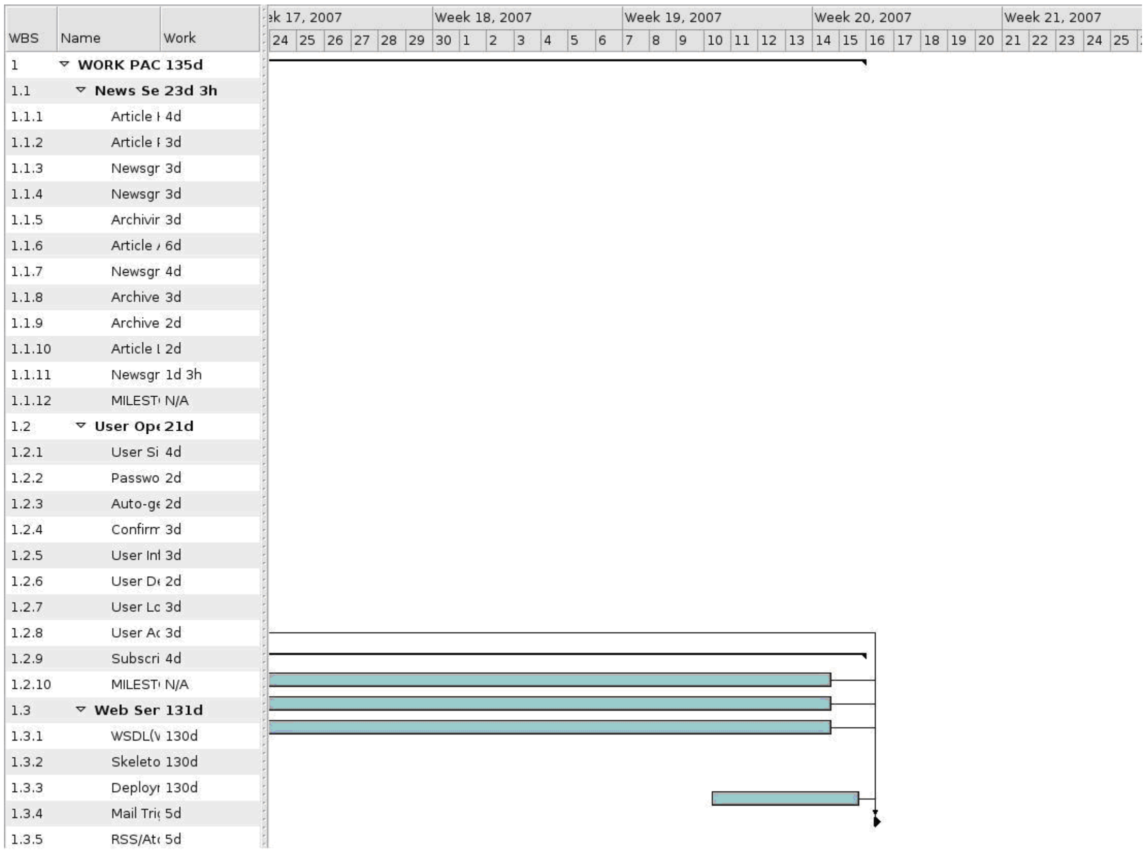
WBS	Name	Work
1	WORK PAC	135d
1.1	News Se	23d 3h
1.1.1	Article I	4d
1.1.2	Article F	3d
1.1.3	Newsgr	3d
1.1.4	Newsgr	3d
1.1.5	Archivir	3d
1.1.6	Article /	6d
1.1.7	Newsgr	4d
1.1.8	Archive	3d
1.1.9	Archive	2d
1.1.10	Article L	2d
1.1.11	Newsgr	1d 3h
1.1.12	MILEST	N/A
1.2	User Op	21d
1.2.1	User Si	4d
1.2.2	Passwo	2d
1.2.3	Auto-ge	2d
1.2.4	Confirm	3d
1.2.5	User Inl	3d
1.2.6	User D	2d
1.2.7	User Lc	3d
1.2.8	User Ac	3d
1.2.9	Subscri	4d
1.2.10	MILEST	N/A
1.3	Web Ser	131d
1.3.1	WSDL(\	130d
1.3.2	Skeleto	130d
1.3.3	Deploy	130d
1.3.4	Mail Tri	5d
1.3.5	RSS/At	5d



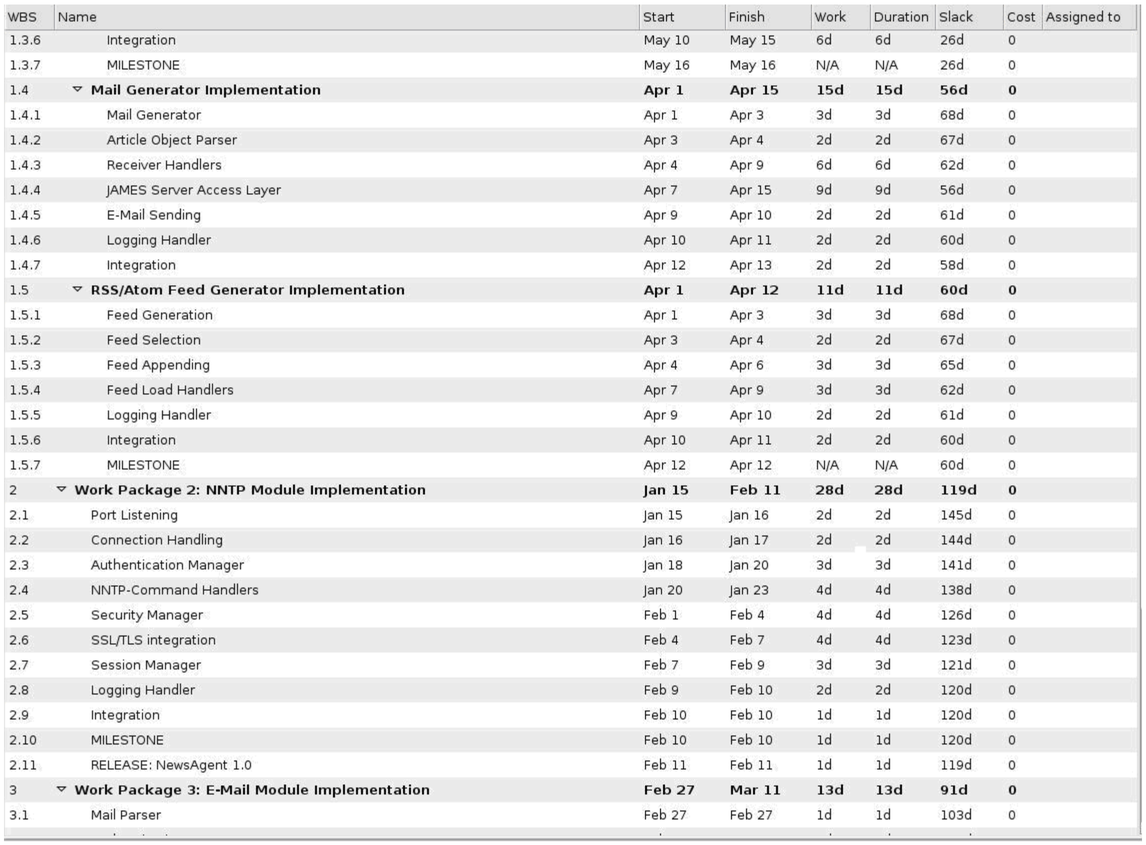
WBS	Name	Work
1	WORK PAC	135d
1.1	News Se	23d 3h
1.1.1	Article I	4d
1.1.2	Article F	3d
1.1.3	Newsgr	3d
1.1.4	Newsgr	3d
1.1.5	Archivir	3d
1.1.6	Article /	6d
1.1.7	Newsgr	4d
1.1.8	Archive	3d
1.1.9	Archive	2d
1.1.10	Article L	2d
1.1.11	Newsgr	1d 3h
1.1.12	MILEST	N/A
1.2	User Op	21d
1.2.1	User Si	4d
1.2.2	Passwo	2d
1.2.3	Auto-ge	2d
1.2.4	Confirm	3d
1.2.5	User Inl	3d
1.2.6	User D	2d
1.2.7	User Lc	3d
1.2.8	User Ac	3d
1.2.9	Subscri	4d
1.2.10	MILEST	N/A
1.3	Web Ser	131d
1.3.1	WSDL(\	130d
1.3.2	Skeleto	130d
1.3.3	Deploy	130d
1.3.4	Mail Tri	5d
1.3.5	RSS/At	5d



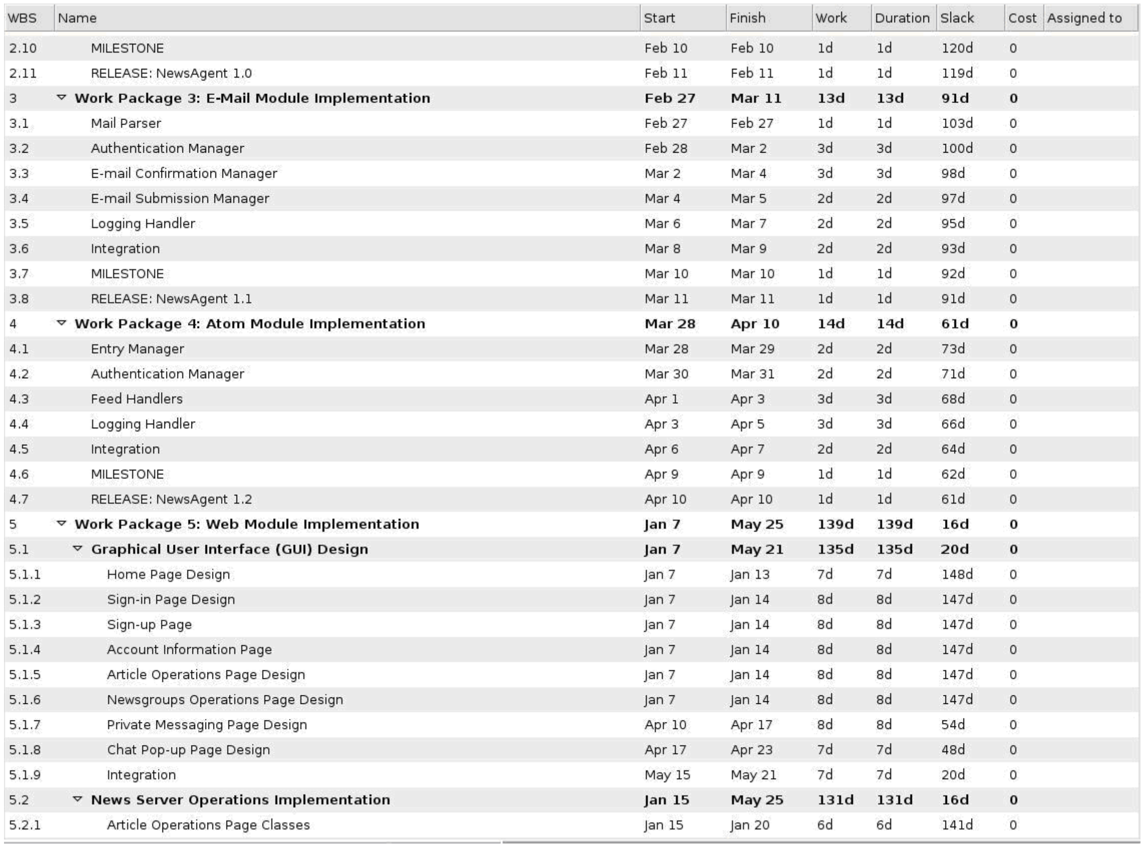
WBS	Name	Work
1	▿ WORK PAC	135d
1.1	▿ News Se	23d 3h
1.1.1	Article I	4d
1.1.2	Article I	3d
1.1.3	Newsgr	3d
1.1.4	Newsgr	3d
1.1.5	Archivir	3d
1.1.6	Article /	6d
1.1.7	Newsgr	4d
1.1.8	Archive	3d
1.1.9	Archive	2d
1.1.10	Article I	2d
1.1.11	Newsgr	1d 3h
1.1.12	MILEST	N/A
1.2	▿ User Op	21d
