MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

ServerTheon Project Detailed Design Report

BY

TheonTech

TABLE OF CONTENTS

1.0 Project Scope and Description

1.1 Problem Definition

Usenet is a set of protocols for generating, storing and retrieving news articles and for exchanging these articles among a readership which is potentially widely distributed. Usenet is one of the oldest computer network communications systems (established in 1980) still in widespread use.

Nowadays, the web forums and RSS newsfeeds are more commonly used for news broadcasting and discussion sessions. For the Unified News Exchange Server with NNTP, Mail, Web and RSS Project, we are expected to implement the followings:

- **The Message Exchange Core:** A basic threaded and secure message exchange service
- **The Extension Modules:** Modules providing e-mail lists, NNTP, WWW and RSS access methods.

1.2 Project Scope

ServerTheon is mainly responsible for distributing messages obtained via any channel, using the other channels. Whenever a subscriber posts a message to one of the extension modules, the following tasks will be accomplished:

- The message will also be posted as mail to the subscriber and to the related mail group.
- The message will also be posted to the related newsgroup.
- The message will also be fed to the RSS clients on web.
- The message will also be seen on the web forum.

ServerTheon Core Service communicates with Apache Web Server and James Mail and NNTP Server while performing the message distribution. It also communicates with the access modules, namely web module, mail module and news module; in a service-oriented manner.

2.0 Project Schedule

2.1 Work Breakdown Structure (Implementation)

1.0 GUI Implementations

1.1 Authenticaton Implementation
1.2 User Profiler Implementation
1.3 Service Selection Implementation
1.4 Password Operations Implementation
1.5 NewsTheon Viewer Implementation
1.6 NewsTheon Sender Implementation
1.7 ForumTheon Viewer Implementation
1.8 ForumTheon Sender Implementation
1.9 Mail Theon Viewer Implementation
1.10 Mail Theon Sender Implementation
1.11 RSS Comment Viewer/Sender Implementation
1.12 Message Searcher Implementation
1.13 Administrator Panel Implementation

4.1 Unit Testing

4.2 Integration Testing

4.3 Debugging & Fixing

5.0 Documentation

5.1 User-Manuel Preparation

5.2 Installation Manuel Preparation

6.0 Packaging and Releasing

6.1 Packaging Whole Project

2.2 Gantt Chart

The Gantt Chart can be seen under Appendix-A.

3.0 System Modules

The ServerTheon Modules can be studied in two parts. The first one is server related modules. The server related modules of the ServerTheon Project that we are going to explain explicitly are **ServerTheon Core Module, Web Module, News Module and Mail Module.** The second one is client related modules. The client related modules are **MailTheon Module, NewsTheon Module, RSS Comment Module, ForumTheon Module, Admin Module.**

3.1 ServerTheon Core Module

Servertheon core service exists in the core of Servertheon. Servertheon core service will be implemented as web services. It is mainly responsible for distributing messages obtained via any channel, using the other channels. ServerTheon Core Service communicates with James Mail and NNTP Server while performing this distribution. It also communicates with the access modules, namely web module, mail module and news module in a service-oriented manner. ServerTheon Core Service will organize interactions among the access modules and between them and the servers. It is also responsible for providing database connectivity of the modules and the servers. The classes in the module are as follows:

- Core Service Class
- Service Message Class
- Service Response Class
- Database Connector Class

The internals of these classes will be explained in the class descriptions part.

3.2 Web Module

Web Module provides the interaction of ServerTheon with the clients using HTTP access method. These clients include forum clients who are using ForumTheon, RSS clients who are using any RSS aggregator application and administrative users who will configure ServerTheon using Administration Web Interface. Web Module obtains HTTP requests, processes them, invokes CoreService whenever required and finally returns HTTP response to the client. It also helps Mail Module and News Module in interacting with the clients using NewsTheon and MailTheon interfaces. The classes in the module are as follows:

- RSS Handler Class
- Forum Handler Class
- Admin Handler Class

The internals of these classes will be explained in the class descriptions part.

3.3 Mail Module

The responsibility of Mail Module is to control the access of clients to ServerTheon via SMTP and POP3 protocols. A client using any e-mail application will interact with Mail Module. Mail Module is the main module that a client using e-mail access through MailTheon web interface will communicate with. However, since MailTheon is served using HTTP, Mail Module will also communicate with Web Module while interacting with a client that uses MailTheon, and this communication will be organized by ServerTheon Core Service. The classes in the module are as follows:

- Authentication Class
- Message Converter Class
- Listener Class
- Database Connector Class

The internals of these classes will be explained in the class descriptions part.

3.4 News Module

News module enables the clients to access ServerTheon via NNTP. This access method may occur in two different ways: Firstly, a client can use any news reader application such as Thunderbird. And secondly, a client may prefer to read newsgroup messages by using NewsTheon web interface.
If the second way is preferred, then as it is in the operation of Mail Module, News Module will interact with Web Module with

the help of ServerTheon Core Service. The classes in the module are as follows:

- News Class
- News Thread Class
- Authentication Class
- Listener Class

The internals of these classes will be explained in the class descriptions part.

4.0 Data Design

2.1 Database Description

Data Design is the first step of the design procedure. Therefore, every other step is dependent to this step. To design our database, we create data object then we organize some relationships between these objects. We have defined our relationships by constructing ER diagrams in the requirement analysis phase. In the following sections we will look at data objects, their relationships, create tables and the data dictionary for data objects.

2.1.1 Data Objects

a. Client

The Client entity saves all information related to a user. Client has to be authenticated. Some of the attributes are optional, i.e. can be ‘null’. Entity contains a primary key which is Client ID. Client ID is auto-incremented with seed one. Attributes are:

- ClientID

- ClientName
- ClientPassword
- ClientSecretQuestion
- ClientSecretAnswer
- Name
- Surname
- Gender
- DateofBirth
- CountryID
- StateID
- ZipCode
- Telephone
- E-mail

b. Admin

The Admin entity saves all information related to an admin. Admin have to be authenticated. All attributes are mandatory, i.e. can not be ‘null’. Entity contains a primary key which is AdminID. AdminID is auto-incremented with seed one. Attributes are:

- <u>AdminID</u>
- AdminName
- AdminPassword
- Name
- Surname
- Gender
- DateofBirth
- CountryID
- StateID

- ZipCode
- Telephone
- E-mail

c. Message

Message entity contains the information about messages. All attributes of the entity must be entered. However, ParentMessageID of first message will be null. Primary key is MessageID. MessageID is randomly generated by the system because of security reasons such as SQL injection. MessageSize is size of the message Attributes are:

- MessageID
- MessageText
- MessageSubject
- MessageSize
- MessageIsRead
- DateTime
- ParentMessageID

d. Attachment

Attachment entity contains the information about attachments. All entries must be entered. Primary key is AttachmentID. AttachmentID is auto-incremented with seed one. Attributes are:

- AttachmentID
- FilePath
- FileName

e. Group

Group entity saves all information about groups such as computer is a group and hardware and software is its subgroups.

All entries must be entered. Primary key is GroupID. GroupID is auto-incremented with seed one. Attributes are:

- GroupID
- GroupName
- GroupDescription
- GroupParentID

f. Service

Service entity saves all information about services. Services are news, mail, forum and RSS. All entries must be entered. Primary key is ServiceID. ServiceID is auto-incremented with seed one. Attributes are:

- ServiceID
- ServiceName

2.1.2 Relationships

a. Send

Each client may send zero or many messages. Every message must be sent by exactly one client. Send relation attributes are:

- MessageID
- ClientID

b. Select0

Each client may select one or more services. Each service may be selected by zero or many client.

- ServiceID
- ClientID

c.Select1

Each client may select one or more groups. Each group may be selected by zero or many client.

- GroupID
- ClientID

d. Has

Every message has exactly one group. Every group may be owned by zero or many messages.

- MessageID
- GroupID

e. Has1

Each message has zero or many attachments. Every attachment must be owned by exactly one message.

- MessageID
- AttachmentID

f. Edit0

Every message may be edited by zero or one client. Each client may edit zero or many message.

- MessageID
- ClientID

g. Delete0

Every message may be deleted by zero or one client. Each client may delete zero or one message.

- MessageID
- ClientID

h. AdSendMessage

Each admin may send zero or many messages. Each message is sent by only one admin

- <u>AdminID</u>
- <u>MessageID</u>

i. AdEditMessage

Each admin may edit zero or many messages. Each message may be edited by zero or many admin.

- <u>AdminID</u>
- <u>MessageID</u>
- AdEditDate

j. AdDeleteMessage

Each admin may delete zero or many messages. Each message may be deleted zero or one admin.

- <u>AdminID</u>
- <u>MessageID</u>
- AdDeleteDate

k. AdEditClient

Each admin may edit zero or many client. Each client may be edited by zero or many admin.

- <u>AdminID</u>
- <u>ClientID</u>
- AdEditDate

l. AdDeleteClient

Each admin may delete zero or many client. Each client may be deleted by zero or one admin.

- <u>AdminID</u>
- <u>ClientID</u>
- AdDeleteDate

m. AddGroup

Each admin must add one or more group. Each group must be added by exactly one admin.

- AdminID
- <u>GroupID</u>
- GroupCreationDate

n. EditGroup

Each admin may edit zero or many group. Each group may be edited by zero or many admin.

- <u>AdminID</u>
- <u>GroupID</u>
- EditDate

o. DeleteGroup

Each admin may delete zero or many group. Each group may be deleted by zero or one admin.

- <u>AdminID</u>
- <u>GroupID</u>
- DeleteDate

2.2 Database Design

As a database management system we use MYSQL. We stored our data objects in tables. We created our tables using following codes.

2.2.1 Database Table Specifications

NOTE: PK = PRIMARY KEY

FK = FOREIGN KEY

a. Client

Data	**Type- Size**	**Format**
ClientID	int, auto-incr, not null , **(PK)**	Number
ClientName	varchar(30),not null	Text
ClientPassword	varchar(10),not null	Hidden Text
Name	varchar(30),not-null	Text
Surname	varchar(30),not-null	Text
Gender	varchar(10),not-null	Text
YearofBirth	Date	Date
CountryID	Int **(FK)**	Text
StateID	Int **(FK)**	Text
ZipCode	Int	Number
Telephone	varchar(15)	Text
E-mail	varchar(40)	Text

b.Admin

Data	**Type- Size**	**Format**
AdminID	int, auto-incr, not null , **(PK)**	Number
AdminName	varchar(30),not null	Text
AdminPassword	varchar(10),not null	Hidden Text
Name	varchar(30),not-	Text

	null	
Surname	varchar(30),not-null	Text
Gender	varchar(10),not-null	Text
YearofBirth	Date	Date
CountryID	Int **(FK)**	Text
StateID	Int **(FK)**	Text
ZipCode	Int	Number
Telephone	varchar(15)	Text
E-mail	varchar(40)	Text

c. Message

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL auto_incr , **(PK)**	Text
MessagText	Text	Text
MessageSubject	varchar(100)	Text
MessageSize	int	Number
MessageIsRead”	Bit	Number
SendDate	DateTime	DateTime
ParentMessageID	Int	Text

d. Attachment

Data	**Type-Size**	**Format**
AttachmentID	Int NOT NULL auto_incr , **(PK)**	Number
FilePath	varchar(100)	Text
FileName	varchar(50)	Text

e. Group

Data	**Type-Size**	**Format**
GroupID	Int NOT NULL auto_incr , **(PK)**	Number
GroupName	varchar(50)	Text
GroupDesc	text	Text
GroupParentID	int	Number

f. Service

Data	**Type-Size**	**Format**
ServiceID	Int NOT NULL auto_incr , **(PK)**	Number
ServiceName	varchar(20)	Text

g. Send

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
ClientID	Int NOT NULL **(FK)**	Number

h. Select0

Data	**Type-Size**	**Format**
ServiceID	Int NOT NULL **(FK)**	Number
ClientID	Int NOT NULL **(FK)**	Number

i. Select1

Data	**Type-Size**	**Format**
GroupID	Int NOT NULL **(FK)**	Number
ClientID	Int NOT NULL **(FK)**	Number

j. Has

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
GroupID	Int NOT NULL **(FK)**	Number

k. Has1

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
AttachmentID	Int NOT NULL **(FK)**	Number

l. Edit0

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
ClientID	Int NOT NULL **(FK)**	Number
EditDate	DateTime	DateTime

m. Delete0

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
ClientID	Int NOT NULL **(FK)**	Number
DeleteDate	DateTime	DateTime

n. AdSendMessage

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number

o. AdEditMessage

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
AdEditDate	DateTime	DateTime

p. AdDeleteMessage

Data	**Type-Size**	**Format**
MessageID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
AdDeleteDate	DateTime	DateTime

r. AdEditClient

Data	**Type-Size**	**Format**
ClientID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
AdEditDate	DateTime	DateTime

s. AdDeleteClient

Data	**Type-Size**	**Format**
ClientID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
AdDeleteDate	DateTime	DateTime

t. AddGroup

Data	**Type-Size**	**Format**
GroupID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
GroupCreationDate	DateTime	DateTime

u. EditGroup

Data	**Type-Size**	**Format**
GroupID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
EditDate	DateTime	DateTime

v. DeleteGroup

Data	**Type-Size**	**Format**
GroupID	Int NOT NULL **(FK)**	Number
AdminID	Int NOT NULL **(FK)**	Number
DeleteDate	DateTime	DateTime

y. Country

Data	**Type-Size**	**Format**
CountryID	Int NOT NULL **(PK)**	Number
CountryName	Varchar(30) NOT NULL	Text

z. State

Data	**Type-Size**	**Format**
StateID	Int NOT NULL **(PK)**	Number
StateName	Varchar(30) NOT NULL	Text
CountryID	Int NOT NULL **(FK)**	Number

2.2.2 Database Table SQL’s

```
CREATE TABLE Country(
CountryID INTEGER NOT NULL AUTO_INCREMENT,
CountryName VARCHAR(30) NOT NULL,
PRIMARY KEY(CountryID)
);

CREATE TABLE State(
StateID INTEGER NOT NULL AUTO_INCREMENT,
StateName VARCHAR(30) NOT NULL,
CountryID INTEGER NOT NULL,
PRIMARY KEY (StateID),
FOREIGN KEY (CountryID) REFERENCES Country (CountryID)
);

CREATE TABLE Client(
     ClientID INTEGER NOT NULL AUTO_INCREMENT,
     ClientName VARCHAR(30) NOT NULL,
     ClientPassword VARCHAR(15) NOT NULL,
     Name VARCHAR(30) NOT NULL,
     Surname VARCHAR(30) NOT NULL,
     Gender VARCHAR(10) NOT NULL,
     ClientSecretQuestion VARCHAR(50) NOT NULL,
     ClientSecretAnswer TEXT NOT NULL,
     YearofBirth DATE NOT NULL,
     CountryID INTEGER NOT NULL,
     StateID INTEGER NOT NULL,
     ZipCode INTEGER NOT NULL,
     Telephone VARCHAR(30),
     EMail VARCHAR(40),
     PRIMARY KEY(ClientID),
     FOREIGN KEY (CountryID) REFERENCES Country(CountryID),
     FOREIGN KEY (StateID) REFERENCES State(StateID)
```

```
);


CREATE TABLE Admin(
     AdminID INTEGER NOT NULL AUTO_INCREMENT,
     AdminName VARCHAR(30) NOT NULL,
     AdminPassword VARCHAR(15) NOT NULL,
     Name VARCHAR(30) NOT NULL,
     Surname VARCHAR(30) NOT NULL,
     Gender VARCHAR(10) NOT NULL,
     AdminSecretQuestion VARCHAR(50) NOT NULL,
     AdminSecretAnswer TEXT NOT NULL,
     YearofBirth DATE NOT NULL,
     CountryID INTEGER NOT NULL,
     StateID INTEGER NOT NULL,
     ZipCode INTEGER NOT NULL,
     Telephone VARCHAR(30),
     EMail VARCHAR(40),
     PRIMARY KEY(AdminID),
     FOREIGN KEY (CountryID) REFERENCES Country(CountryID),
     FOREIGN KEY (StateID) REFERENCES State(StateID)

);

CREATE TABLE Message (
     MessageID INTEGER NOT NULL AUTO_INCREMENT,
     MessageText TEXT,
     MessageSubject VARCHAR(100) DEFAULT '[NONE]',
     MessageSize INTEGER NOT NULL,
     MessageIsRead BIT DEFAULT 0,
     SendDate DATETIME NOT NULL,
     ParentMessageID INTEGER DEFAULT NULL,
     PRIMARY KEY (MessageID)
);


CREATE TABLE Attachment(
     AttachmentID INTEGER NOT NULL AUTO_INCREMENT,
     FilePath VARCHAR(300),
     FileName VARCHAR(100),
     PRIMARY KEY(AttachmentID)
);

CREATE TABLE Groups(
     GroupID INTEGER NOT NULL AUTO_INCREMENT,
     GroupName VARCHAR(50),
     GroupDesc TEXT,
     GroupParentID INTEGER DEFAULT NULL,
     PRIMARY KEY(GroupID)
);

CREATE TABLE Service(
     ServiceID INTEGER NOT NULL AUTO_INCREMENT,
     ServiceName VARCHAR(20),
     PRIMARY KEY(ServiceID)
);

CREATE TABLE Send (
     MessageID INTEGER NOT NULL,
     ClientID INTEGER NOT NULL,
     UNIQUE(MessageID),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
```

```
);

CREATE TABLE Select0 (
     ServiceID INTEGER NOT NULL,
     ClientID INTEGER NOT NULL,
     UNIQUE (ServiceID,ClientID),
     FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID),
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE Select1 (
     GroupID INTEGER NOT NULL,
     ClientID INTEGER NOT NULL,
     UNIQUE (GroupID ,ClientID),
     FOREIGN KEY (GroupID) REFERENCES Groups(GroupID ),
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE Has (
     MessageID INTEGER NOT NULL,
     GroupID INTEGER NOT NULL,
     UNIQUE (MessageID,GroupID),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (GroupID) REFERENCES Groups(GroupID)
);


CREATE TABLE Has1 (
     MessageID INTEGER NOT NULL,
     AttachmentID INTEGER NOT NULL,
     UNIQUE (AttachmentID),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (AttachmentID) REFERENCES
     Attachment(AttachmentID)
);



CREATE TABLE Edit0 (
     MessageID INTEGER NOT NULL,
     ClientID  INTEGER NOT NULL,
     EditDate DATETIME NOT NULL,
     UNIQUE (MessageID,EditDate),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);

CREATE TABLE Delete0 (
     MessageID INTEGER NOT NULL,
     ClientID  INTEGER NOT NULL,
     DeleteDate DATETIME NOT NULL,
     UNIQUE (MessageID,ClientID,DeleteDate),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID),
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
);


CREATE TABLE AdSendMessage (
     MessageID INTEGER NOT NULL,
     AdminID INTEGER NOT NULL,
     UNIQUE(MessageID),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);

CREATE TABLE AdEditMessage (
     MessageID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     AdEditDate DATETIME NOT NULL,
     UNIQUE (MessageID,AdminID,AdEditDate),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
     ON DELETE CASCADE,
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);

CREATE TABLE AdDeleteMessage (
     MessageID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     AdDeleteDate DATETIME NOT NULL,
     UNIQUE (MessageID,AdDeleteDate),
     FOREIGN KEY (MessageID) REFERENCES Message(MessageID),
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);


CREATE TABLE AdEditClient (
     ClientID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     AdEditDate DATETIME NOT NULL,
     UNIQUE (ClientID,AdminID,AdEditDate),
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID)
     ON DELETE CASCADE,
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);

CREATE TABLE AdDeleteClient (
     ClientID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     AdDeleteDate DATETIME NOT NULL,
     UNIQUE (ClientID),
     FOREIGN KEY (ClientID) REFERENCES Client(ClientID),
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);

CREATE TABLE AddGroup (
     GroupID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     GroupCreationDate DATETIME NOT NULL,
     UNIQUE (GroupID),
     FOREIGN KEY (GroupID) REFERENCES Groups(GroupID)
     ON DELETE CASCADE,
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);

CREATE TABLE EditGroup (
     GroupID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     EditDate DATETIME NOT NULL,
     UNIQUE (GroupID,EditDate),
     FOREIGN KEY (GroupID) REFERENCES Groups(GroupID)
     ON DELETE CASCADE,
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID)
);
```

```
CREATE TABLE DeleteGroup (
     GroupID INTEGER NOT NULL,
     AdminID  INTEGER NOT NULL,
     DeleteDate DATETIME NOT NULL,
     UNIQUE (GroupID),
     FOREIGN KEY (GroupID) REFERENCES Groups(GroupID),
     FOREIGN KEY (AdminID) REFERENCES Admin(AdminID));
```

5.0 System Design

5.1 Use-Case

5.1.1 Use-Case Scenarios

The scenarios often called use-cases provide a description of how the system will be used. Once actors have been identified, use-cases can be developed. The use-case describes the manner in which an actor interacts with the system.

To create our use-cases; we have to identify the actors that use the ServerTheon. There are 6 actors in our system. These are Visitor, Newsgroup Client, Web Client, Mail Client, RSS Client and Administrator:

Actor-1: Visitor

Usage Scenario -1:

1. The visitor accesses our system via internet.
2. The visitor makes registration by using the sign-up feature of our system.
3. During registration, the visitor selects services which he/she wants.
4. After registration, the visitor logins by using the sign-in feature of our system.