1.2.1	User Si	4d
1.2.2	Passwo	2d
1.2.3	Auto-ge	2d
1.2.4	Confirm	3d
1.2.5	User Inl	3d
1.2.6	User De	2d
1.2.7	User Lc	3d
1.2.8	User Ac	3d
1.2.9	Subscri	4d
1.2.10	MILEST	N/A
1.3	▿ Web Ser	131d
1.3.1	WSDL(V	130d
1.3.2	Skeleto	130d
1.3.3	Deployı	130d
1.3.4	Mail Tri	5d
1.3.5	RSS/At	5d



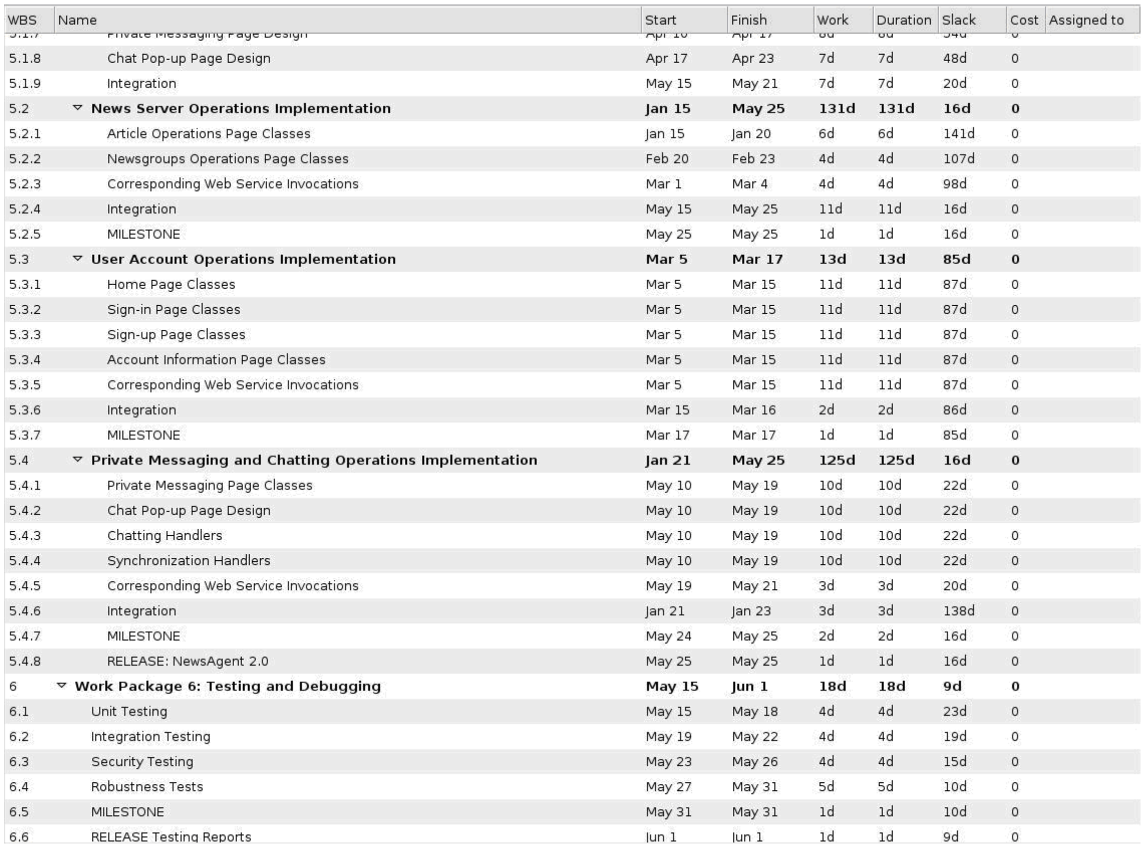
WBS	Name	Work
1	▿ WORK PAC	135d
1.1	▿ News Se	23d 3h
1.1.1	Article I	4d
1.1.2	Article I	3d
1.1.3	Newsgr	3d
1.1.4	Newsgr	3d
1.1.5	Archivir	3d
1.1.6	Article /	6d
1.1.7	Newsgr	4d
1.1.8	Archive	3d
1.1.9	Archive	2d
1.1.10	Article I	2d
1.1.11	Newsgr	1d 3h
1.1.12	MILEST	N/A
1.2	▿ User Op	21d
1.2.1	User Si	4d
1.2.2	Passwo	2d
1.2.3	Auto-ge	2d
1.2.4	Confirm	3d
1.2.5	User Inl	3d
1.2.6	User De	2d
1.2.7	User Lc	3d
1.2.8	User Ac	3d
1.2.9	Subscri	4d
1.2.10	MILEST	N/A
1.3	▿ Web Ser	131d
1.3.1	WSDL(V	130d
1.3.2	Skeleto	130d
1.3.3	Deployı	130d
1.3.4	Mail Tri	5d
1.3.5	RSS/At	5d



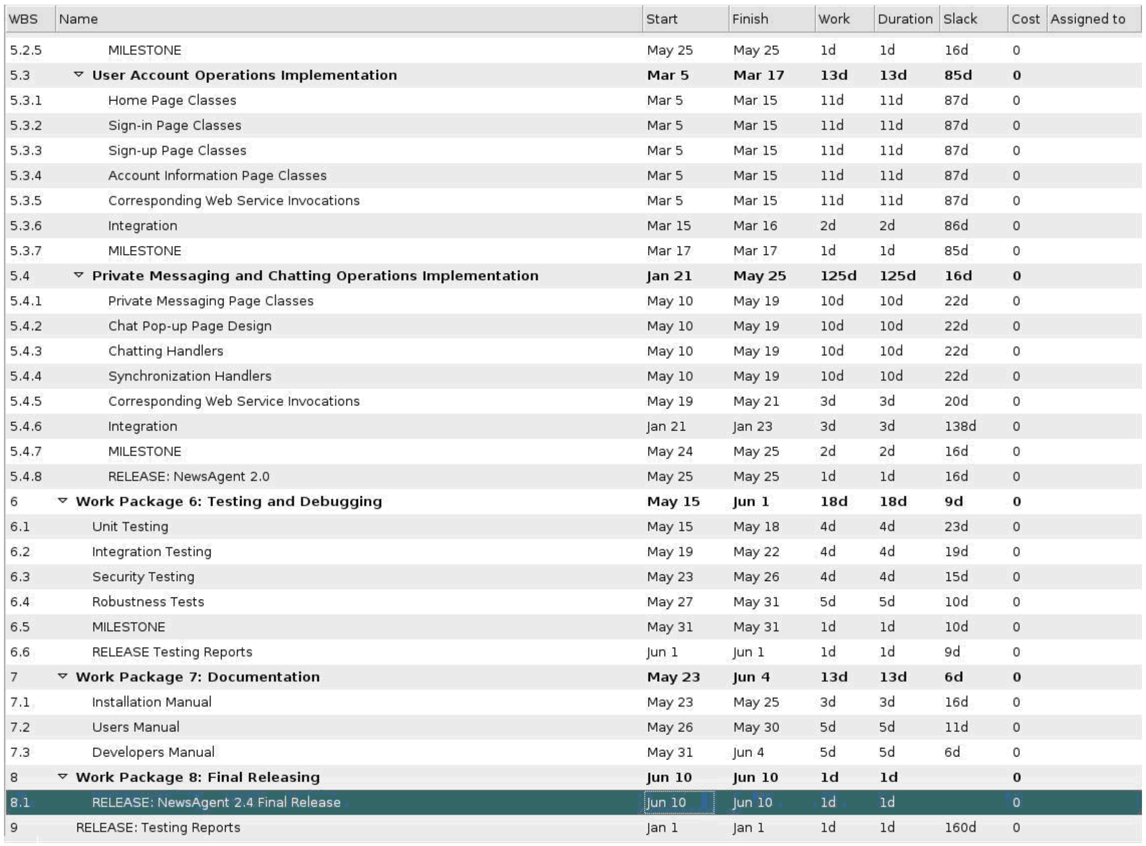
WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
1.3.6	Integration	May 10	May 15	6d	6d	26d	0	
1.3.7	MILESTONE	May 16	May 16	N/A	N/A	26d	0	
1.4	**Mail Generator Implementation**	**Apr 1**	**Apr 15**	**15d**	**15d**	**56d**	**0**	
1.4.1	Mail Generator	Apr 1	Apr 3	3d	3d	68d	0	
1.4.2	Article Object Parser	Apr 3	Apr 4	2d	2d	67d	0	
1.4.3	Receiver Handlers	Apr 4	Apr 9	6d	6d	62d	0	
1.4.4	JAMES Server Access Layer	Apr 7	Apr 15	9d	9d	56d	0	
1.4.5	E-Mail Sending	Apr 9	Apr 10	2d	2d	61d	0	
1.4.6	Logging Handler	Apr 10	Apr 11	2d	2d	60d	0	
1.4.7	Integration	Apr 12	Apr 13	2d	2d	58d	0	
1.5	**RSS/Atom Feed Generator Implementation**	**Apr 1**	**Apr 12**	**11d**	**11d**	**60d**	**0**	
1.5.1	Feed Generation	Apr 1	Apr 3	3d	3d	68d	0	
1.5.2	Feed Selection	Apr 3	Apr 4	2d	2d	67d	0	
1.5.3	Feed Appending	Apr 4	Apr 6	3d	3d	65d	0	
1.5.4	Feed Load Handlers	Apr 7	Apr 9	3d	3d	62d	0	
1.5.5	Logging Handler	Apr 9	Apr 10	2d	2d	61d	0	
1.5.6	Integration	Apr 10	Apr 11	2d	2d	60d	0	
1.5.7	MILESTONE	Apr 12	Apr 12	N/A	N/A	60d	0	
2	**Work Package 2: NNTP Module Implementation**	**Jan 15**	**Feb 11**	**28d**	**28d**	**119d**	**0**	
2.1	Port Listening	Jan 15	Jan 16	2d	2d	145d	0	
2.2	Connection Handling	Jan 16	Jan 17	2d	2d	144d	0	
2.3	Authentication Manager	Jan 18	Jan 20	3d	3d	141d	0	
2.4	NNTP-Command Handlers	Jan 20	Jan 23	4d	4d	138d	0	
2.5	Security Manager	Feb 1	Feb 4	4d	4d	126d	0	
2.6	SSL/TLS integration	Feb 4	Feb 7	4d	4d	123d	0	
2.7	Session Manager	Feb 7	Feb 9	3d	3d	121d	0	
2.8	Logging Handler	Feb 9	Feb 10	2d	2d	120d	0	
2.9	Integration	Feb 10	Feb 10	1d	1d	120d	0	
2.10	MILESTONE	Feb 10	Feb 10	1d	1d	120d	0	
2.11	RELEASE: NewsAgent 1.0	Feb 11	Feb 11	1d	1d	119d	0	
3	**Work Package 3: E-Mail Module Implementation**	**Feb 27**	**Mar 11**	**13d**	**13d**	**91d**	**0**	
3.1	Mail Parser	Feb 27	Feb 27	1d	1d	103d	0	



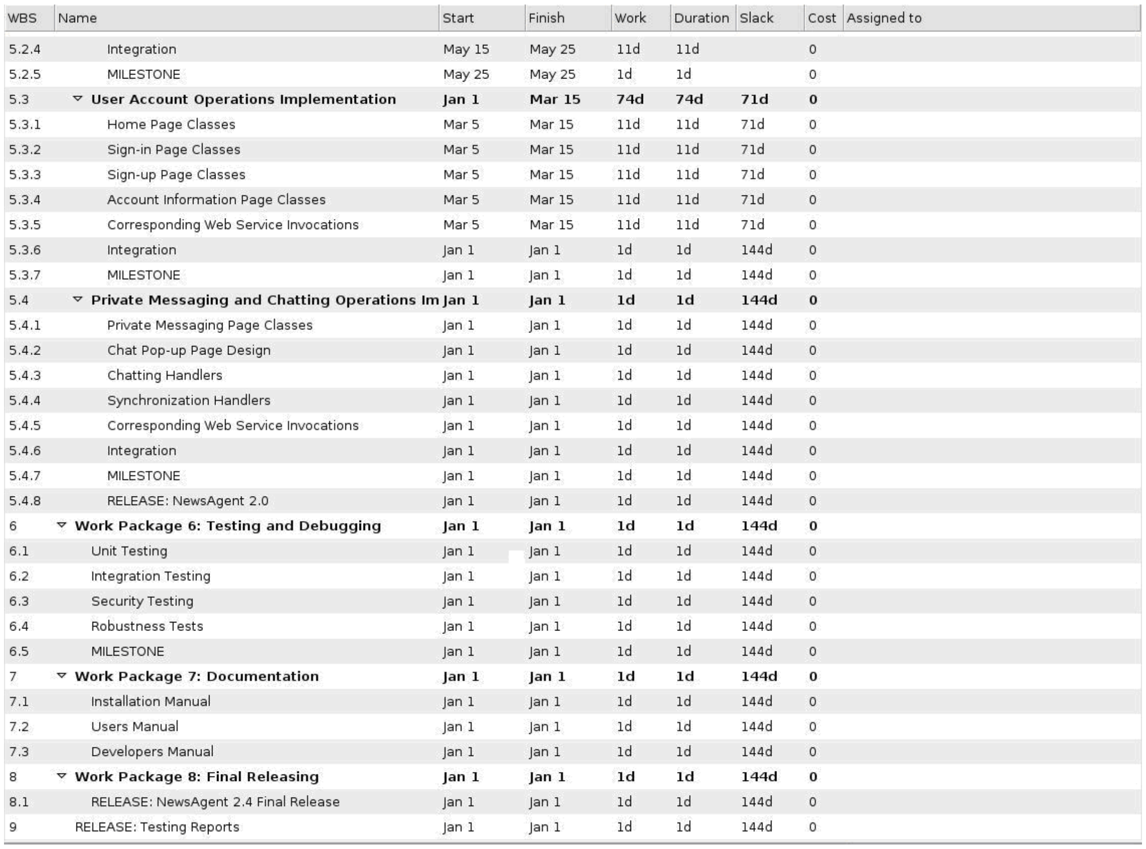
WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
2.10	MILESTONE	Feb 10	Feb 10	1d	1d	120d	0	
2.11	RELEASE: NewsAgent 1.0	Feb 11	Feb 11	1d	1d	119d	0	
3	**Work Package 3: E-Mail Module Implementation**	**Feb 27**	**Mar 11**	**13d**	**13d**	**91d**	**0**	
3.1	Mail Parser	Feb 27	Feb 27	1d	1d	103d	0	