Actor-2: Newsgroup Client

Usage Scenario –2:

1. The Newsgroup client subscribes to a group or groups which she/he wants to follow.
2. The Newsgroup client unsubscribes from a group or groups which she/he does not want to follow anymore.
3. The Newsgroup client reads messages in groups which he/she subscribed.
4. The Newsgroup client creates a new thread in a group which he/she subscribed.
5. The Newsgroup client sends a follow-up to a thread.
6. While replying a message, client quotes.
7. The Newsgroup client deletes his/her message.
8. The Newsgroup client logs out of our system.

Actor-3: Web Client

Usage Scenario –3:

1. The WebForum client reads a message in any group.
2. The WebForum client creates a new topic in any group.
3. The WebForum client replies to a message.
4. While replying a message, client quotes.
5. The WebForum client deletes or edits his/her message that he/she does not want to be read.
6. The WebForum client makes a search to find a keyword that he/she is looking for.
7. The WebForum client logs out of our system.

Actor-4: Mail Client

Usage Scenario –4:

1. The Mail client subscribes to a group or groups which she/he wants to follow.
2. The Mail client unsubscribes from a group or groups which she/he does not want to follow anymore.

3. The Mail client gets mail in groups which he/she subscribed.
4. The Mail client sends a mail to a group which he/she subscribed.
5. The Mail client replies to a mail.
6. While replying a message, client quotes.
7. The Mail client deletes his/her mail.
8. The Mail client logs out of our system.

Actor-5: RSS Client

Usage Scenario -5:

1. The RSS client can reach and read RSS comments.
2. The RSS client can write/send a comment.

Actor-6: Administrator (Admin)

Usage Scenario -6:

1. Admin reads message in any group.
2. Admin sends message to any group when he/she wants to make an announcement or reply a message.
3. Admin deletes a message when he/she thinks that the message is inappropriate.
4. Admin edits the information of a client.
5. Admin deletes the account of a client.
6. Admin logs out of our system.

5.1.2 User Profiles

5.1.2.1 Newsgroup Client

- can subscribe to a group
- can unsubscribe from a group
- can read messages in groups
- can create a new thread in a group
- can send a follow-up to a thread
- can quote

- can delete his/her message
- can log out of the system

5.1.2.2 Web Client

- can read a message in any group
- can create a new topic in any group
- can reply to a message
- can quote
- can delete or edit his/her message
- can make a search to find a keyword
- can log out of the system

5.1.2.3 Mail Client

- can subscribe to a group
- can unsubscribe from a group
- can get mail from his/her groups
- can send a mail to one of his/her groups
- can reply to a mail
- can quote
- can delete his/her mail
- can log out of our system

5.1.2.4 RSS Client

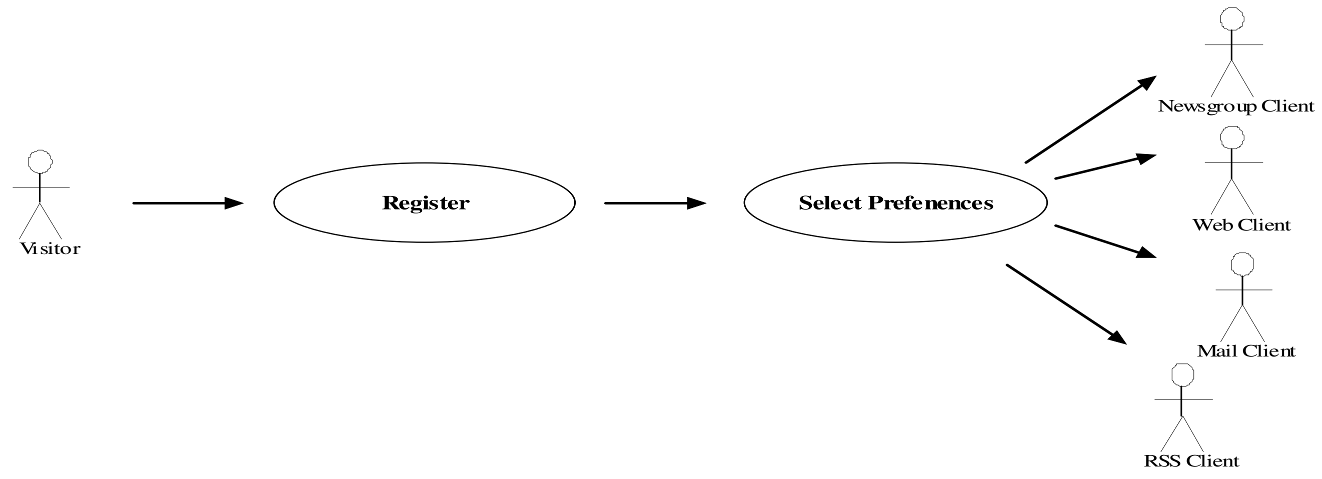
- can reach and read RSS comments.
- can write/send a comment

5.1.2.5 Administrator

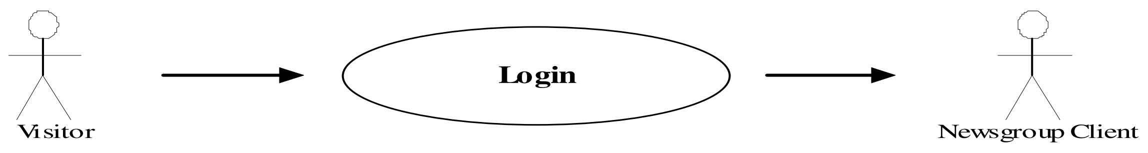
- can read message in any group

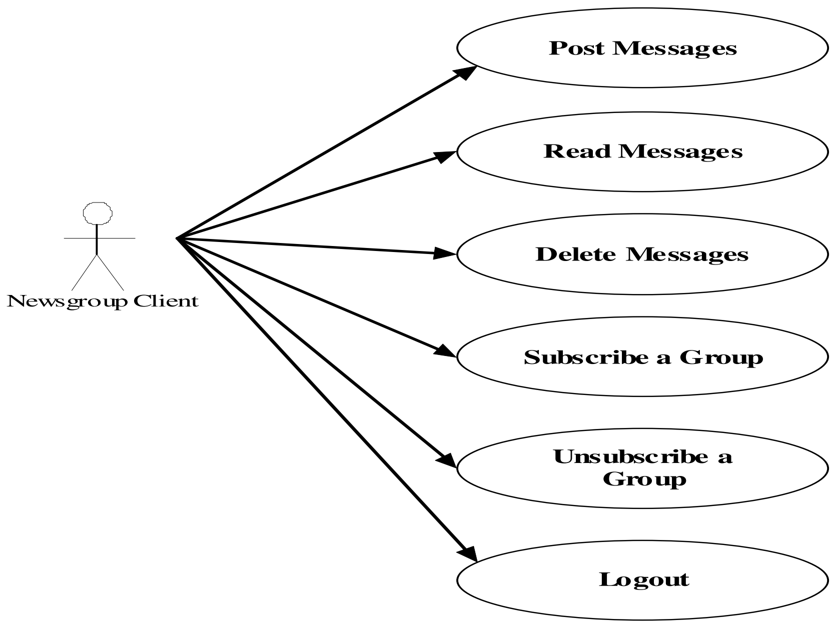
- can send message to any group
- can delete inappropriate messages
- can edit the client information
- can delete the client account
- can log out of our system

5.1.3 Use-Case Diagrams

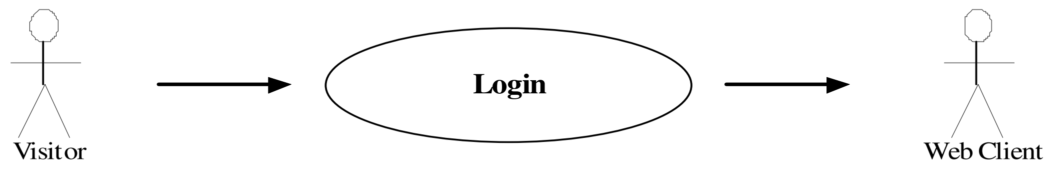


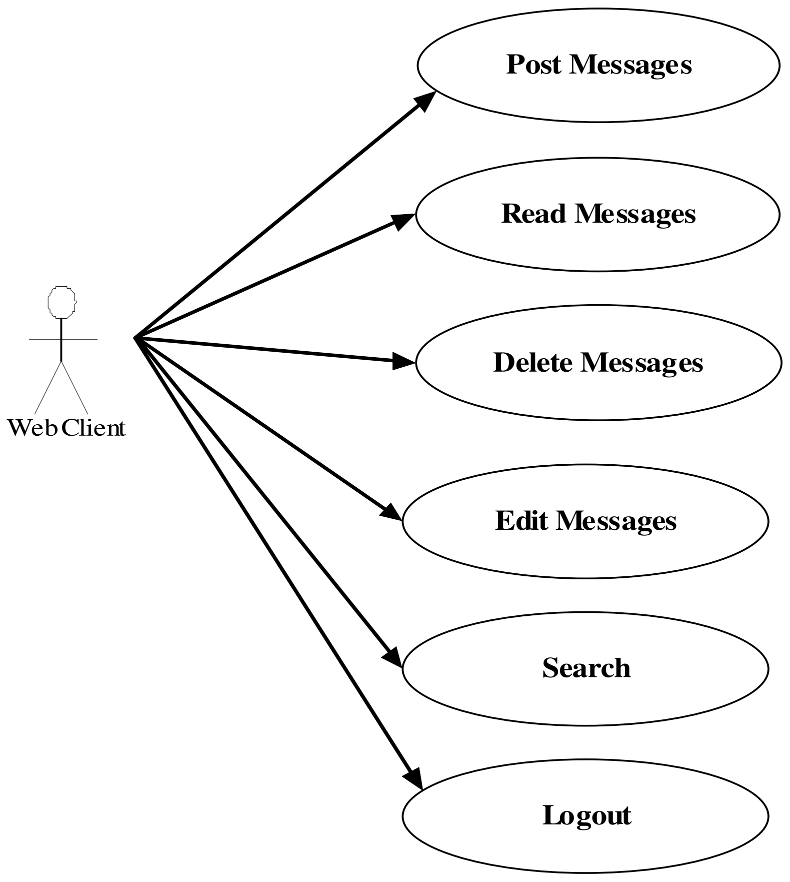
5.1.3.1 The Use-Case of the Newsgroup Client



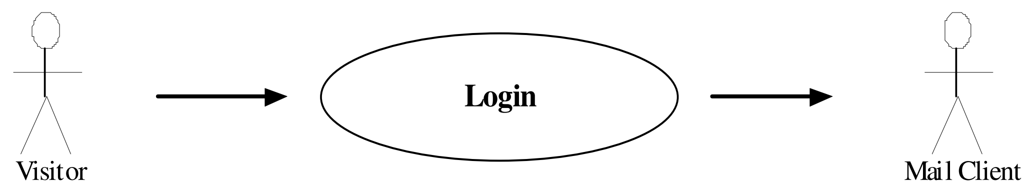


5.1.3.2 The Use-Case of the Web Client

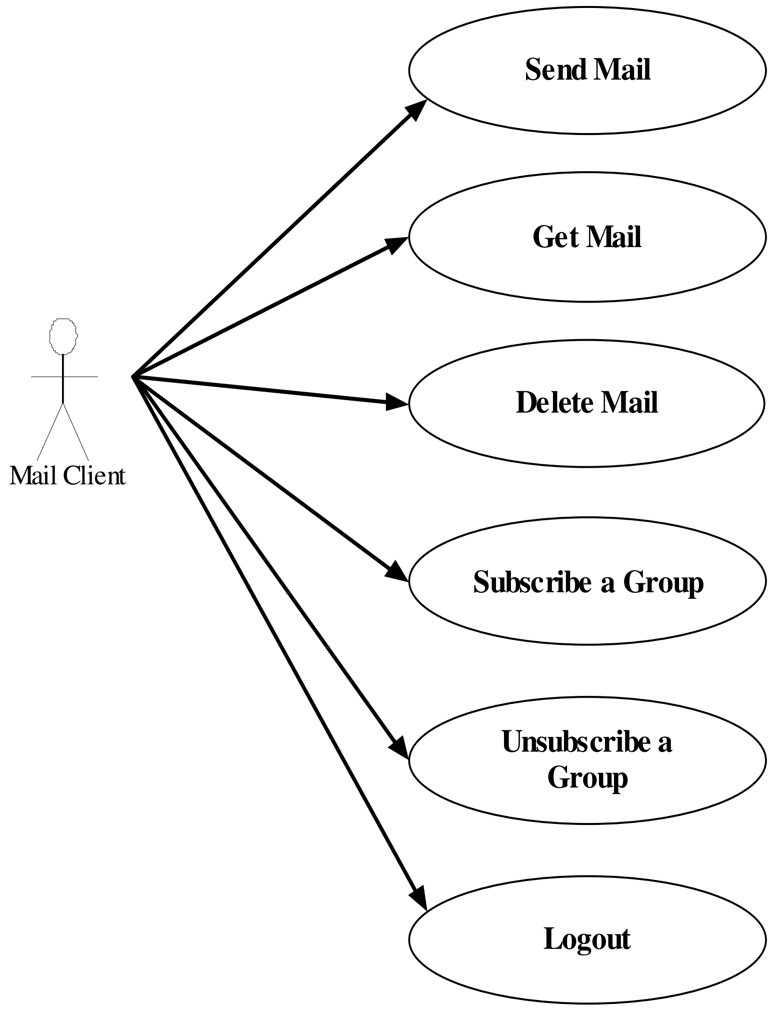




5.1.3.3 The Use-Case of the Mail Client

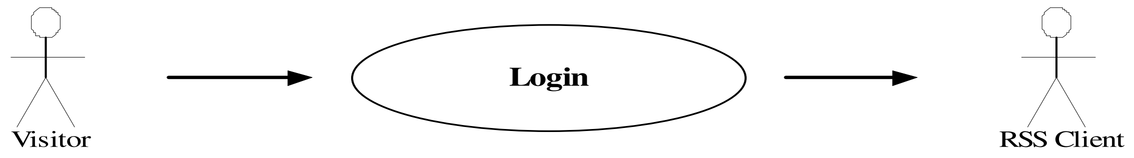


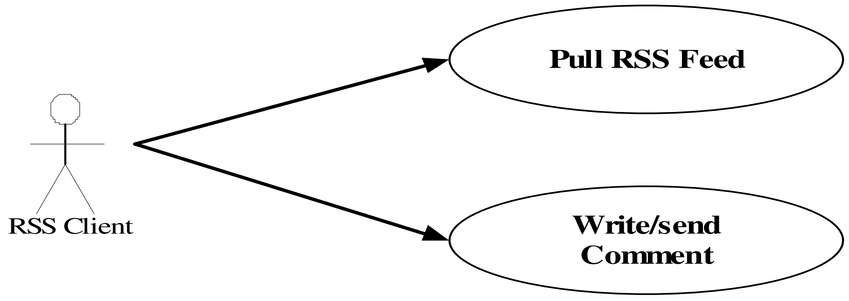
Visitor
Login
Mail Client



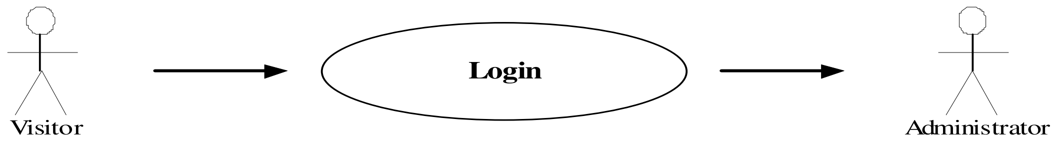
Mail Client
Send Mail
Get Mail
Delete Mail
Subscribe a Group
Unsubscribe a Group
Logout

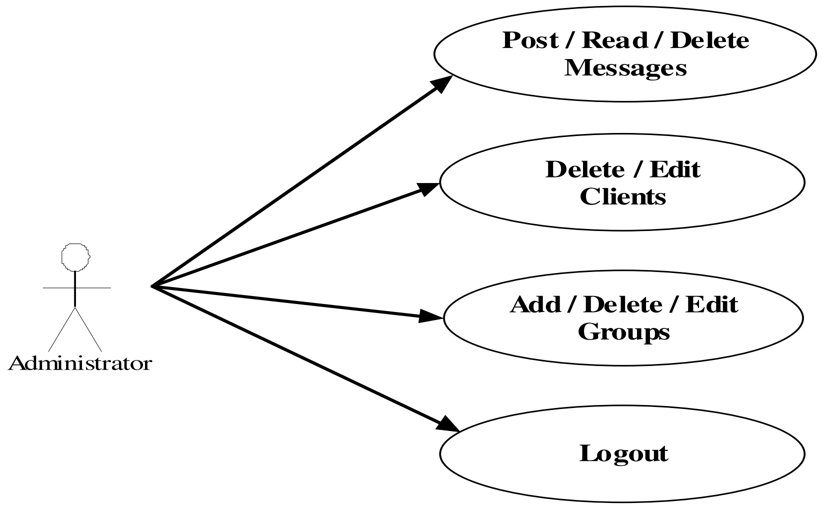
5.1.3.4 The Use-Case of the RSS Client





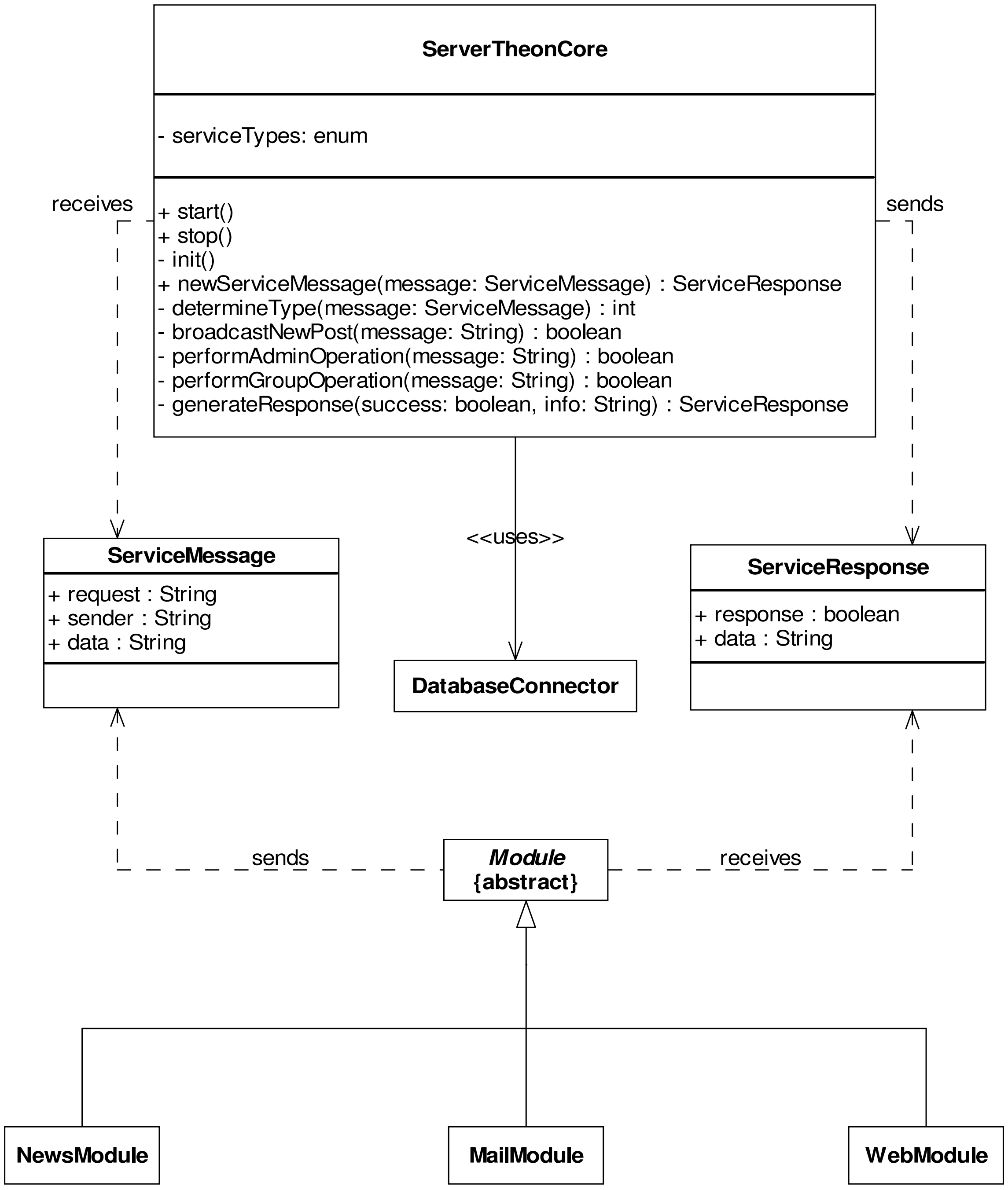
5.1.3.5 The Use-Case of the Administrator





5.2 Class Diagrams

5.2.1 Description of Core Service


ServerTheonCore
- serviceTypes: enum
+ start()
+ stop()
- init()
+ newServiceMessage(message: ServiceMessage) : ServiceResponse
- determineType(message: ServiceMessage) : int
- broadcastNewPost(message: String) : boolean
- performAdminOperation(message: String) : boolean
- performGroupOperation(message: String) : boolean
- generateResponse(success: boolean, info: String) : ServiceResponse
receives
sends
<<uses>>
ServiceMessage
+ request : String
+ sender : String
+ data : String
DatabaseConnector
ServiceResponse
+ response : boolean
+ data : String
sends
receives
Module
{abstract}
NewsModule
MailModule
WebModule

Attributes of *ServiceMessage*

Attribute Name	Description
request	Identifies the type of request
sender	Identifies module that sends this request
data	Contains additional information about the request such as message content that is posted or messageID of the message to be edited or deleted,etc.