3.2	Authentication Manager	Feb 28	Mar 2	3d	3d	100d	0	
3.3	E-mail Confirmation Manager	Mar 2	Mar 4	3d	3d	98d	0	
3.4	E-mail Submission Manager	Mar 4	Mar 5	2d	2d	97d	0	
3.5	Logging Handler	Mar 6	Mar 7	2d	2d	95d	0	
3.6	Integration	Mar 8	Mar 9	2d	2d	93d	0	
3.7	MILESTONE	Mar 10	Mar 10	1d	1d	92d	0	
3.8	RELEASE: NewsAgent 1.1	Mar 11	Mar 11	1d	1d	91d	0	
4	**Work Package 4: Atom Module Implementation**	**Mar 28**	**Apr 10**	**14d**	**14d**	**61d**	**0**	
4.1	Entry Manager	Mar 28	Mar 29	2d	2d	73d	0	
4.2	Authentication Manager	Mar 30	Mar 31	2d	2d	71d	0	
4.3	Feed Handlers	Apr 1	Apr 3	3d	3d	68d	0	
4.4	Logging Handler	Apr 3	Apr 5	3d	3d	66d	0	
4.5	Integration	Apr 6	Apr 7	2d	2d	64d	0	
4.6	MILESTONE	Apr 9	Apr 9	1d	1d	62d	0	
4.7	RELEASE: NewsAgent 1.2	Apr 10	Apr 10	1d	1d	61d	0	
5	**Work Package 5: Web Module Implementation**	**Jan 7**	**May 25**	**139d**	**139d**	**16d**	**0**	
5.1	**Graphical User Interface (GUI) Design**	**Jan 7**	**May 21**	**135d**	**135d**	**20d**	**0**	
5.1.1	Home Page Design	Jan 7	Jan 13	7d	7d	148d	0	
5.1.2	Sign-in Page Design	Jan 7	Jan 14	8d	8d	147d	0	
5.1.3	Sign-up Page	Jan 7	Jan 14	8d	8d	147d	0	
5.1.4	Account Information Page	Jan 7	Jan 14	8d	8d	147d	0	
5.1.5	Article Operations Page Design	Jan 7	Jan 14	8d	8d	147d	0	
5.1.6	Newsgroups Operations Page Design	Jan 7	Jan 14	8d	8d	147d	0	
5.1.7	Private Messaging Page Design	Apr 10	Apr 17	8d	8d	54d	0	
5.1.8	Chat Pop-up Page Design	Apr 17	Apr 23	7d	7d	48d	0	
5.1.9	Integration	May 15	May 21	7d	7d	20d	0	
5.2	**News Server Operations Implementation**	**Jan 15**	**May 25**	**131d**	**131d**	**16d**	**0**	
5.2.1	Article Operations Page Classes	Jan 15	Jan 20	6d	6d	141d	0	



WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
5.1.7	Private Messaging Page Design	Apr 10	Apr 17	8d	8d	34d	0	
5.1.8	Chat Pop-up Page Design	Apr 17	Apr 23	7d	7d	48d	0	
5.1.9	Integration	May 15	May 21	7d	7d	20d	0	
5.2	▽ **News Server Operations Implementation**	**Jan 15**	**May 25**	**131d**	**131d**	**16d**	**0**	
5.2.1	Article Operations Page Classes	Jan 15	Jan 20	6d	6d	141d	0	
5.2.2	Newsgroups Operations Page Classes	Feb 20	Feb 23	4d	4d	107d	0	
5.2.3	Corresponding Web Service Invocations	Mar 1	Mar 4	4d	4d	98d	0	
5.2.4	Integration	May 15	May 25	11d	11d	16d	0	
5.2.5	MILESTONE	May 25	May 25	1d	1d	16d	0	
5.3	▽ **User Account Operations Implementation**	**Mar 5**	**Mar 17**	**13d**	**13d**	**85d**	**0**	
5.3.1	Home Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.2	Sign-in Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.3	Sign-up Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.4	Account Information Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.5	Corresponding Web Service Invocations	Mar 5	Mar 15	11d	11d	87d	0	
5.3.6	Integration	Mar 15	Mar 16	2d	2d	86d	0	
5.3.7	MILESTONE	Mar 17	Mar 17	1d	1d	85d	0	
5.4	▽ **Private Messaging and Chatting Operations Implementation**	**Jan 21**	**May 25**	**125d**	**125d**	**16d**	**0**	
5.4.1	Private Messaging Page Classes	May 10	May 19	10d	10d	22d	0	
5.4.2	Chat Pop-up Page Design	May 10	May 19	10d	10d	22d	0	
5.4.3	Chatting Handlers	May 10	May 19	10d	10d	22d	0	