Attributes of *ServiceResponse*

Method Name	Description
response	Indicates the result of the service operation (successful or not)
data	Contains additional information about the service response such as type of or the reason for the error.

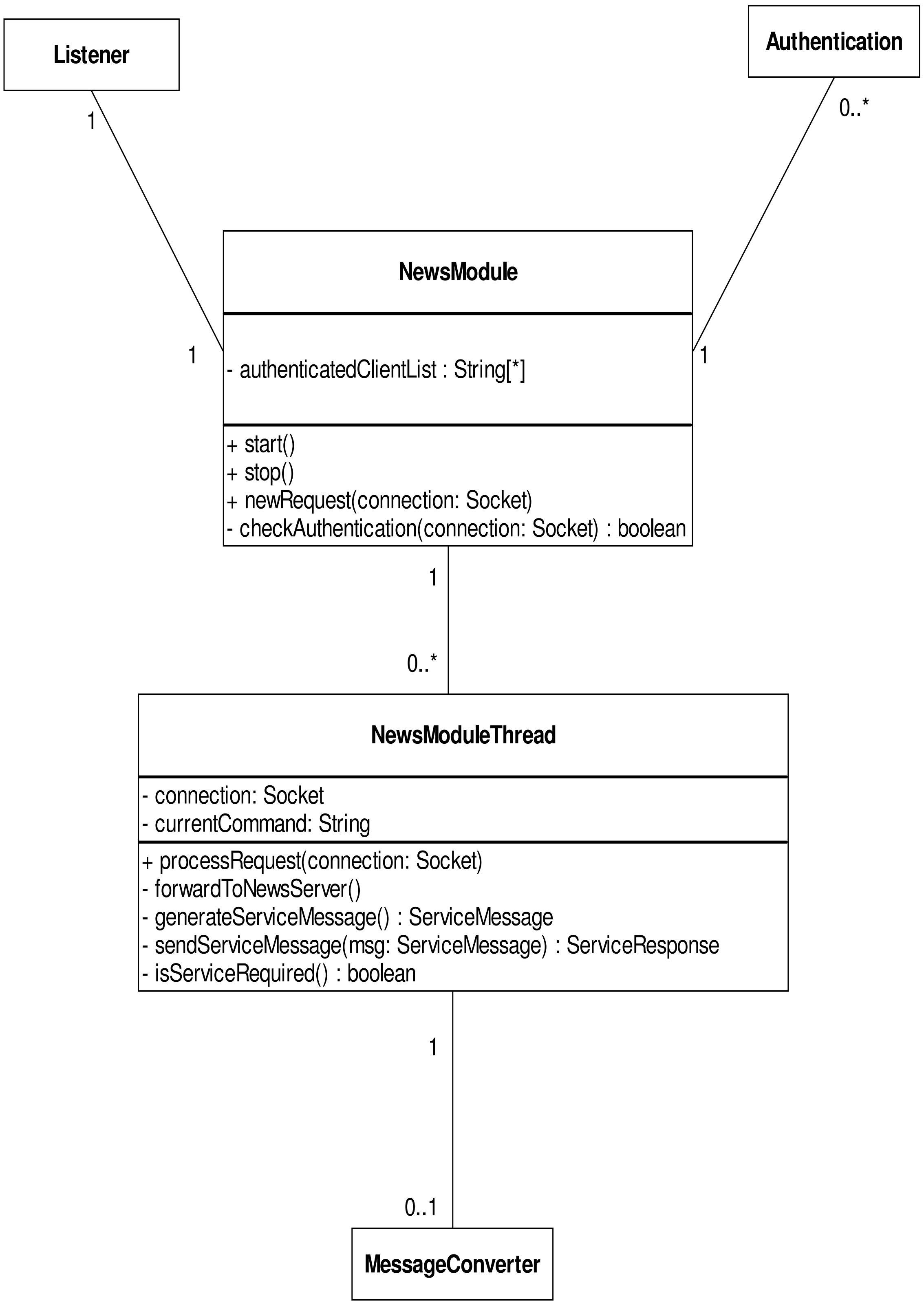
Attributes and methods of *ServerTheonCore*

Attribute Name	Description
serviceTypes	An enumeration of the service request types, used for determining which operation will be performed.

Method Name	Description
start	Starts ServerTheon by instantiating Server Theon modules to listen on specified ports in the configuration

Method Name	Description
	file.
stop	Stops ServerTheon by calling each module’s stop method.
newServiceMessage	Firstly, determines type of service request by invoking determineType method. Then, invokes related method, namely performAdminOperation, performGroupOperation, or broadcastNewPost. After the invoked method returns a result, this method calls generateResponse method for relevant ServiceResponse creation and finally, it returns back the generated response to the caller Module.
determineType	Returns the type of service requested as a value from serviceTypes enumeration. It uses request attribute of ServiceMessage class passed to it by newServiceMessage.
performGroupOperation	Processes service requests that related group subscriptions of the clients.
performAdminOperation	Processes service requests that have origin from AdminModule
broadcastNewPost	Distributes new messages posted using one access method, via the other access methods.
generateResponse	According to the parameters passed this method instantiate a ServiceResponse object and returns it to the caller method, namely newServiceRequest.

5.2.2 Description of News Module


Listener
Authentication
1
0..*
NewsModule
1
- authenticatedClientList : String[*]
1
+ start()
+ stop()
+ newRequest(connection: Socket)
- checkAuthentication(connection: Socket) : boolean
1
0..*
NewsModuleThread
- connection: Socket
- currentCommand: String
+ processRequest(connection: Socket)
- forwardToNewsServer()
- generateServiceMessage() : ServiceMessage
- sendServiceMessage(msg: ServiceMessage) : ServiceResponse
- isServiceRequired() : boolean
1
0..1
MessageConverter

Attributes and methods of *NewsModule*

Attribute Name	Description
authenticatedClientList	Array of strings for fast determining whether the origin of the incoming connection request is authenticated or not. Each string stores IP and port information generated by Authentication class.

Method Name	Description
start	Starts the module on the specified port determined via class constructor. Initiates a Listener object and makes it start listening.
stop	Stops the module after stopping the listener attached to it.
newRequest	Checks whether the connection made is authenticated or not by calling checkAuthentication method. If it is authenticated, runs a new thread of NewsModuleThread supplied with the connection socket. Otherwise, invokes Authentication class for requesting authentication from the user, then if Authentication class indicates a successful login saves the IP and port information in authenticatedClientList
checkAuthentication	Checks the authentication of the incoming connection by searching IP and

Method Name	Description
	remote port information in authenticatedClientList.

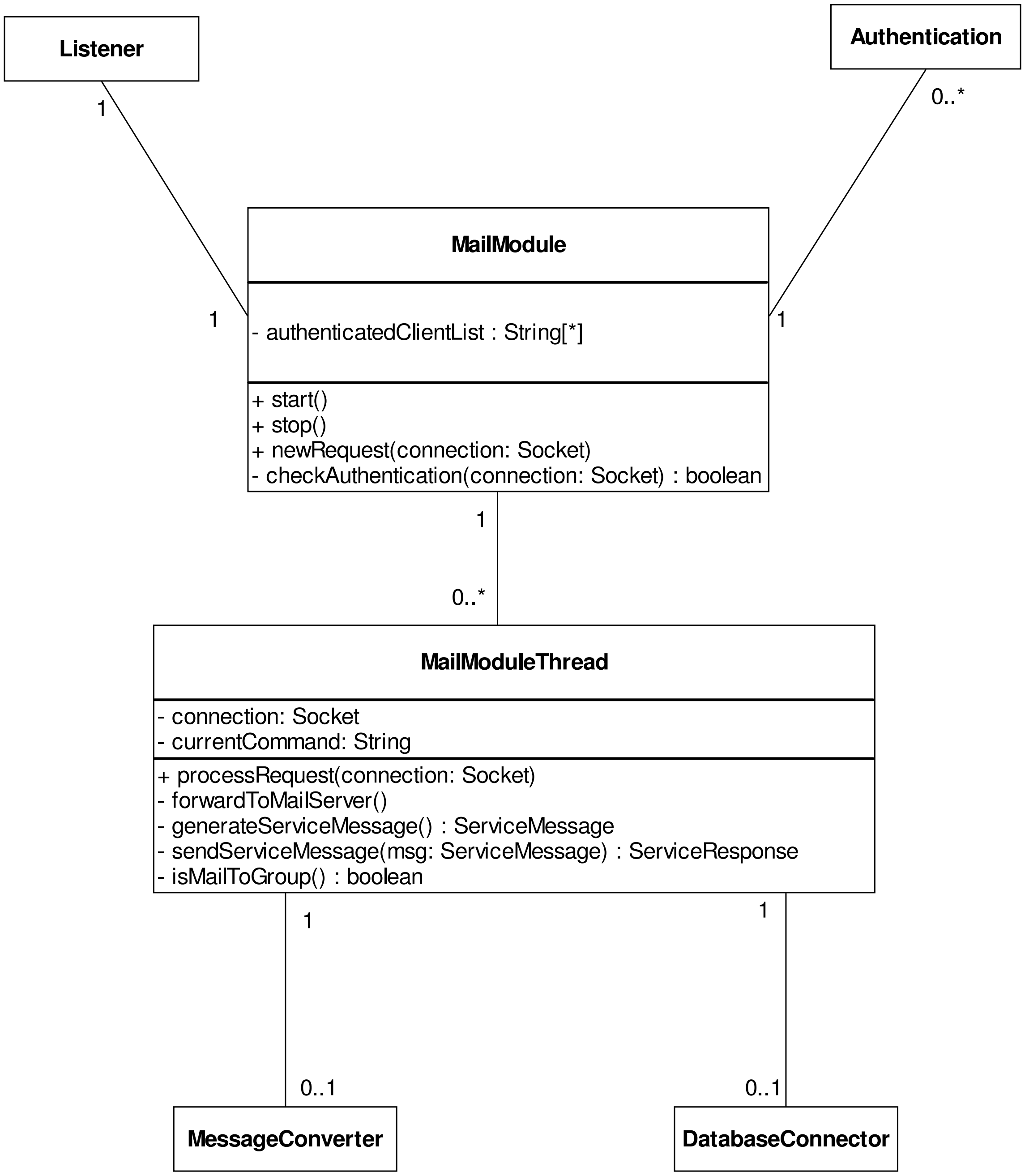
Attributes and methods of *NewsModuleThread*

Attribute Name	Description
connection	Socket of the connection with authenticated client, passed by NewsModule while running the thread.
currentCommand	String containing the packet data involving currently received NNTP command from the client. Updated continuously by processRequest method

Method Name	Description
processRequest	Connects the incoming socket to NNTP server and continuously checks the data flow through the socket. Using isServiceRequired method, this method controls the NNTP requests. If they are of type post or cancel, generateServiceMessage and sendServiceMessage methods are called respectively in order to notify ServerTheonCore. Otherwise, NNTP request is forwarded directly to NNTP server via using forwardToNewsServer method.

Method Name	**Description**
forwardToNewsServer	Forwards NNTP request by opening a socket to local news server port, hence response from the news server can directly be sent back using this socket.
generateServiceMessage	Instantiates a ServiceMessage object in accordance with the request contained in currentCommand attribute. Invokes a MessageConverter object for converting message in NNTP format to common format used by ServerTheon, then sets the attributes of this object and returns it back to caller method, namely processRequest.
sendServiceMessage	Invokes ServerTheonCore's newServiceMessage method passing ServiceMessage object generated by generateServiceMessage method.
isServiceRequired	Returns true if the currentCommand attribute contains NNTP post or cancel request, false otherwise.

5.2.3 Description of Mail Module



Listener
Authentication
1
0..*
MailModule
1
- authenticatedClientList : String[*]
1
+ start()
+ stop()
+ newRequest(connection: Socket)
- checkAuthentication(connection: Socket) : boolean
1
0..*
MailModuleThread
- connection: Socket
- currentCommand: String
+ processRequest(connection: Socket)
- forwardToMailServer()
- generateServiceMessage() : ServiceMessage
- sendServiceMessage(msg: ServiceMessage) : ServiceResponse
- isMailToGroup() : boolean
1
1
0..1
0..1
MessageConverter
DatabaseConnector

Attributes and methods of *MailModule*

Attribute Name	Description
authenticatedClientList	Array of strings for fast determining whether the origin of the incoming connection request is authenticated or not. Each string stores IP and port information generated by Authentication class.

Method Name	Description
start	Starts the module on the specified port determined via class constructor. Initiates a Listener object and makes it start listening.
stop	Stops the module after stopping the listener attached to it.
newRequest	Checks whether the connection made is authenticated or not by calling checkAuthentication method. If it is authenticated, runs a new thread of MailModuleThread supplied with the connection socket. Otherwise, invokes Authentication class for requesting authentication from the user, then if Authentication class indicates a successful login saves the IP and port information in authenticatedClientList
checkAuthentication	Checks the authentication of the incoming connection by searching IP and

Method Name	Description
	remote port information in authenticatedClientList.

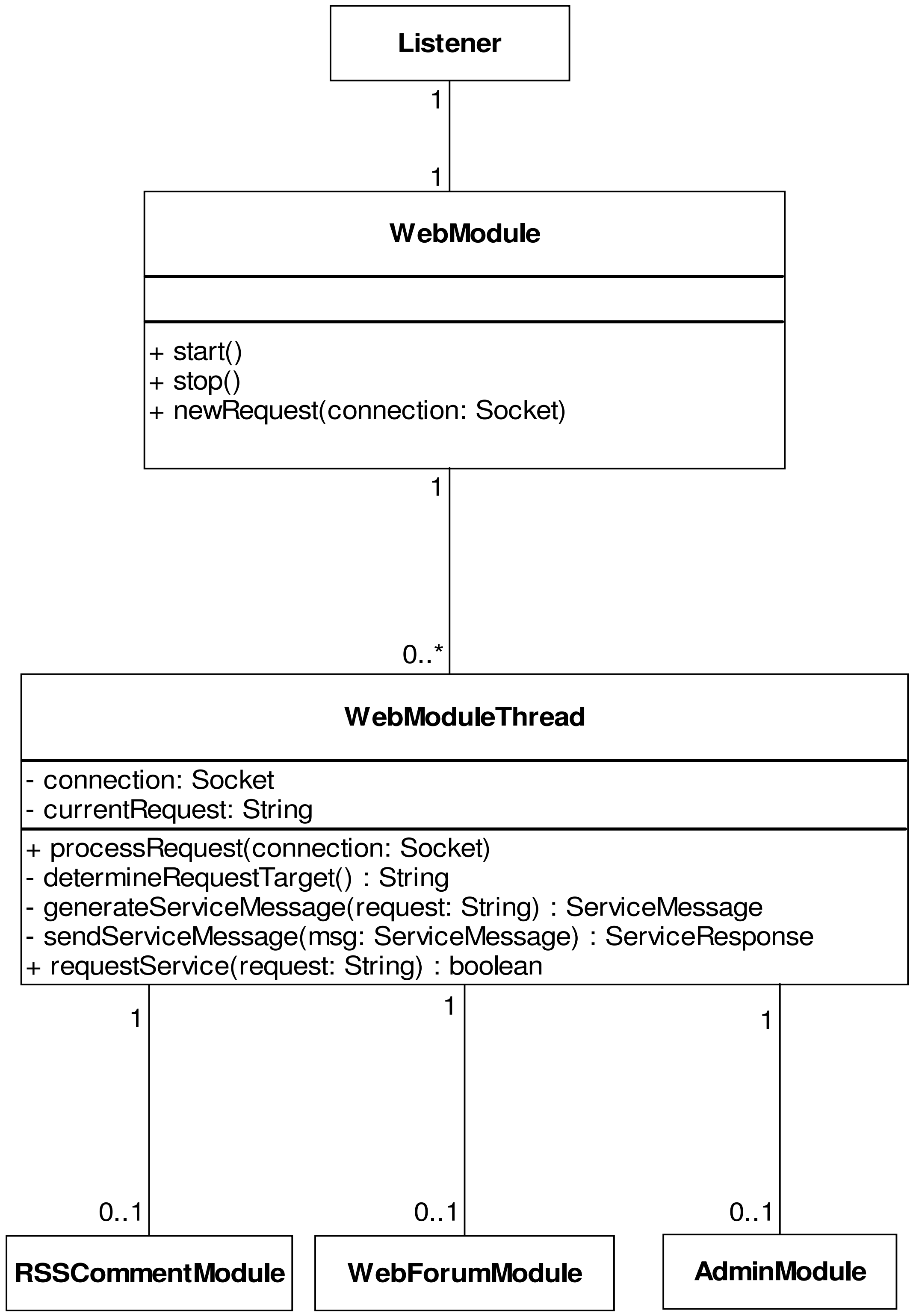
Attributes and methods of *MailModuleThread*

Attribute Name	Description
connection	Socket of the connection with authenticated client, passed by MailModule while running the thread.
currentCommand	String containing the packet data involving currently received SMTP command from the client. Updated continuously by processRequest method

Method Name	Description
processRequest	Connects the incoming socket to SMTP server and continuously checks the data flow through the socket. Using isMailToGroup method, this method controls the SMTP requests. If they contain any mailing list recipients, generateServiceMessage and sendServiceMessage methods are called respectively in order to notify ServerTheonCore. Otherwise, SMTP request is forwarded directly to SMTP server via using forwardToMailServer method.

Method Name	**Description**
forwardToMailServer	Forwards SMTP request by opening a socket to local mail server port, hence response from the mail server can directly be sent back using this socket.
generateServiceMessage	Instantiates a ServiceMessage object in accordance with the request contained in currentCommand attribute. Invokes a MessageConverter object for converting message in SMTP format to common format used by ServerTheon, then sets the attributes of this object and returns it back to caller method, namely processRequest.
sendServiceMessage	Invokes ServerTheonCore's newServiceMessage method passing ServiceMessage object generated by generateServiceMessage method.
isMailToGroup	Returns true if the currentCommand attribute contains RCPT line that points to a mailing list served by ServerTheon, false otherwise. While comparing e-mail addresses contained in RCPT lines, this Method invokes a DatabaseConnector object to fetch a list of mailing lists that are served by ServerTheon.

5.2.4 Description of Web Module



Listener
1
1
WebModule
+ start()
+ stop()
+ newRequest(connection: Socket)
1
0..*
WebModuleThread
- connection: Socket
- currentRequest: String
+ processRequest(connection: Socket)
- determineRequestTarget() : String
- generateServiceMessage(request: String) : ServiceMessage
- sendServiceMessage(msg: ServiceMessage) : ServiceResponse
+ requestService(request: String) : boolean
1
1
1
0..1
0..1
0..1
RSSCommentModule
WebForumModule
AdminModule

Methods of *WebModule*

Method Name	Description
start	Starts the module on the specified port determined via class constructor. Initiates a Listener object and makes it start listening.
stop	Stops the module after stopping the listener attached to it.
newRequest	Runs a new thread of WebModuleThread passing it the socket for the current incoming connection.