5.4.4	Synchronization Handlers	May 10	May 19	10d	10d	22d	0	
5.4.5	Corresponding Web Service Invocations	May 19	May 21	3d	3d	20d	0	
5.4.6	Integration	Jan 21	Jan 23	3d	3d	138d	0	
5.4.7	MILESTONE	May 24	May 25	2d	2d	16d	0	
5.4.8	RELEASE: NewsAgent 2.0	May 25	May 25	1d	1d	16d	0	
6	▽ **Work Package 6: Testing and Debugging**	**May 15**	**Jun 1**	**18d**	**18d**	**9d**	**0**	
6.1	Unit Testing	May 15	May 18	4d	4d	23d	0	
6.2	Integration Testing	May 19	May 22	4d	4d	19d	0	
6.3	Security Testing	May 23	May 26	4d	4d	15d	0	
6.4	Robustness Tests	May 27	May 31	5d	5d	10d	0	
6.5	MILESTONE	May 31	May 31	1d	1d	10d	0	
6.6	RELEASE Testing Reports	Jun 1	Jun 1	1d	1d	9d	0	



WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
5.2.5	MILESTONE	May 25	May 25	1d	1d	16d	0	
5.3	▽ **User Account Operations Implementation**	**Mar 5**	**Mar 17**	**13d**	**13d**	**85d**	**0**	
5.3.1	Home Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.2	Sign-in Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.3	Sign-up Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.4	Account Information Page Classes	Mar 5	Mar 15	11d	11d	87d	0	
5.3.5	Corresponding Web Service Invocations	Mar 5	Mar 15	11d	11d	87d	0	
5.3.6	Integration	Mar 15	Mar 16	2d	2d	86d	0	
5.3.7	MILESTONE	Mar 17	Mar 17	1d	1d	85d	0	
5.4	▽ **Private Messaging and Chatting Operations Implementation**	**Jan 21**	**May 25**	**125d**	**125d**	**16d**	**0**	
5.4.1	Private Messaging Page Classes	May 10	May 19	10d	10d	22d	0	
5.4.2	Chat Pop-up Page Design	May 10	May 19	10d	10d	22d	0	