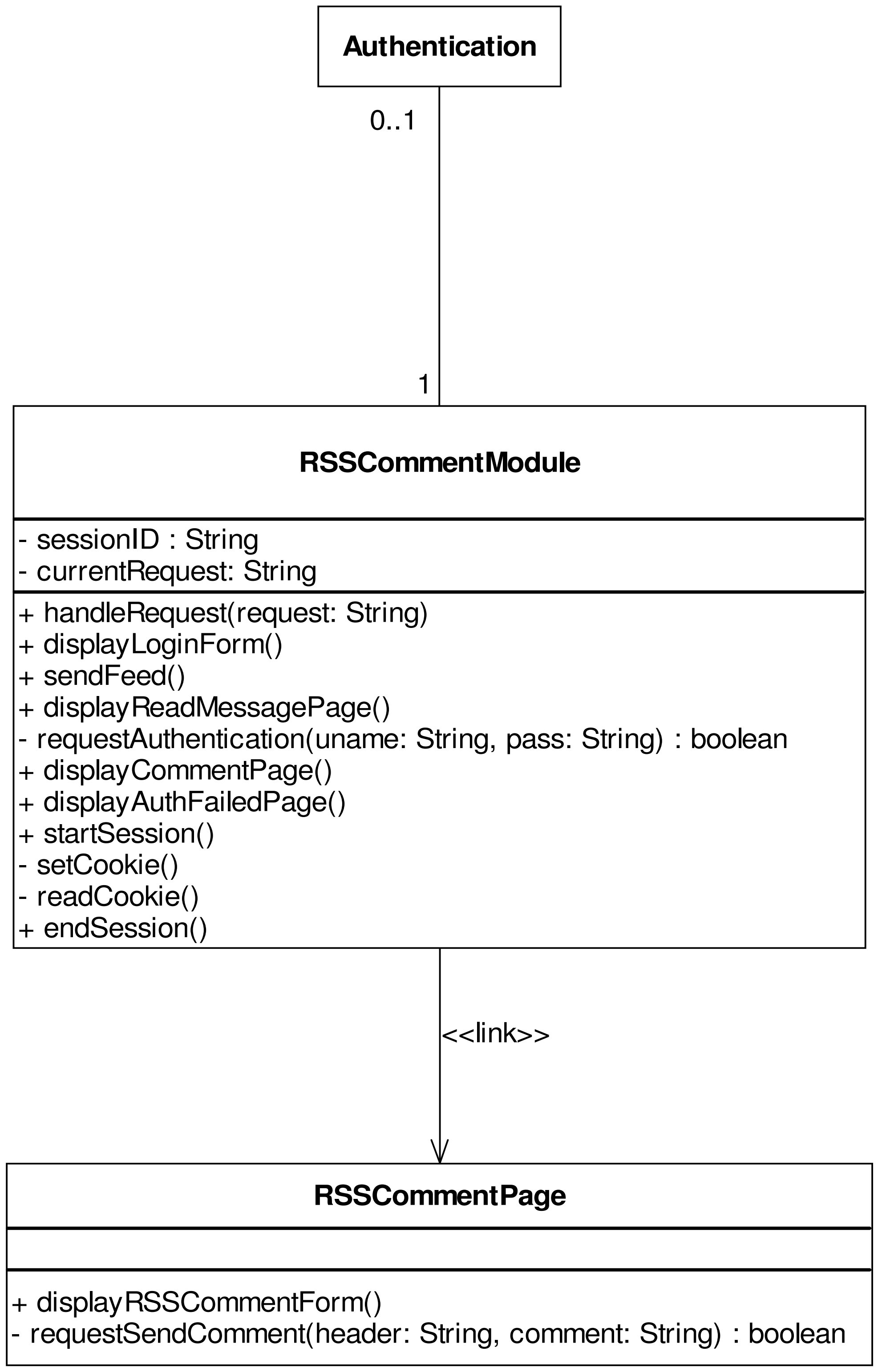
Attributes and methods of *WebModuleThread*

Attribute Name	Description
connection	Socket of the connection with the client, passed by WebModule while running the thread.
currentRequest	String containing the packet data involving currently received HTTP request from the client.

Method Name	Description
processRequest	Firstly, determines the target module of the incoming HTTP request by calling determineRequestTarget. Then, the target module indicated by the return value is invoked, and this

Method Name	**Description**
	thread waits for the invoked module to finish its job since it may invoke requestService method of this thread.
requestService	This method is called by the module invoked by processRequest method when necessary. It calls first generateServiceMessage and then sendServiceMessage methods for informing ServerTheonCore about an operation.
generateServiceMessage	Instantiates a ServiceMessage object in accordance with the request contained in request parameter, then sets the attributes of this object and returns it back to caller method, namely processRequest.
sendServiceMessage	Invokes ServerTheonCore's newServiceMessage method passing ServiceMessage object generated by generateServiceMessage method and returns ServiceResponse coming from ServerTheonCore to caller method.
determineRequestType	Determines the module to be instantiated by parsing the requested URI contained in currentRequest attribute.

5.2.5 Description of RSS Comment Module


Authentication
0..1
1
RSSCommentModule
- sessionID : String
- currentRequest: String
+ handleRequest(request: String)
+ displayLoginForm()
+ sendFeed()
+ displayReadMessagePage()
- requestAuthentication(uname: String, pass: String) : boolean
+ displayCommentPage()
+ displayAuthFailedPage()
+ startSession()
- setCookie()
- readCookie()
+ endSession()
<<link>>
RSSCommentPage
+ displayRSSCommentForm()
- requestSendComment(header: String, comment: String) : boolean

Attributes and methods of *RSSCommentModule*

Attribute Name	Description
sessionID	Session data of the currently connected client.
currentRequest	String containing the packet data involving currently received HTTP request from the client. Updated continuously by handleRequest method

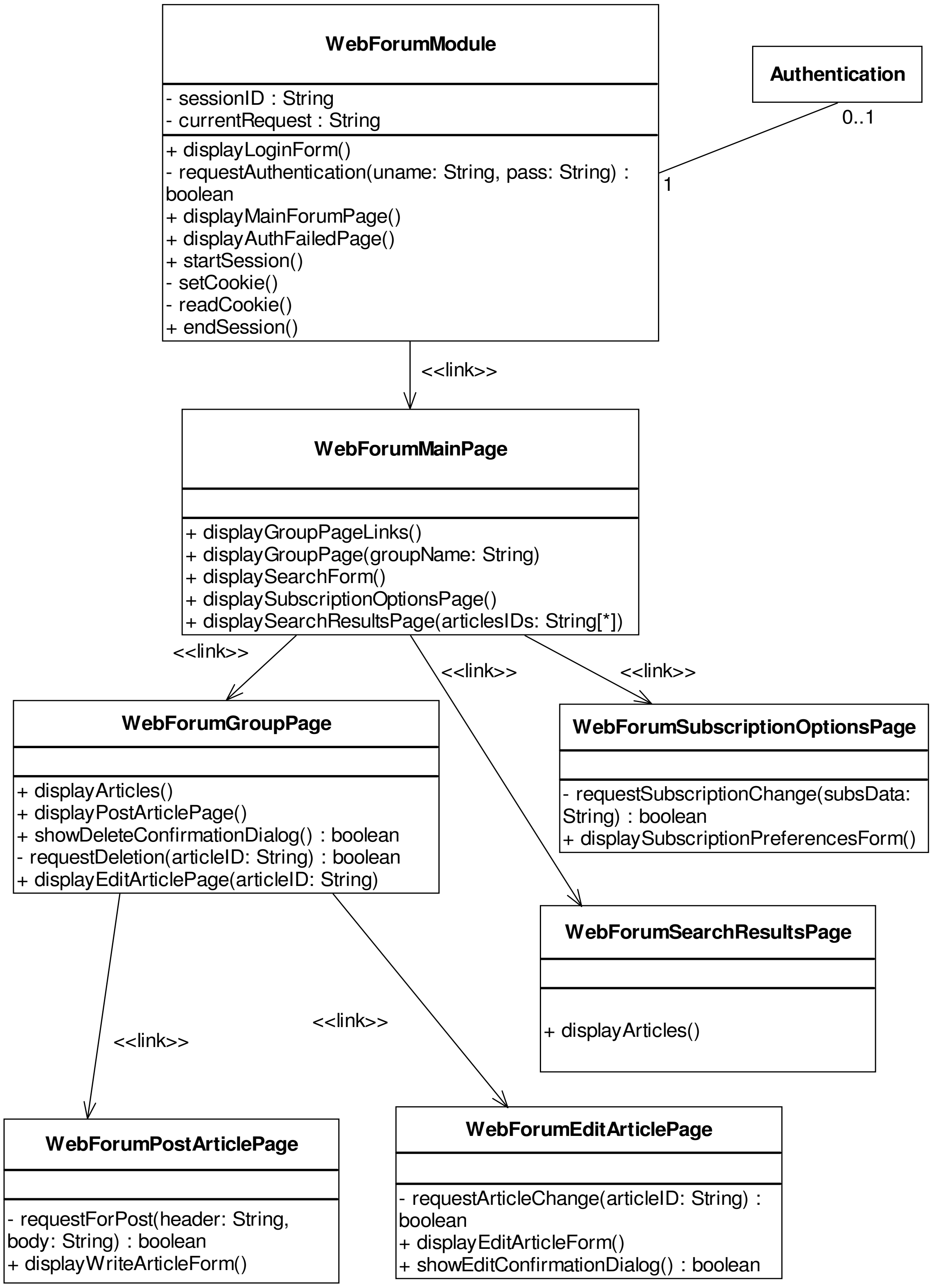
Method Name	Description
handleRequest	Connects the client to the web server then processes each HTTP request until session ends. If the client requesting a feed first check authentication by invoking Authentication class, after authentication succeeds it sends the requested feed by calling sendFeed method. If the client requesting access to a specific message displayReadMessagePage method is invoked provided that session data is correct. If the client is requesting to make a comment displayLoginForm method and either displayAuthFailedPage or displayCommentPage methods are invoked respectivey.
sendFeed	Triggers web server for sending the feed that is requested by the client.

Method Name	**Description**
displayLoginForm	Triggers the web server for sending HTTP response containing a web page that includes an html login form.
displayAuthFailedPage	Triggers the web server for sending HTTP response containing a web page that includes an authentication failed message.
displayReadMessagePage	Triggers the web server for sending HTTP response containing a web page that includes the content of the message that is determined by the HTTP request passed by this method to the web server.
displayCommentPage	Instantiates an object of class RSSCommentPage.
requestAuthentication	Invokes Authentication class with obtained username and password via login form.
startSession endSession setCookie readCookie	These methods are used for session operations that will be performed on both sides (client and server) for HTTP request/response communication.

Methods of *RSSCommentPage*

Method Name	**Description**
displayRSSCommentForm	Triggers web server for sending the html form used for comment message entry.
requestSendComment	Invokes requestService method of WebModuleThread from the instance that instantiated this RSSCommentModule object.

5.2.6 Description of Web Forum Module



WebForumModule
- sessionID : String
- currentRequest : String
+ displayLoginForm()
- requestAuthentication(uname: String, pass: String) : boolean
+ displayMainForumPage()
+ displayAuthFailedPage()
+ startSession()
- setCookie()
- readCookie()
+ endSession()
Authentication
0..1
1
<<link>>
WebForumMainPage
+ displayGroupPageLinks()
+ displayGroupPage(groupName: String)
+ displaySearchForm()
+ displaySubscriptionOptionsPage()
+ displaySearchResultsPage(articlesIDs: String[*])
<<link>>
<<link>>
<<link>>
WebForumGroupPage
+ displayArticles()
+ displayPostArticlePage()
+ showDeleteConfirmationDialog() : boolean
- requestDeletion(articleID: String) : boolean
+ displayEditArticlePage(articleID: String)
WebForumSubscriptionOptionsPage
- requestSubscriptionChange(subsData: String) : boolean
+ displaySubscriptionPreferencesForm()
WebForumSearchResultsPage
+ displayArticles()
<<link>>
<<link>>
WebForumPostArticlePage
- requestForPost(header: String, body: String) : boolean
+ displayWriteArticleForm()
WebForumEditArticlePage
- requestArticleChange(articleID: String) : boolean
+ displayEditArticleForm()
+ showEditConfirmationDialog() : boolean

Attributes and methods of *WebForumModule*

Attribute Name	**Description**
sessionID	Session data of the currently connected client.
currentRequest	String containing the packet data involving currently received HTTP request from the client. Updated continuously by handleRequest method

Method Name	**Description**
handleRequest	Connects the client to the web server then processes each HTTP request until session ends. displayLoginForm method and either displayAuthFailedPage or displayMainForumPage methods are invoked respectivey.
displayLoginForm	Triggers the web server for sending HTTP response containing a web page that includes an html login form.
displayAuthFailedPage	Triggers the web server for sending HTTP response containing a web page that includes an authentication failed message.
displayMainForumPage	Instantiates an object of class WebForumMainPage.
requestAuthentication	Invokes Authentication class with obtained username and password via login form.

Method Name	**Description**
startSession endSession setCookie readCookie	These methods are used for session operations that will be performed on both sides (client and server) for HTTP request/response communication.

Methods of *WebForumMainPage*

Method Name	**Description**
displayGroupPageLinks displaySearchForm	These methods constitute the frames for main ForumTheon page, via triggering the web server.
displaySubscriptionOptionsPage	Instantiates an object of class WebForumSubscriptionOptionsPage.
displaySearchResultsPage	Instantiates an object of class WebForumSearchResultsPage, passing as parameter the results obtained via the search made by triggering web server.
displayGroupPage	Instantiates an object of class WebForumGroupPage, passing it the name of the slected group.

Methods of *WebForumGroupPage*

Method Name	**Description**
displayArticles	Triggers the web server for sending HTTP response containing a web page that includes the

Method Name	Description
	contents of the messages in group determined by parameter passed.
displayPostArticlePage	Instantiates an object of class WebForumPostArticlePage.
displayEditArticlePage	Instantiates an object of class WebForumEditArticlePage, passing as parameter the id of the message to be edited.
showDeleteConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for deletion.
requestDeletion	Invokes requestService method of WebModuleThread from the instance that instantiated this WebForumModule object, sending message deletion request.

Methods of *WebForumPostArticlePage*

Method Name	Description
displayWriteArticleForm	Triggers the web server for sending HTTP response containing a web page that includes a form for article posting.
requestForPost	Invokes requestService method of WebModuleThread from the instance that instantiated this

Method Name	Description
	WebForumModule object, sending message posting request.

Methods of *WebForumEditArticlePage*

Method Name	Description
displayEditArticleForm	Triggers the web server for sending HTTP response containing a web page that includes a form for article editing.
showEditConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for editing.
requestArticleChange	Invokes requestService method of WebModuleThread from the instance that instantiated this WebForumModule object, sending message changing request.

Methods of *WebForumSubscriptionOptionsPage*

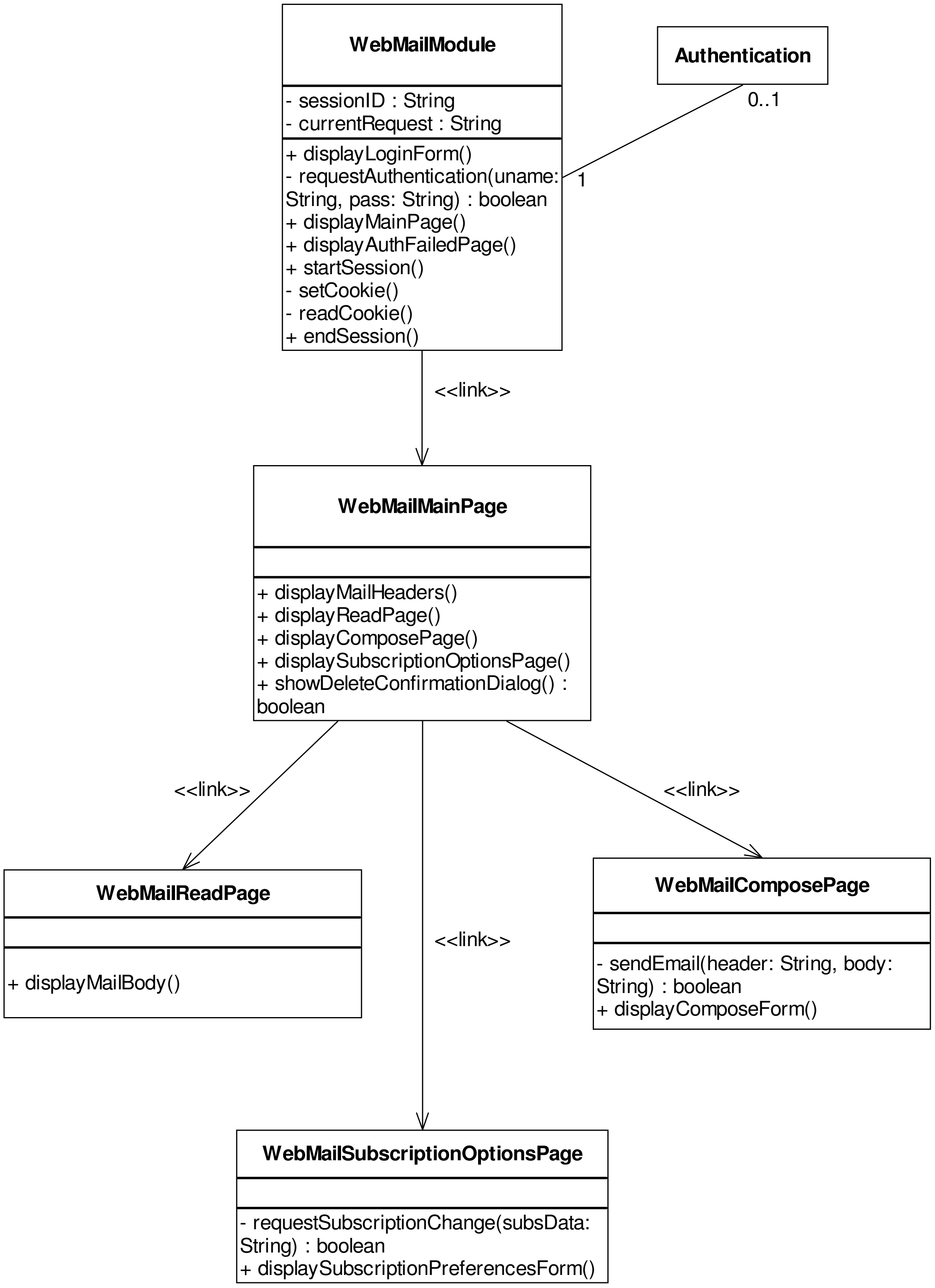
Method Name	Description
displaySubscriptionPreferencesForm	Triggers the web server for sending HTTP response

Method Name	Description
	containing a web page that includes a form for subscription preferences including old preferences of the user.
requestSubscriptionChange	Invokes requestService method of WebModuleThread from the instance that instantiated this WebForumModule object, sending subscription changing request.

Methods of *WebForumSearchResultsPage*

Method Name	Description
displayArticles	Triggers the web server for sending HTTP response containing a web page that includes the contents of the messages that is determined by search request for parameter obtained and passed by this method to the web server.

5.2.7 Description of Web Mail Module


WebMailModule
- sessionID : String
- currentRequest : String
+ displayLoginForm()
- requestAuthentication(uname: String, pass: String) : boolean
+ displayMainPage()
+ displayAuthFailedPage()
+ startSession()
- setCookie()
- readCookie()
+ endSession()
Authentication
0..1
1
<<link>>
WebMailMainPage
+ displayMailHeaders()
+ displayReadPage()
+ displayComposePage()
+ displaySubscriptionOptionsPage()
+ showDeleteConfirmationDialog() : boolean
<<link>>
<<link>>
<<link>>
WebMailReadPage
+ displayMailBody()
WebMailComposePage
- sendEmail(header: String, body: String) : boolean
+ displayComposeForm()
WebMailSubscriptionOptionsPage
- requestSubscriptionChange(subsData: String) : boolean
+ displaySubscriptionPreferencesForm()

Attributes and methods of *WebMailModule*

Attribute Name	Description
sessionID	Session data of the currently connected client.
currentRequest	String containing the packet data involving currently received HTTP request from the client. Updated continuously by handleRequest method

Method Name	Description
handleRequest	Connects the client to the web server then processes each HTTP request until session ends. displayLoginForm method and either displayAuthFailedPage or displayMainPage methods are invoked respectivey.
displayLoginForm	Triggers the web server for sending HTTP response containing a web page that includes an html login form.
displayAuthFailedPage	Triggers the web server for sending HTTP response containing a web page that includes an authentication failed message.
displayMainForumPage	Instantiates an object of class WebMailMainPage.
requestAuthentication	Invokes Authentication class with obtained username and password via login form.

Method Name	Description
startSession endSession setCookie readCookie	These methods are used for session operations that will be performed on both sides (client and server) for HTTP request/response communication.

Methods of *WebMailMainPage*

Method Name	Description
displayMailHeaders	Triggers the web server for sending HTTP response containing a web page that includes the messages of user inbox.
displaySubscriptionOptionsPage	Instantiates an object of class WebMailSubscriptionOptionsPage.
displayReadPage	Instantiates an object of class WebMailReadPage, passing as parameter the id of the message to be read.
displayComposePage	Instantiates an object of class WebMailComposePage.
showDeleteConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for deletion.

Methods of *WebMailComposePage*

Method Name	Description
displayComposeForm	Triggers the web server for sending HTTP response containing a web page that includes a form for composing e-mail.
sendEmail	Sends e-mail directly to SMTP server.

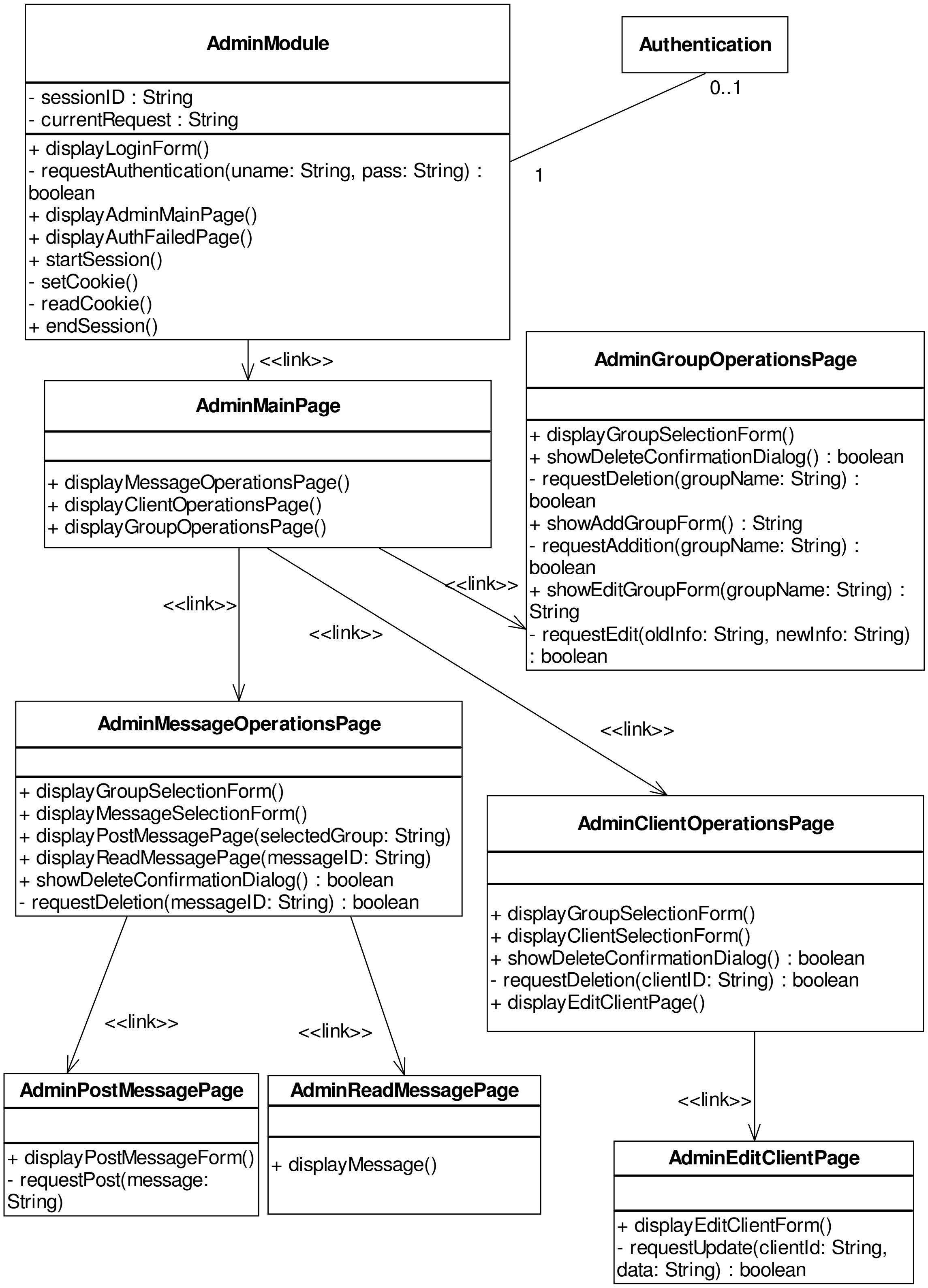
Methods of *WebMailSubscriptionOptionsPage*

Method Name	Description
displaySubscriptionPreferencesForm	Triggers the web server for sending HTTP response containing a web page that includes a form for subscription preferences including old preferences of the user.
requestSubscriptionChange	Invokes requestService method of WebModuleThread from the instance that instantiated this WebMailModule object, sending subscription changing request.

Methods of *WebMailReadPage*

Method Name	**Description**
displayMailBody	Triggers the web server for sending HTTP response containing a web page that includes the content of the message that is determined by message id obtained as parameter obtained and passed by this method to the web server.

5.2.8 Description of Admin Module


AdminModule
- sessionID : String
- currentRequest : String
+ displayLoginForm()
- requestAuthentication(uname: String, pass: String) : boolean
+ displayAdminMainPage()
+ displayAuthFailedPage()
+ startSession()
- setCookie()
- readCookie()
+ endSession()
Authentication
0..1
1
<<link>>
AdminMainPage
+ displayMessageOperationsPage()
+ displayClientOperationsPage()
+ displayGroupOperationsPage()
AdminGroupOperationsPage
+ displayGroupSelectionForm()
+ showDeleteConfirmationDialog() : boolean
- requestDeletion(groupName: String) : boolean
+ showAddGroupForm() : String
- requestAddition(groupName: String) : boolean
+ showEditGroupForm(groupName: String) : String
- requestEdit(oldInfo: String, newInfo: String) : boolean
<<link>>
<<link>>
<<link>>
<<link>>
AdminMessageOperationsPage
+ displayGroupSelectionForm()
+ displayMessageSelectionForm()
+ displayPostMessagePage(selectedGroup: String)
+ displayReadMessagePage(messageID: String)
+ showDeleteConfirmationDialog() : boolean
- requestDeletion(messageID: String) : boolean
AdminClientOperationsPage
+ displayGroupSelectionForm()
+ displayClientSelectionForm()
+ showDeleteConfirmationDialog() : boolean
- requestDeletion(clientID: String) : boolean
+ displayEditClientPage()
<<link>>
<<link>>
<<link>>
AdminPostMessagePage
+ displayPostMessageForm()
- requestPost(message: String)
AdminReadMessagePage
+ displayMessage()
AdminEditClientPage
+ displayEditClientForm()
- requestUpdate(clientId: String, data: String) : boolean

Attributes and methods of *AdminModule*

Attribute Name	Description
sessionID	Session data of the currently connected client.
currentRequest	String containing the packet data involving currently received HTTP request from the client. Updated continuously by handleRequest method

Method Name	Description
handleRequest	Connects the client to the web server then processes each HTTP request until session ends. displayLoginForm method and either displayAuthFailedPage or displayAdminMainPage methods are invoked respectivey.
displayLoginForm	Triggers the web server for sending HTTP response containing a web page that includes an html login form.
displayAuthFailedPage	Triggers the web server for sending HTTP response containing a web page that includes an authentication failed message.
displayMainForumPage	Instantiates an object of class AdminMainPage.
requestAuthentication	Invokes Authentication class with obtained username and password via login form.

Method Name	Description
startSession endSession setCookie readCookie	These methods are used for session operations that will be performed on both sides (client and server) for HTTP request/response communication.

Methods of *AdminMainPage*

Method Name	Description
displayGroupOperationsPage	Instantiates an object of class AdminGroupOperationsPage.
displayClientOperationsPage	Instantiates an object of class AdminClientOperationsPage.
displayMessageOperationsPage	Instantiates an object of class AdminMessageOperationsPage.

Methods of *AdminGroupOperationsPage*

Method Name	Description
displayGroupSelectionForm	Triggers the web server for sending HTTP response containing a web page that includes a form for group selection.
showDeleteConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for deletion of group.
showAddGroupForm	Triggers the web server for

Method Name	Description
	sending HTTP response containing that includes a form for new group information.
showEditGroupForm	Triggers the web server for sending HTTP response containing that includes a form for editing group information.
requestAddition	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending group addition request.
requestEdit	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending group editing request.
requestDeletion	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending group deletion request.

Methods of *AdminMessageOperationsPage*

Method Name	Description
displayGroupSelectionForm displayMessageSelectionForm	These methods constitute the frames for main Admin page, via triggering the web server.

Method Name	Description
showDeleteConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for deletion of a message.
displayAdminReadMessagePage	Instantiates an object of class AdminReadMessagePage, passing as parameter the id of the message to be read.
displayAdminPostMessagePage	Instantiates an object of class AdminPostMessagePage, passing as parameter the id of the group to which the message will be posted.
requestDeletion	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending message deletion request.

Methods of *AdminReadMessagePage*

Method Name	Description
displayMessage	Triggers the web server for sending HTTP response containing a web page that includes the

Method Name	Description
	content of the message whose id is passed as parameter.

Methods of *AdminReadMessagePage*

Method Name	Description
displayPostMessageForm	Triggers the web server for sending HTTP response containing a web page that includes a form for article posting.
requestPost	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending message posting request.

Methods of *AdminClientOperationsPage*

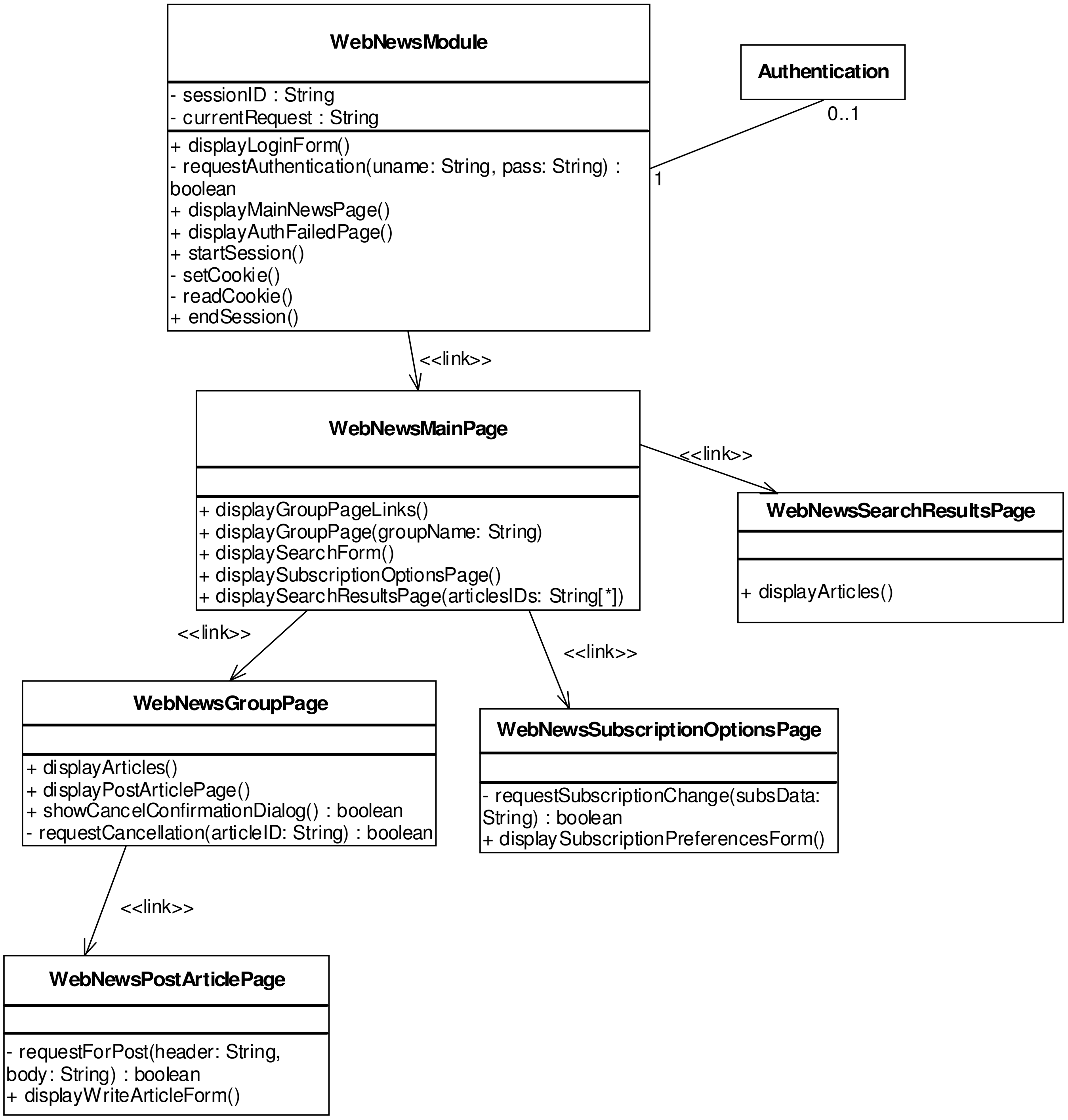
Method Name	Description
displayGroupSelectionForm displayClientSelectionForm	These methods constitute the frames for main client operations page, via triggering the web server.
showDeleteConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for deletion of a user.

Method Name	Description
displayEditClientPage	Instantiates an object of class AdminEditClientPage, passing as parameter the id of the message to be read.
requestDeletion	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending user deletion request.

Methods of *AdminEditClientPage*

Method Name	Description
displayEditClientForm	Triggers the web server for sending HTTP response containing a web page that includes a form for user editing.
requestUpdate	Invokes requestService method of WebModuleThread from the instance that instantiated this AdminModule object, sending user update request.

5.2.9 Description of Web News (NewsTheon) Module


WebNewsModule
- sessionID : String
- currentRequest : String
+ displayLoginForm()
- requestAuthentication(uname: String, pass: String) : boolean
+ displayMainNewsPage()
+ displayAuthFailedPage()
+ startSession()
- setCookie()
- readCookie()
+ endSession()
Authentication
0..1
1
<<link>>
WebNewsMainPage
+ displayGroupPageLinks()
+ displayGroupPage(groupName: String)
+ displaySearchForm()
+ displaySubscriptionOptionsPage()
+ displaySearchResultsPage(articlesIDs: String[*])
<<link>>
WebNewsSearchResultsPage
+ displayArticles()
<<link>>
WebNewsGroupPage
+ displayArticles()
+ displayPostArticlePage()
+ showCancelConfirmationDialog() : boolean
- requestCancellation(articleID: String) : boolean
<<link>>
WebNewsSubscriptionOptionsPage
- requestSubscriptionChange(subsData: String) : boolean
+ displaySubscriptionPreferencesForm()
<<link>>
WebNewsPostArticlePage
- requestForPost(header: String, body: String) : boolean
+ displayWriteArticleForm()

Attributes and methods of *WebNewsModule*

Attribute Name	Description
sessionID	Session data of the currently connected client.
currentRequest	String containing the packet data involving currently received HTTP request from the client. Updated continuously by handleRequest method

Method Name	Description
handleRequest	Connects the client to the web server then processes each HTTP request until session ends. displayLoginForm method and either displayAuthFailedPage or displayMainForumPage methods are invoked respectivey.
displayLoginForm	Triggers the web server for sending HTTP response containing a web page that includes an html login form.
displayAuthFailedPage	Triggers the web server for sending HTTP response containing a web page that includes an authentication failed message.
displayMainNewsPage	Instantiates an object of class WebNewsMainPage.
requestAuthentication	Invokes Authentication class with obtained username and password via login form.

Method Name	**Description**
startSession endSession setCookie readCookie	These methods are used for session operations that will be performed on both sides (client and server) for HTTP request/response communication.

Methods of *WebNewsMainPage*

Method Name	**Description**
displayGroupPageLinks displaySearchForm	These methods constitute the frames for main NewsTheon page, via triggering the web server.
displaySubscriptionOptionsPage	Instantiates an object of class WebNewsSubscriptionOptionsPage.
displaySearchResultsPage	Instantiates an object of class WebNewsSearchResultsPage, passing as parameter the results obtained via the search made by triggering web server.
displayGroupPage	Instantiates an object of class WebNewsGroupPage, passing it the name of the selected group.

Methods of *WebNewsGroupPage*

Method Name	**Description**
displayArticles	Triggers the web server for sending HTTP response containing a web page that includes the

Method Name	Description
	contents of the messages in group determined by parameter passed.
displayPostArticlePage	Instantiates an object of class WebNewsPostArticlePage.
showCancelConfirmationDialog	Triggers the web server for sending HTTP response containing a dialog data that includes confirmation request for cancellation.
requestCancellation	Invokes requestService method of WebModuleThread from the instance that instantiated this WebNewsModule object, sending message request.

Methods of *WebNewsPostArticlePage*

Method Name	Description
displayWriteArticleForm	Triggers the web server for sending HTTP response containing a web page that includes a form for article posting.
requestForPost	Invokes requestService method of WebNewsThread from the instance that instantiated this WebNewsModule object, sending message posting request.

Methods of *WebNewsSubscriptionOptionsPage*

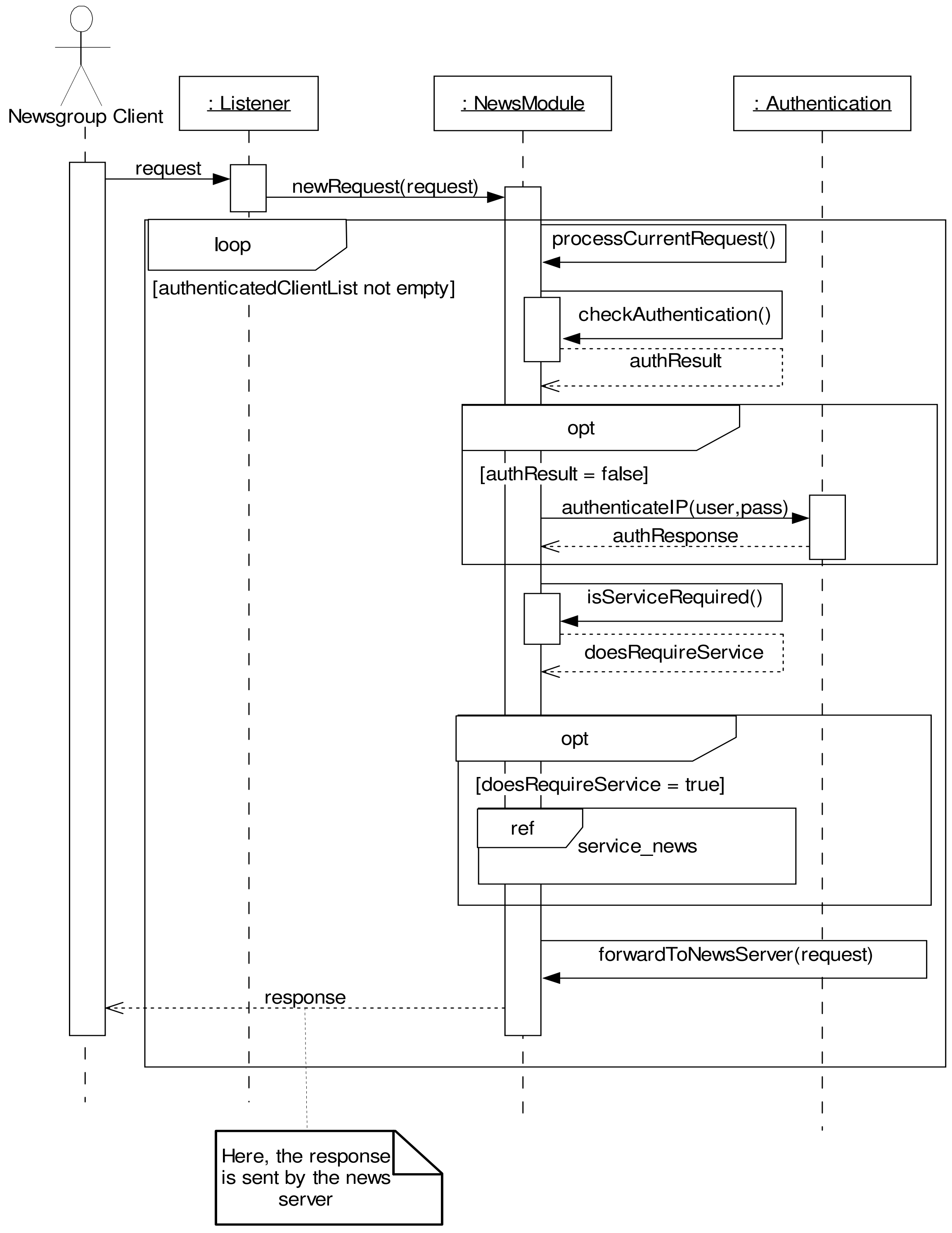
Method Name	**Description**
displaySubscriptionPreferencesForm	Triggers the web server for sending HTTP response containing a web page that includes a form for subscription preferences including old preferences of the user.
requestSubscriptionChange	Invokes requestService method of WebModuleThread from the instance that instantiated this WebNewsModule object, sending subscription changing request.