5.4.3	Chatting Handlers	May 10	May 19	10d	10d	22d	0	
5.4.4	Synchronization Handlers	May 10	May 19	10d	10d	22d	0	
5.4.5	Corresponding Web Service Invocations	May 19	May 21	3d	3d	20d	0	
5.4.6	Integration	Jan 21	Jan 23	3d	3d	138d	0	
5.4.7	MILESTONE	May 24	May 25	2d	2d	16d	0	
5.4.8	RELEASE: NewsAgent 2.0	May 25	May 25	1d	1d	16d	0	
6	▽ **Work Package 6: Testing and Debugging**	**May 15**	**Jun 1**	**18d**	**18d**	**9d**	**0**	
6.1	Unit Testing	May 15	May 18	4d	4d	23d	0	
6.2	Integration Testing	May 19	May 22	4d	4d	19d	0	
6.3	Security Testing	May 23	May 26	4d	4d	15d	0	
6.4	Robustness Tests	May 27	May 31	5d	5d	10d	0	
6.5	MILESTONE	May 31	May 31	1d	1d	10d	0	
6.6	RELEASE Testing Reports	Jun 1	Jun 1	1d	1d	9d	0	
7	▽ **Work Package 7: Documentation**	**May 23**	**Jun 4**	**13d**	**13d**	**6d**	**0**	
7.1	Installation Manual	May 23	May 25	3d	3d	16d	0	
7.2	Users Manual	May 26	May 30	5d	5d	11d	0	
7.3	Developers Manual	May 31	Jun 4	5d	5d	6d	0	
8	▽ **Work Package 8: Final Releasing**	**Jun 10**	**Jun 10**	**1d**	**1d**		**0**	
8.1	RELEASE: NewsAgent 2.4 Final Release	Jun 10	Jun 10	1d	1d		0	
9	RELEASE: Testing Reports	Jan 1	Jan 1	1d	1d	160d	0	



WBS	Name	Start	Finish	Work	Duration	Slack	Cost	Assigned to
5.2.4	Integration	May 15	May 25	11d	11d		0	
5.2.5	MILESTONE	May 25	May 25	1d	1d		0	
5.3	**User Account Operations Implementation**	**Jan 1**	**Mar 15**	**74d**	**74d**	**71d**	**0**	
5.3.1	Home Page Classes	Mar 5	Mar 15	11d	11d	71d	0	
5.3.2	Sign-in Page Classes	Mar 5	Mar 15	11d	11d	71d	0	
5.3.3	Sign-up Page Classes	Mar 5	Mar 15	11d	11d	71d	0	