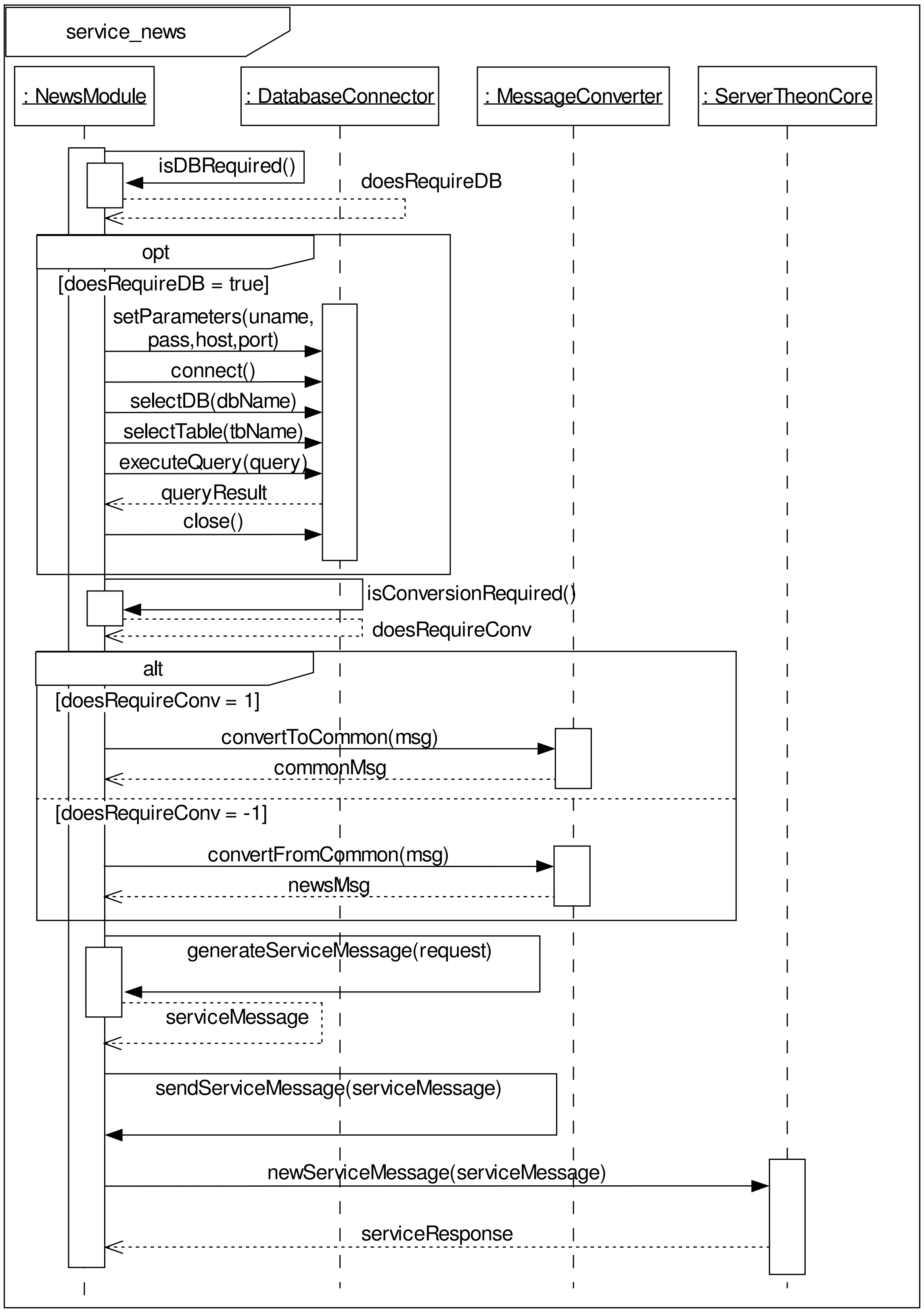
Methods of *WebNewsSearchResultsPage*

Method Name	**Description**
displayArticles	Triggers the web server for sending HTTP response containing a web page that includes the contents of the messages that is determined by search request for parameter obtained and passed by this method to the web server.

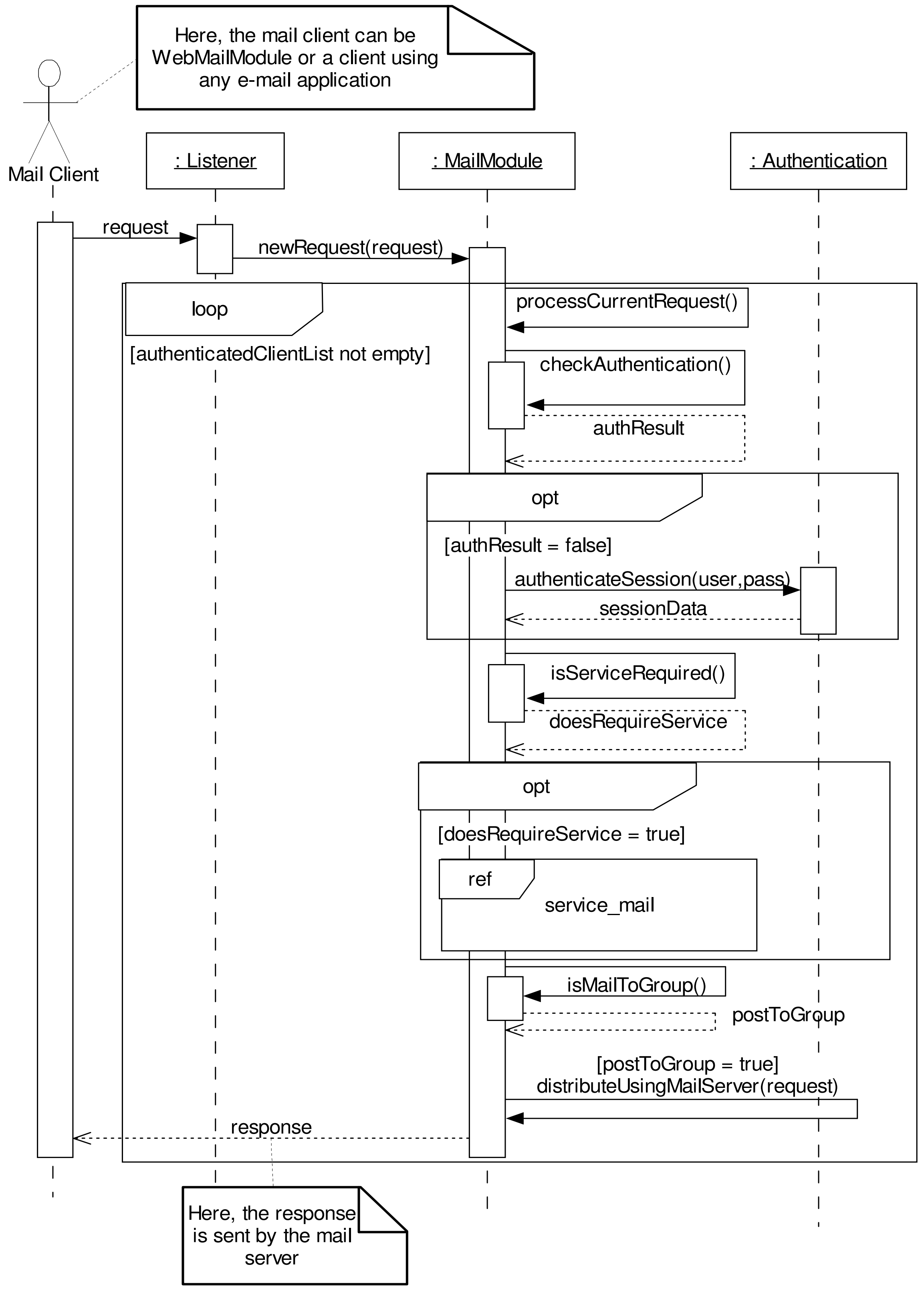
5.3 Sequence Diagrams

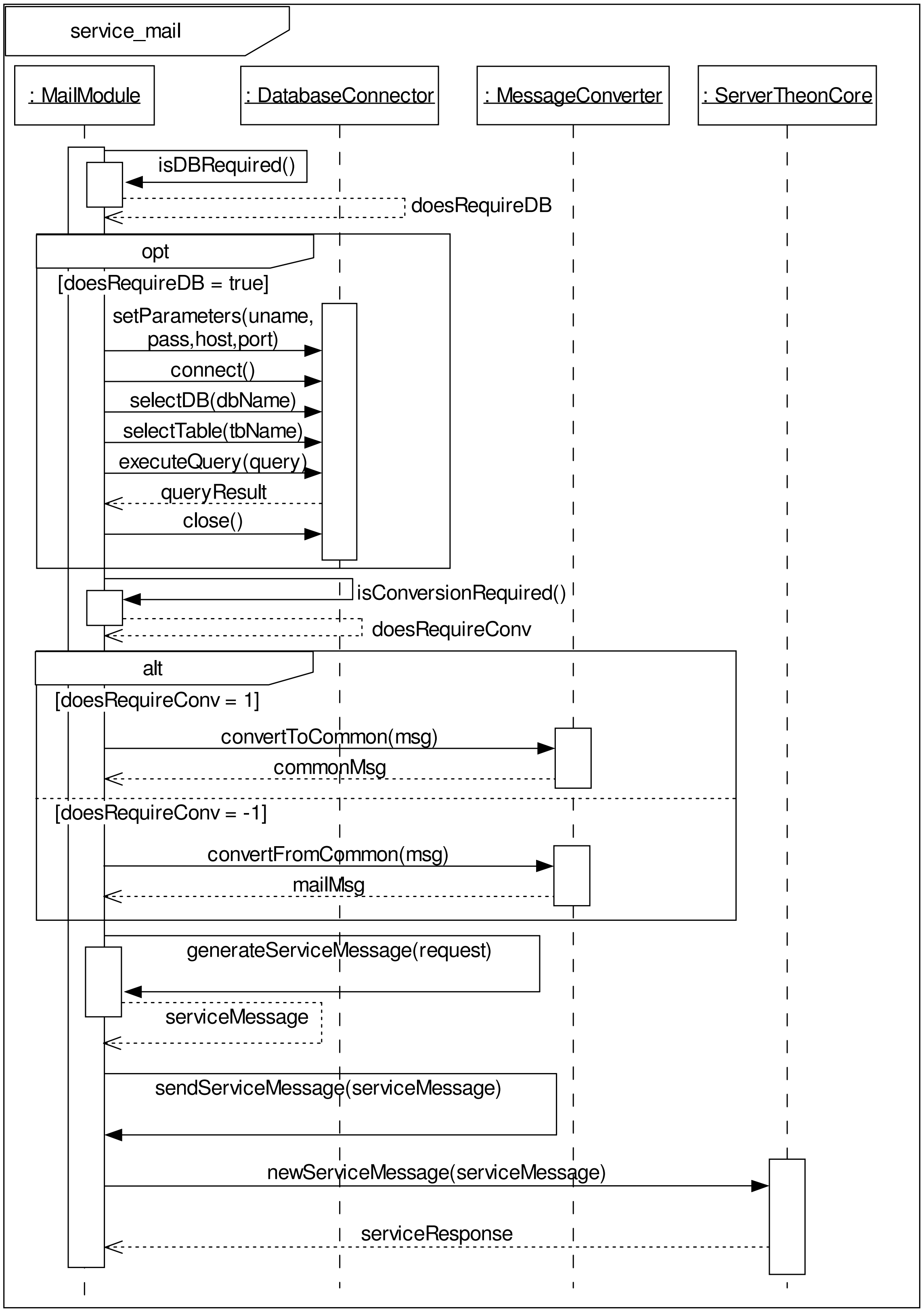
5.3.1 News Module



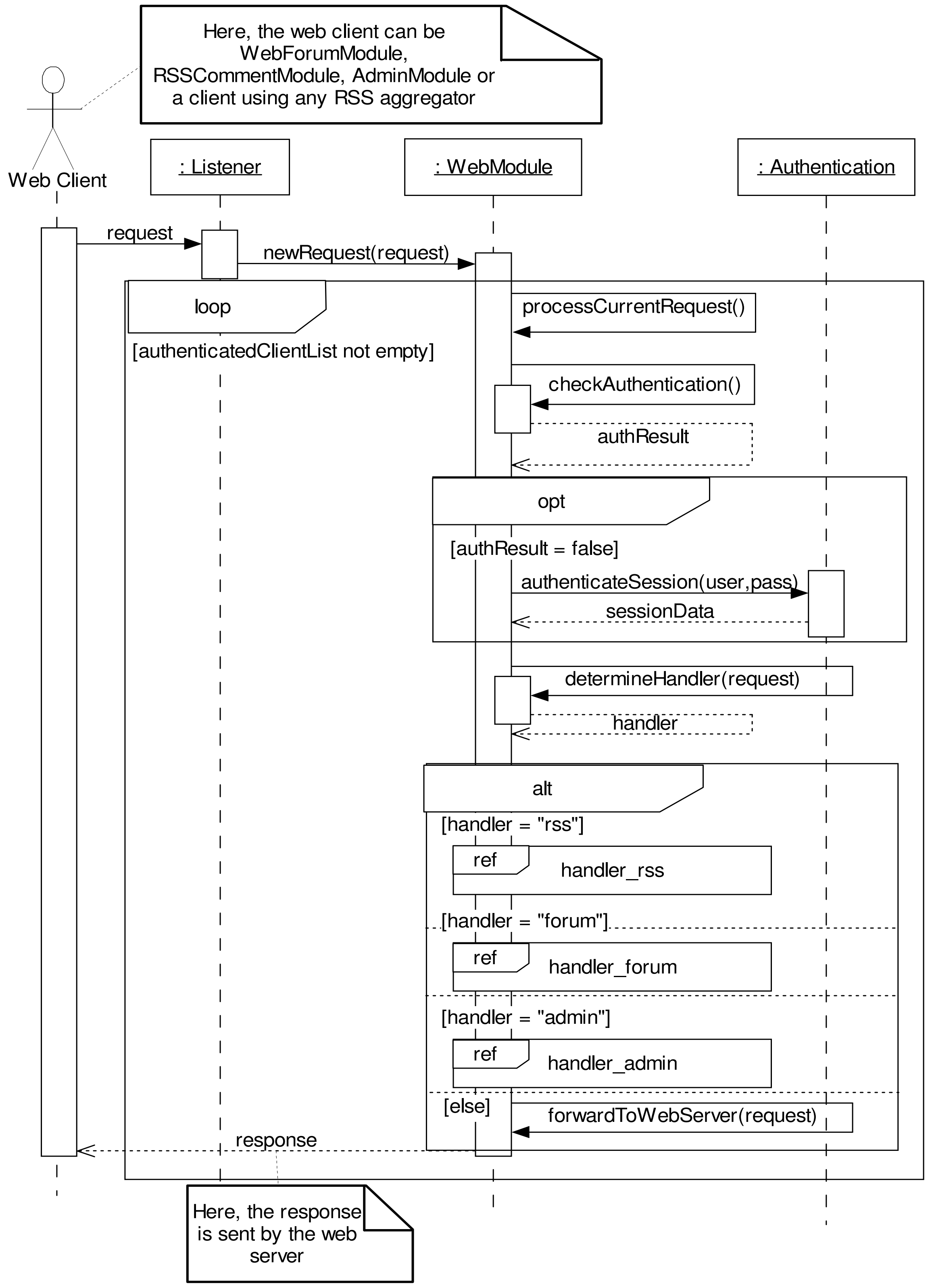

service_news
: NewsModule
: DatabaseConnector
: MessageConverter
: ServerTheonCore
isDBRequired()
doesRequireDB
opt
[doesRequireDB = true]
setParameters(uname, pass,host,port)
connect()
selectDB(dbName)
selectTable(tbName)
executeQuery(query)
queryResult
close()
isConversionRequired()
doesRequireConv
alt
[doesRequireConv = 1]
convertToCommon(msg)
commonMsg
[doesRequireConv = -1]
convertFromCommon(msg)
newsMsg
generateServiceMessage(request)
serviceMessage
sendServiceMessage(serviceMessage)
newServiceMessage(serviceMessage)
serviceResponse