5.3.4	Account Information Page Classes	Mar 5	Mar 15	11d	11d	71d	0	
5.3.5	Corresponding Web Service Invocations	Mar 5	Mar 15	11d	11d	71d	0	
5.3.6	Integration	Jan 1	Jan 1	1d	1d	144d	0	
5.3.7	MILESTONE	Jan 1	Jan 1	1d	1d	144d	0	
5.4	**Private Messaging and Chatting Operations Im**	**Jan 1**	**Jan 1**	**1d**	**1d**	**144d**	**0**	
5.4.1	Private Messaging Page Classes	Jan 1	Jan 1	1d	1d	144d	0	
5.4.2	Chat Pop-up Page Design	Jan 1	Jan 1	1d	1d	144d	0	
5.4.3	Chatting Handlers	Jan 1	Jan 1	1d	1d	144d	0	
5.4.4	Synchronization Handlers	Jan 1	Jan 1	1d	1d	144d	0	
5.4.5	Corresponding Web Service Invocations	Jan 1	Jan 1	1d	1d	144d	0	
5.4.6	Integration	Jan 1	Jan 1	1d	1d	144d	0	
5.4.7	MILESTONE	Jan 1	Jan 1	1d	1d	144d	0	
5.4.8	RELEASE: NewsAgent 2.0	Jan 1	Jan 1	1d	1d	144d	0	
6	**Work Package 6: Testing and Debugging**	**Jan 1**	**Jan 1**	**1d**	**1d**	**144d**	**0**	
6.1	Unit Testing	Jan 1	Jan 1	1d	1d	144d	0	
6.2	Integration Testing	Jan 1	Jan 1	1d	1d	144d	0	
6.3	Security Testing	Jan 1	Jan 1	1d	1d	144d	0	
6.4	Robustness Tests	Jan 1	Jan 1	1d	1d	144d	0	
6.5	MILESTONE	Jan 1	Jan 1	1d	1d	144d	0	
7	**Work Package 7: Documentation**	**Jan 1**	**Jan 1**	**1d**	**1d**	**144d**	**0**	
7.1	Installation Manual	Jan 1	Jan 1	1d	1d	144d	0	
7.2	Users Manual	Jan 1	Jan 1	1d	1d	144d	0	
7.3	Developers Manual	Jan 1	Jan 1	1d	1d	144d	0	
8	**Work Package 8: Final Releasing**	**Jan 1**	**Jan 1**	**1d**	**1d**	**144d**	**0**	
8.1	RELEASE: NewsAgent 2.4 Final Release	Jan 1	Jan 1	1d	1d	144d	0	
9	RELEASE: Testing Reports	Jan 1	Jan 1	1d	1d	144d	0	