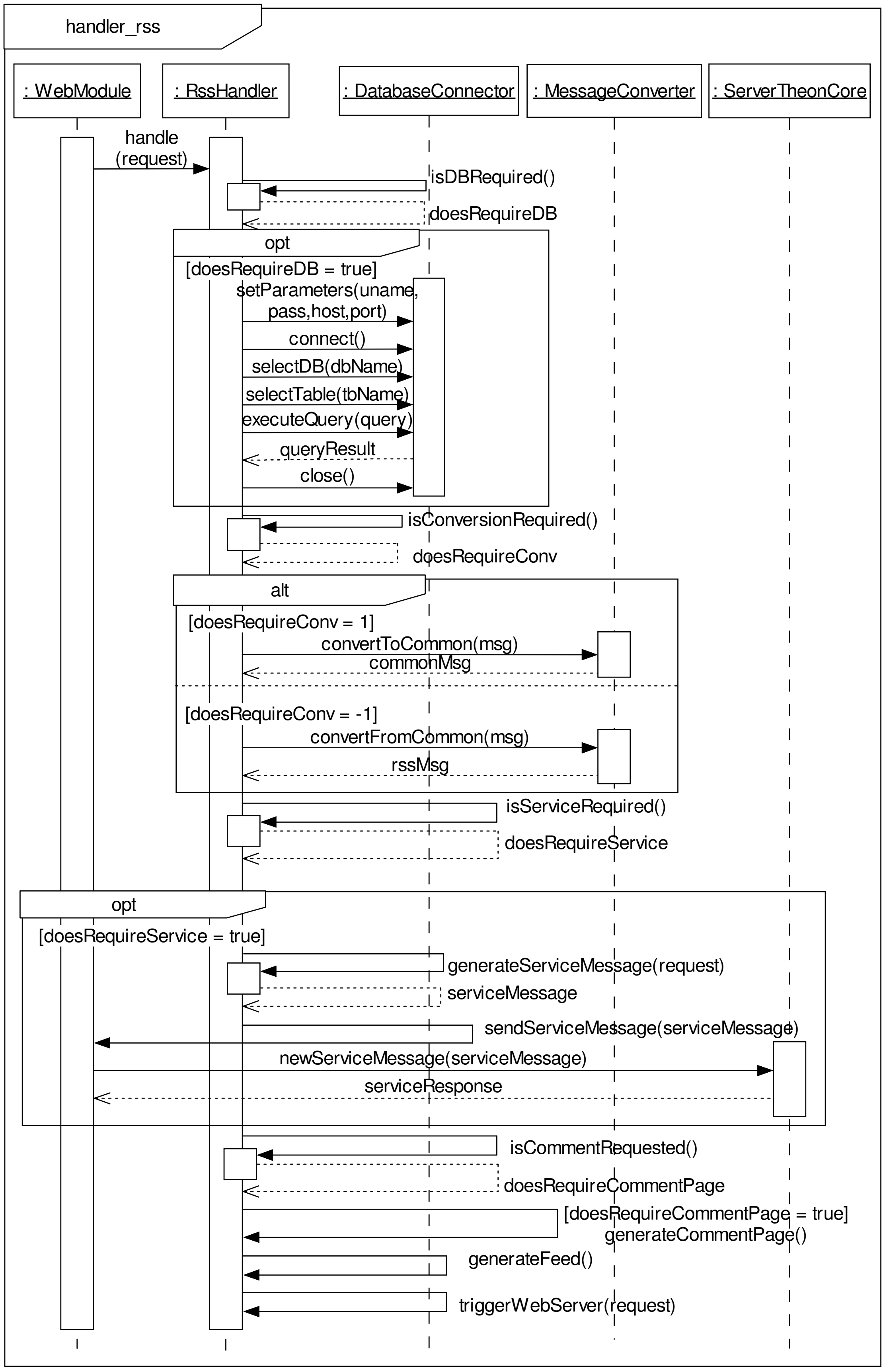
5.3.2 Mail Module

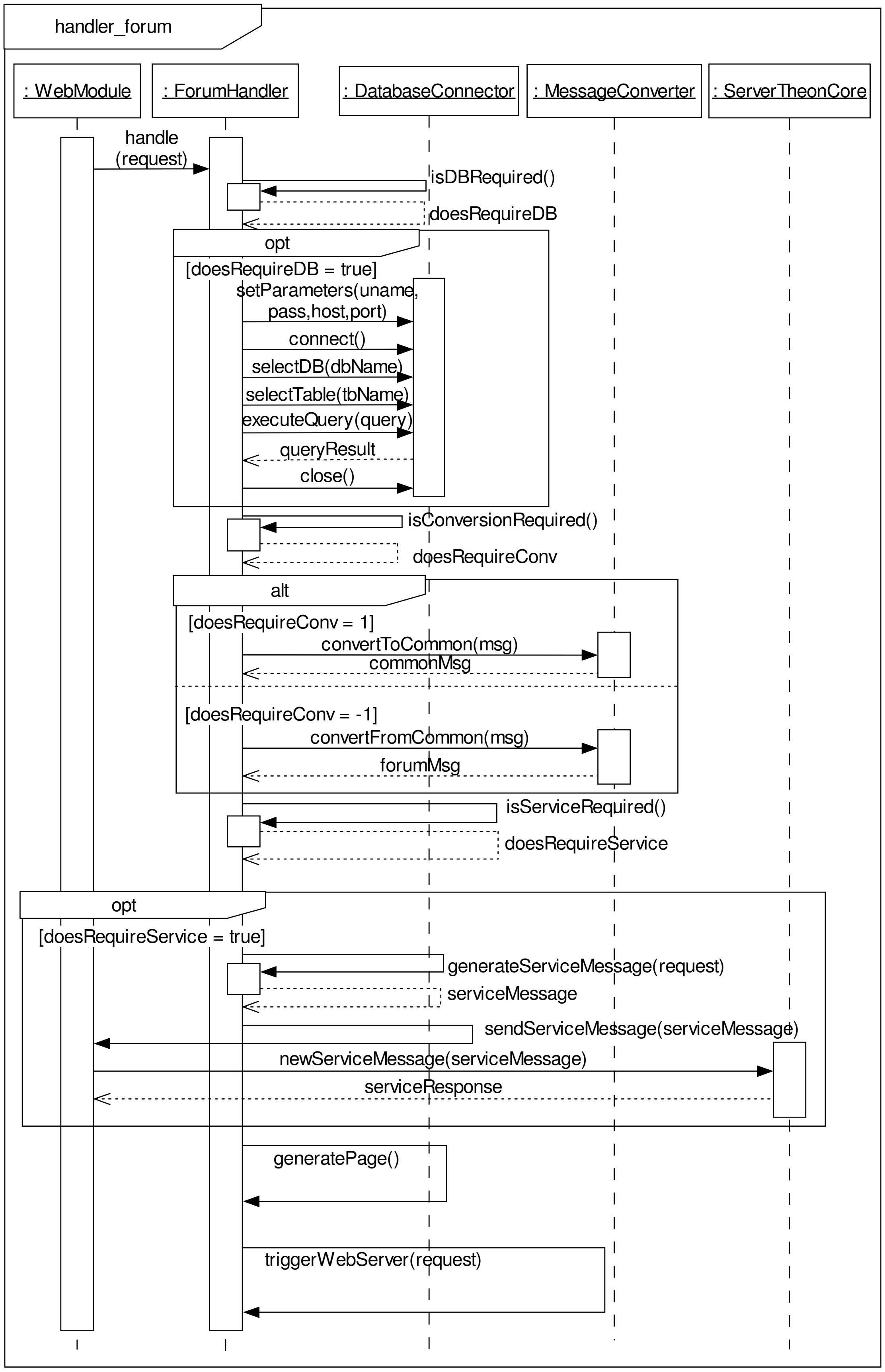


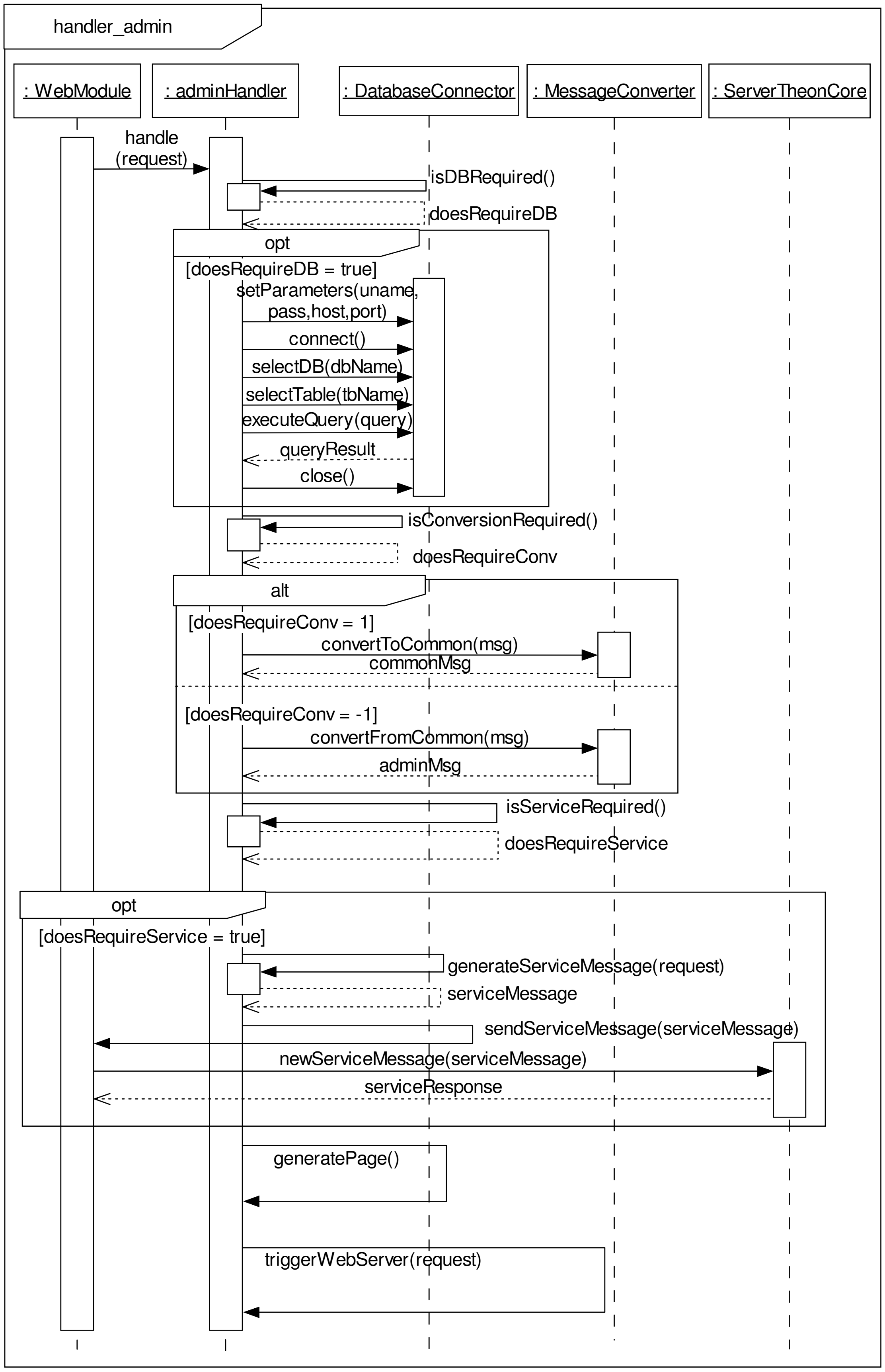

service_mail
: MailModule
: DatabaseConnector
: MessageConverter
: ServerTheonCore
isDBRequired()
doesRequireDB
opt
[doesRequireDB = true]
setParameters(uname, pass,host,port)
connect()
selectDB(dbName)
selectTable(tbName)
executeQuery(query)
queryResult
close()
isConversionRequired()
doesRequireConv
alt
[doesRequireConv = 1]
convertToCommon(msg)
commonMsg
[doesRequireConv = -1]
convertFromCommon(msg)
mailMsg
generateServiceMessage(request)
serviceMessage
sendServiceMessage(serviceMessage)
newServiceMessage(serviceMessage)
serviceResponse

5.3.3 Web Module

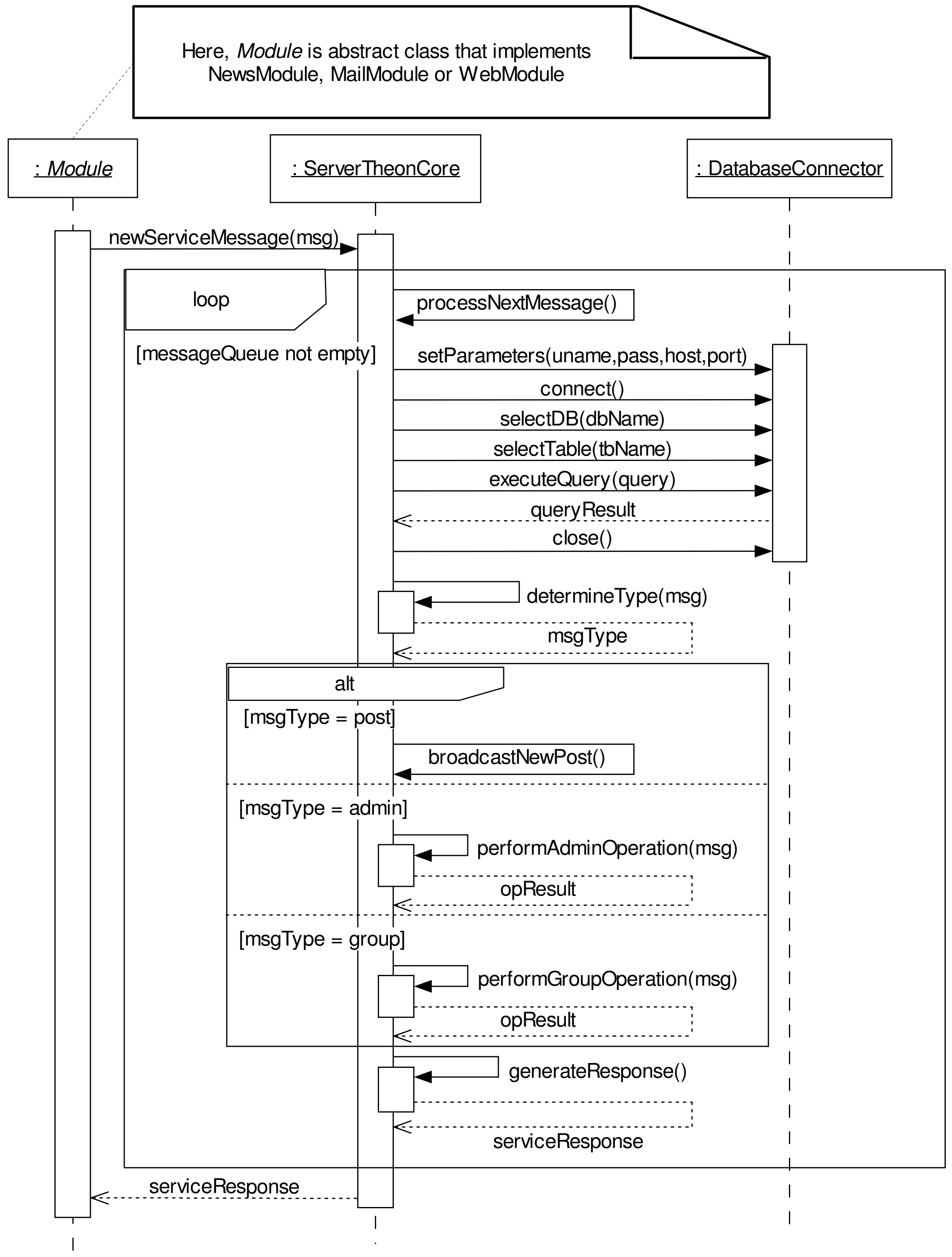



handler_rss
: WebModule
: RssHandler
: DatabaseConnector
: MessageConverter
: ServerTheonCore
handle (request)
isDBRequired()
doesRequireDB
opt
[doesRequireDB = true]
setParameters(uname, pass,host,port)
connect()
selectDB(dbName)
selectTable(tbName)
executeQuery(query)
queryResult
close()
isConversionRequired()
doesRequireConv
alt
[doesRequireConv = 1]
convertToCommon(msg)
commonMsg
[doesRequireConv = -1]
convertFromCommon(msg)
rssMsg
isServiceRequired()
doesRequireService
opt
[doesRequireService = true]
generateServiceMessage(request)
serviceMessage
sendServiceMessage(serviceMessage)
newServiceMessage(serviceMessage)
serviceResponse
isCommentRequested()
doesRequireCommentPage
[doesRequireCommentPage = true] generateCommentPage()
generateFeed()
triggerWebServer(request)


handler_forum
: WebModule
: ForumHandler
: DatabaseConnector
: MessageConverter
: ServerTheonCore
handle (request)
isDBRequired()
doesRequireDB
opt
[doesRequireDB = true]
setParameters(uname, pass,host,port)
connect()
selectDB(dbName)
selectTable(tbName)
executeQuery(query)
queryResult
close()
isConversionRequired()
doesRequireConv
alt
[doesRequireConv = 1]
convertToCommon(msg)
commonMsg
[doesRequireConv = -1]
convertFromCommon(msg)
forumMsg
isServiceRequired()
doesRequireService
opt
[doesRequireService = true]
generateServiceMessage(request)
serviceMessage
sendServiceMessage(serviceMessage)
newServiceMessage(serviceMessage)
serviceResponse
generatePage()
triggerWebServer(request)

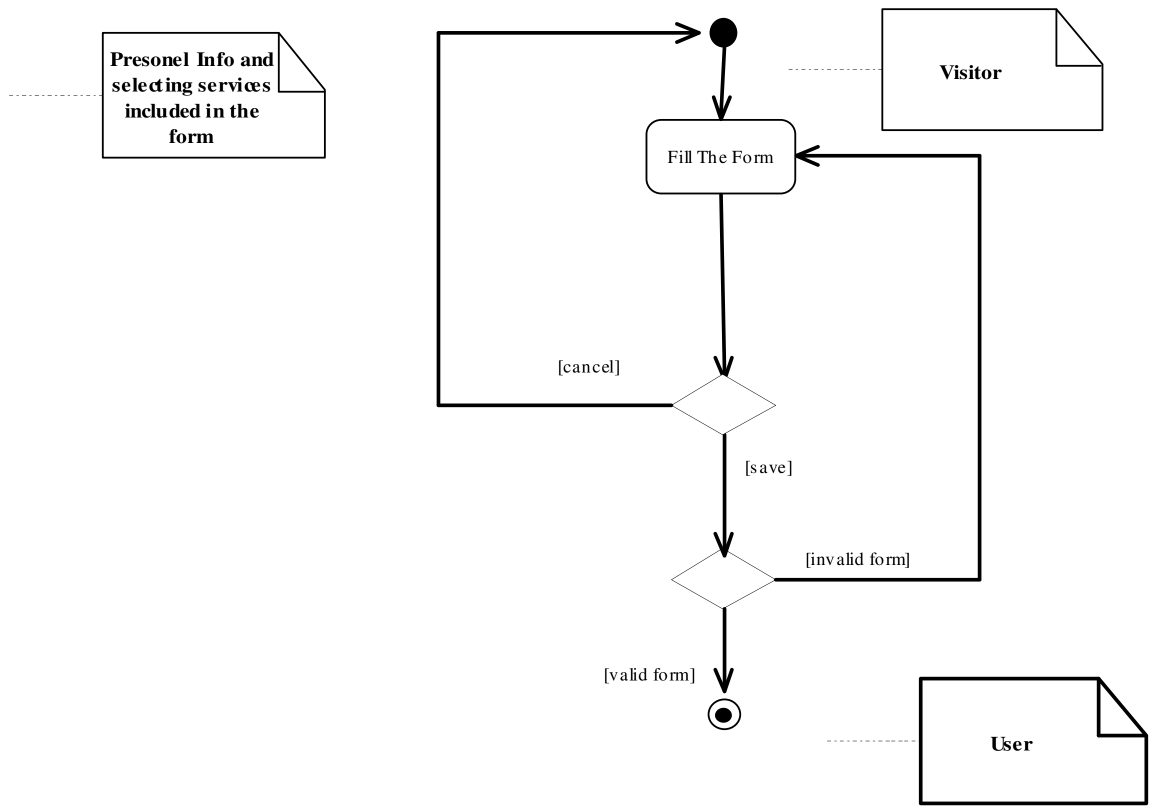

handler_admin
: WebModule
: adminHandler
: DatabaseConnector
: MessageConverter
: ServerTheonCore
handle (request)
isDBRequired()
doesRequireDB
opt
[doesRequireDB = true]
setParameters(uname, pass,host,port)
connect()
selectDB(dbName)
selectTable(tbName)
executeQuery(query)
queryResult
close()
isConversionRequired()
doesRequireConv
alt
[doesRequireConv = 1]
convertToCommon(msg)
commonMsg
[doesRequireConv = -1]
convertFromCommon(msg)
adminMsg
isServiceRequired()
doesRequireService
opt
[doesRequireService = true]
generateServiceMessage(request)
serviceMessage
sendServiceMessage(serviceMessage)
newServiceMessage(serviceMessage)
serviceResponse
generatePage()
triggerWebServer(request)

5.3.4 Core Service

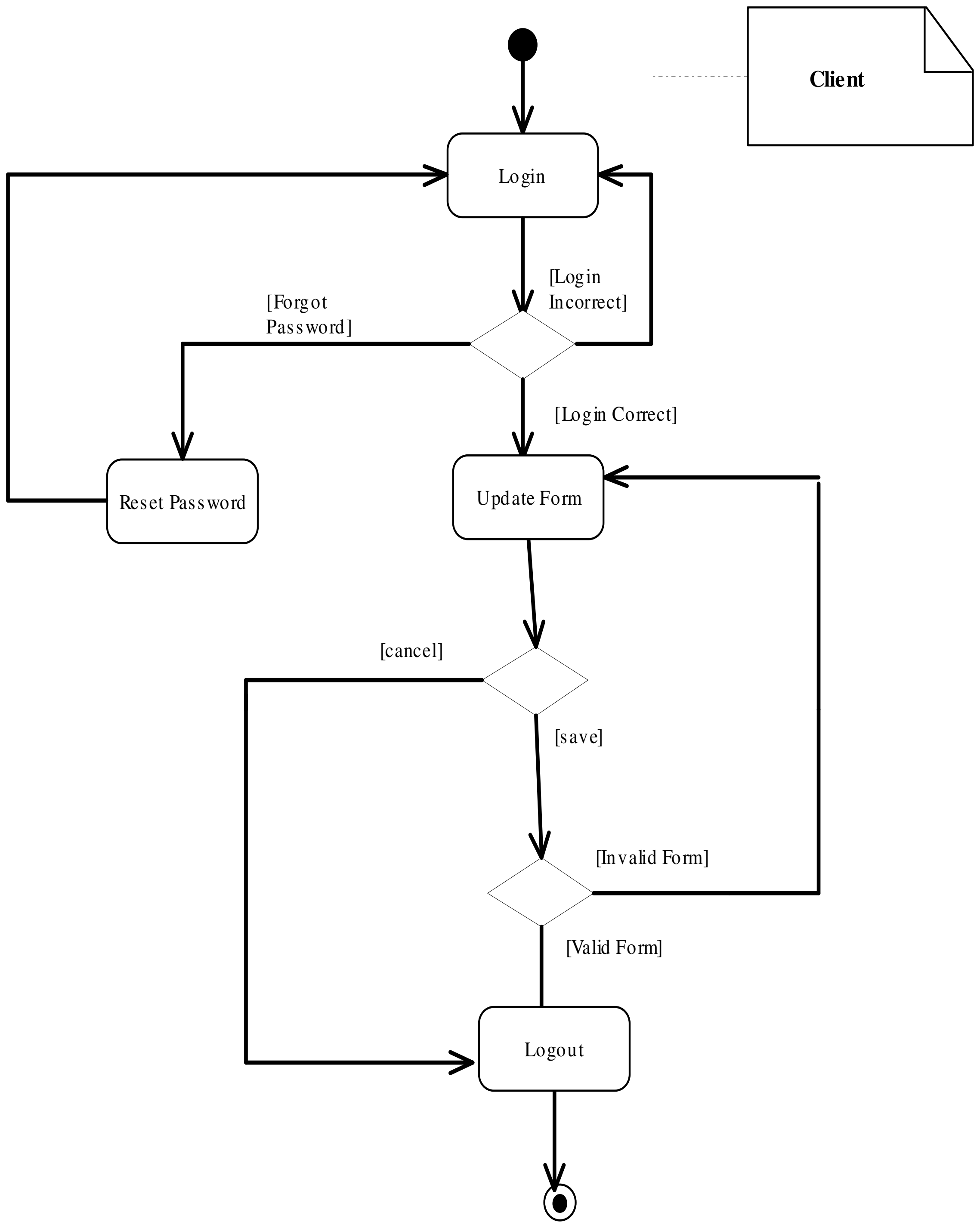


5.4 Activity Diagrams

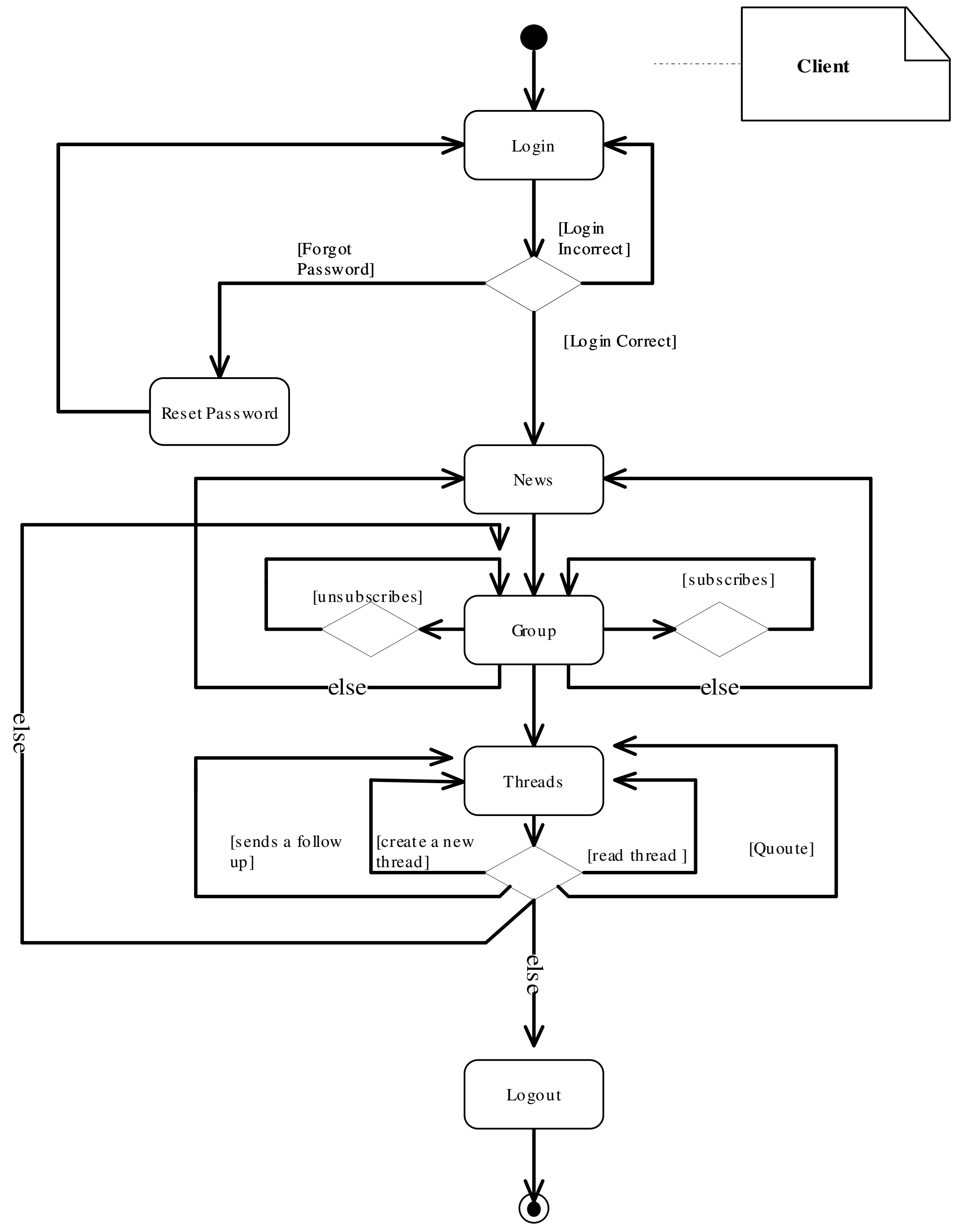
5.4.1 Registration



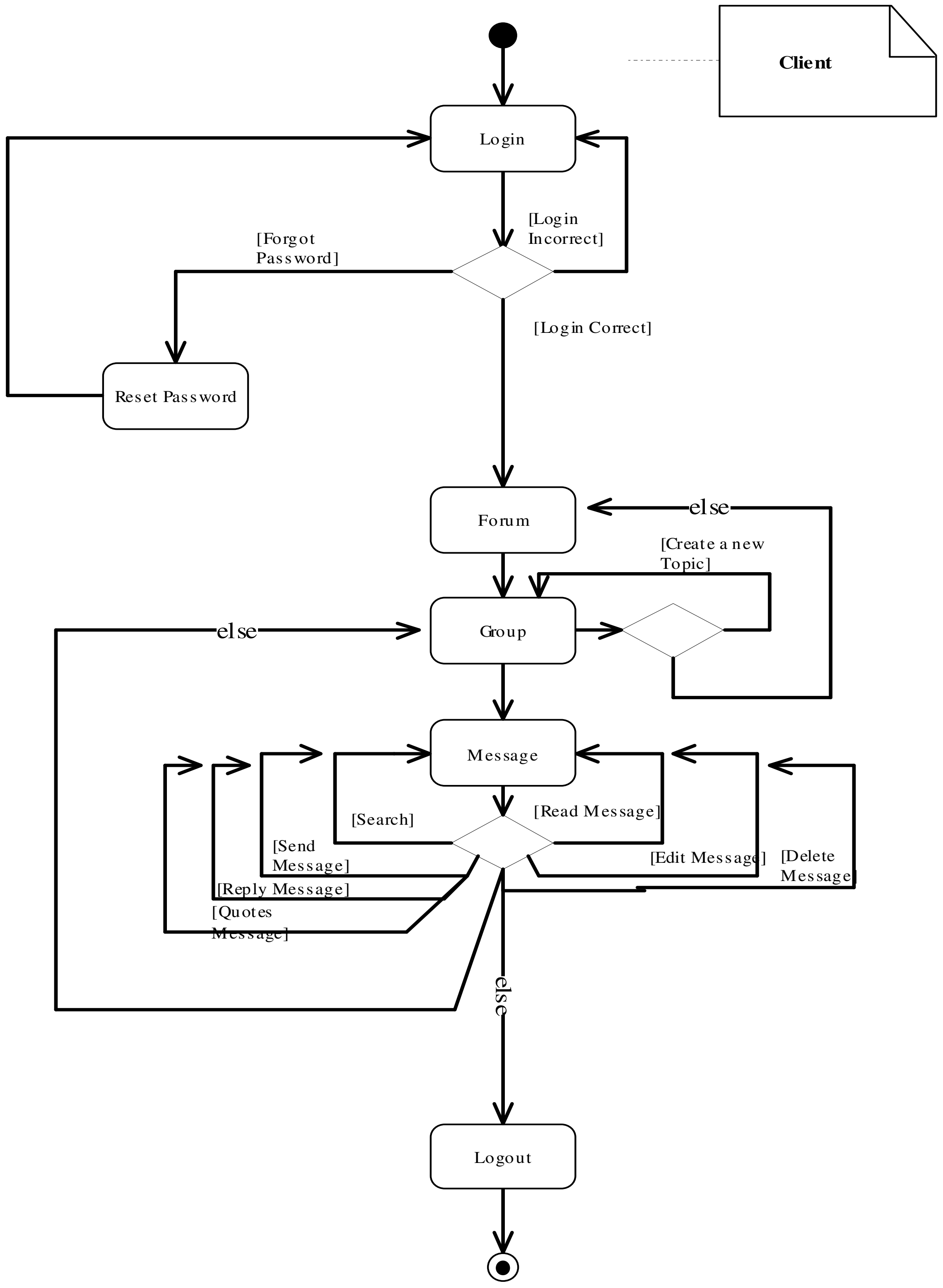
5.4.2 User Profile Update



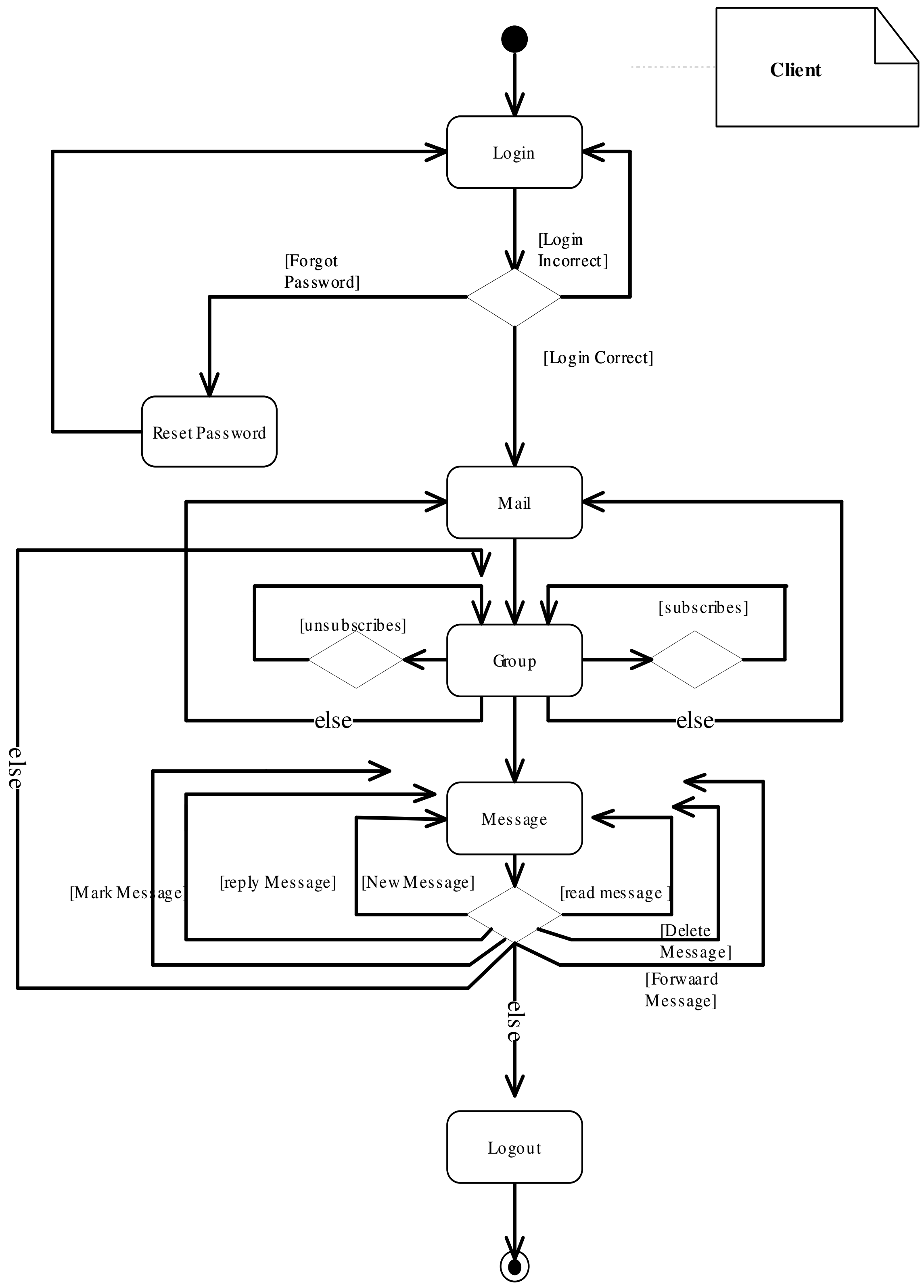
5.4.3 News Module



5.4.4 Web Module



5.4.5 Mail Module



6.0 User Interface Design

6.1 Main Page:

- Visitor can sign up using the “Sign Up” link.
- Client can view his/her profile via “View My Profile” link.
- Client can access a service page via corresponding service links (“TheonForum”, “TheonNews”).

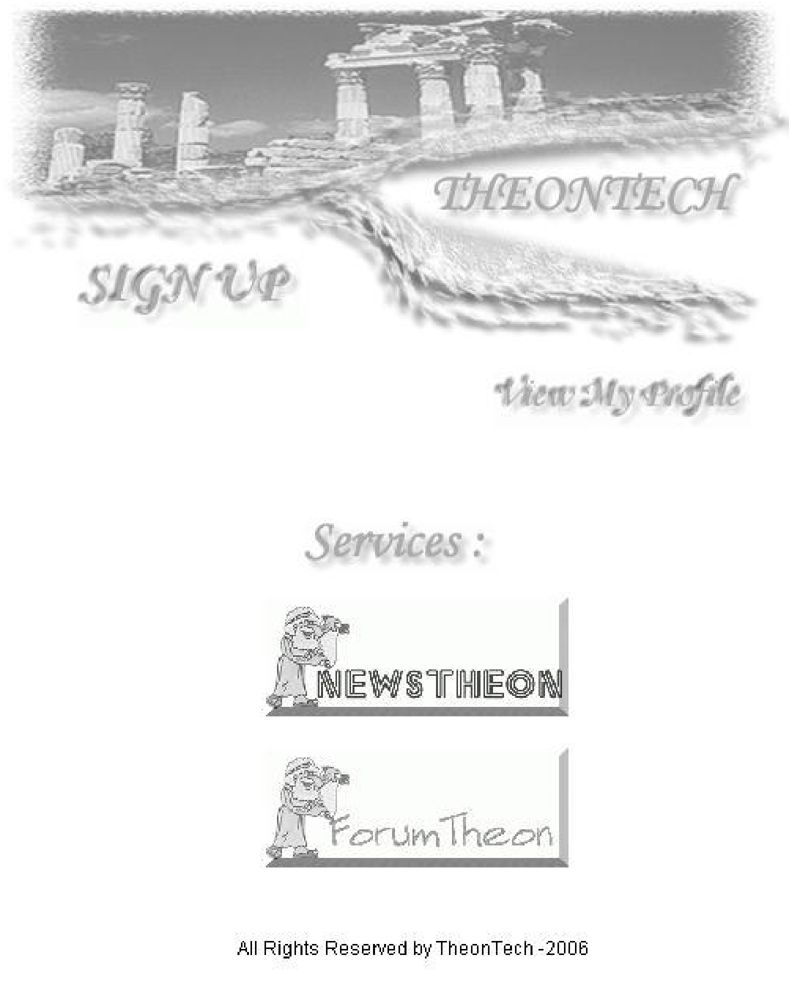


Figure: Main Page

New Registration Page:

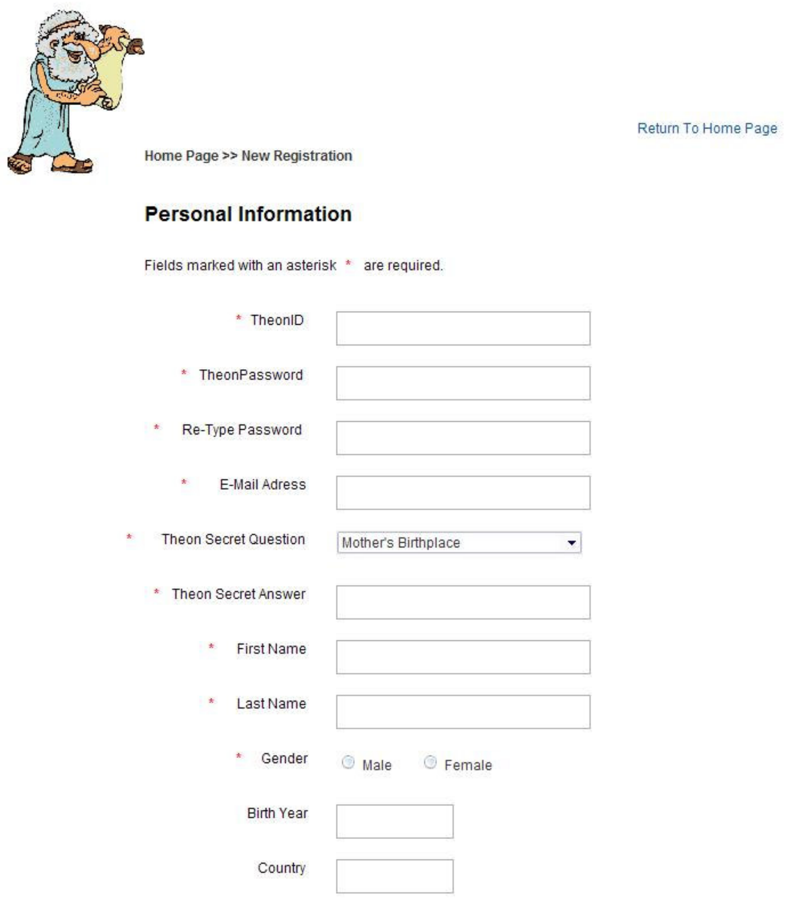
- Visitor fills the required fields of the form and selects the services which he/she wants via checkboxes.

-This form will include:

- * TheonID (TextBox)
- * TheonPassword (TextBox)
- E-mail Address (TextBox)
- * Theon Secret Question (ComboBox)
  -Mother's birthplace
  -Favorite food
  -Favorite teacher
  -Grandfather's occupation
- * Theon Secret Answer (TextBox)
- * First Name (TextBox)
- * Last Name (TextBox)
- * Gender (RadioButton)
  -Male
  -Female
- * Birth Year (TextBox)
- * Country (ComboBox)
- * State (ComboBox)
- * Zip Code (TextBox)
- Telephone (TextBox)
- Services
  -TheonForum (CheckBox)
  -TheonNews (CheckBox)
  -TheonRSS (CheckBox)
  -TheonMail (CheckBox)

- Visitor has to fill starred (*) entries for the form to be valid.

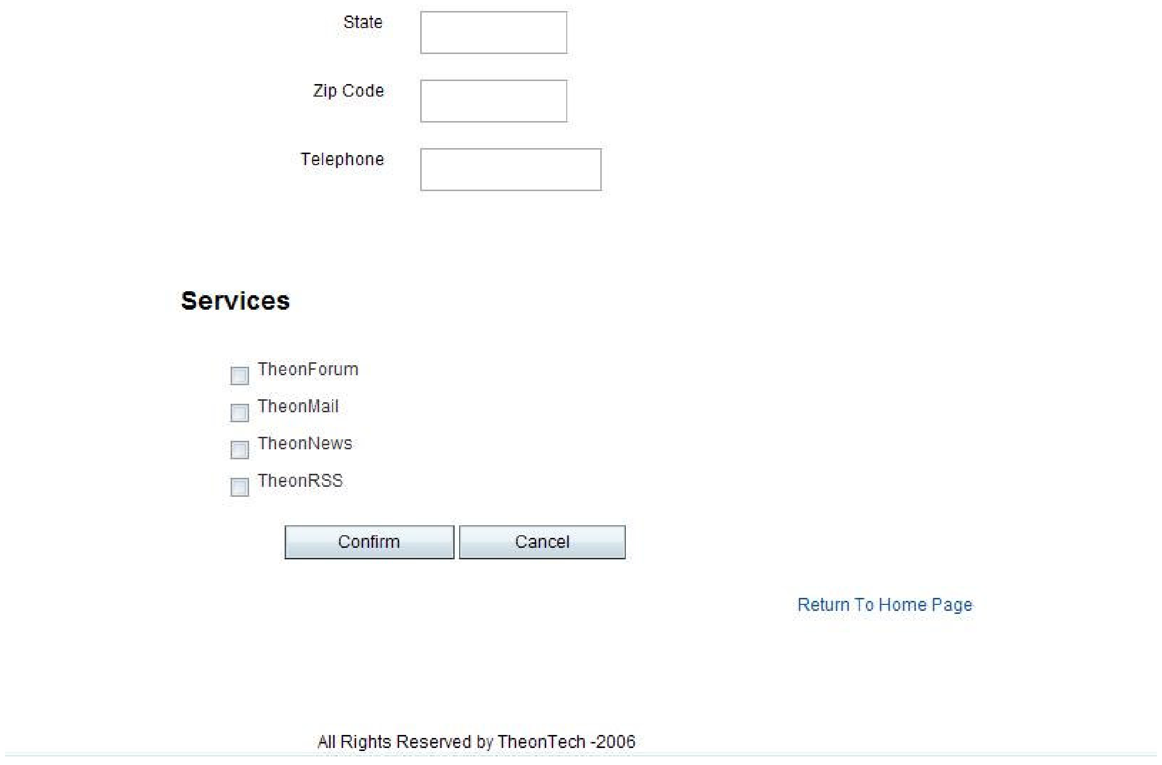
- If User does not fill required filled areas, system throws an error message such as "Error: Please, fill the "…" area!" or "Error: Please, fill the (*) areas" or "Error: Password Length must be …!" or "Error: Your E-mail address is not valid!" etc.
- At the bottom of the page, there is "I accept" and "Cancel" button if client presses former registration process will be ended and the client will be directed to the main page else if client presses latter one process will be killed and client will be directed to the main page.



Personal Information

Fields marked with an asterisk * are required.

* TheonID	
* TheonPassword	
* Re-Type Password	
* E-Mail Adress	
* Theon Secret Question	Mother's Birthplace
* Theon Secret Answer	
* First Name	
* Last Name	
* Gender	Male Female
Birth Year	
Country	



User Profile Login Page:

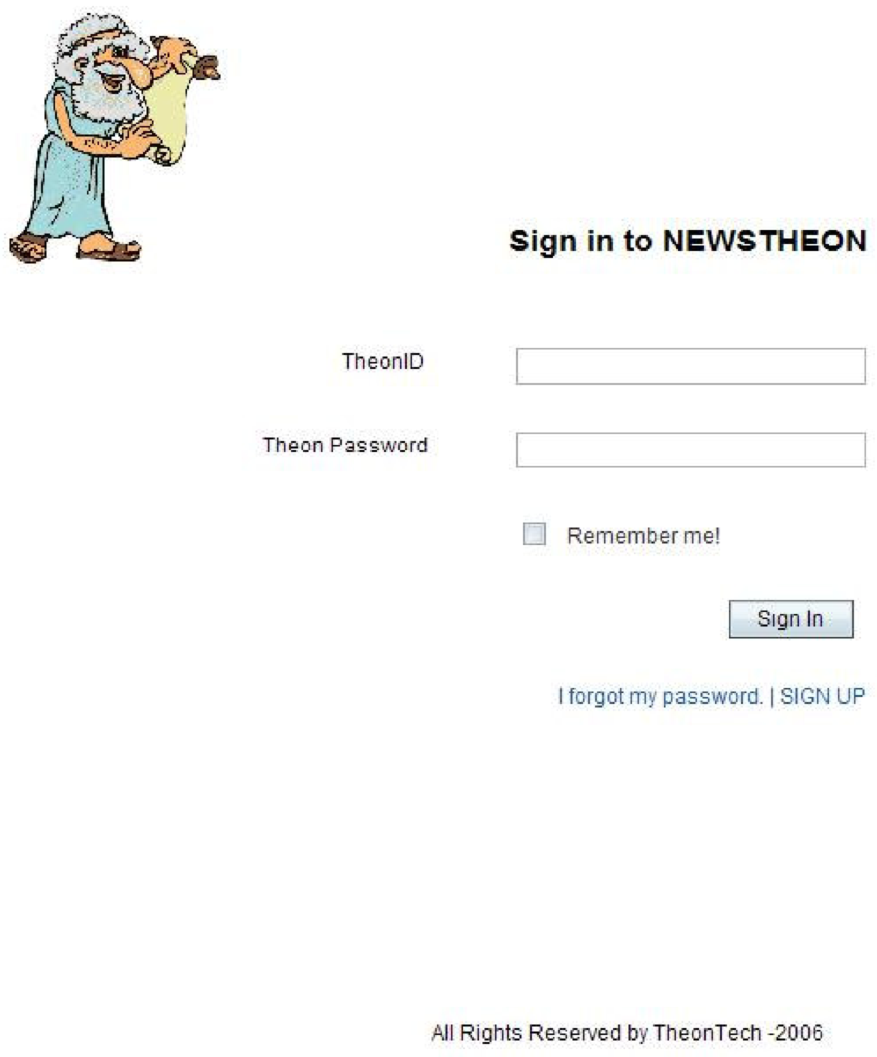
- Client can log into user profile page using the login group.

- TheonID (TextBox)
- TheonPassword (TextBox)
- Remind Me (CheckBox)
- Sign In (Button)
- I forgot my password! (HtmlLink)

- If user clicks "I forgot my password" link, user will be directed to the Reset Password Page.

- If user supplies wrong info, system throws an error message like "Invalid ID or Password!"

- If the information is valid, the client will be directed to the user profile page.

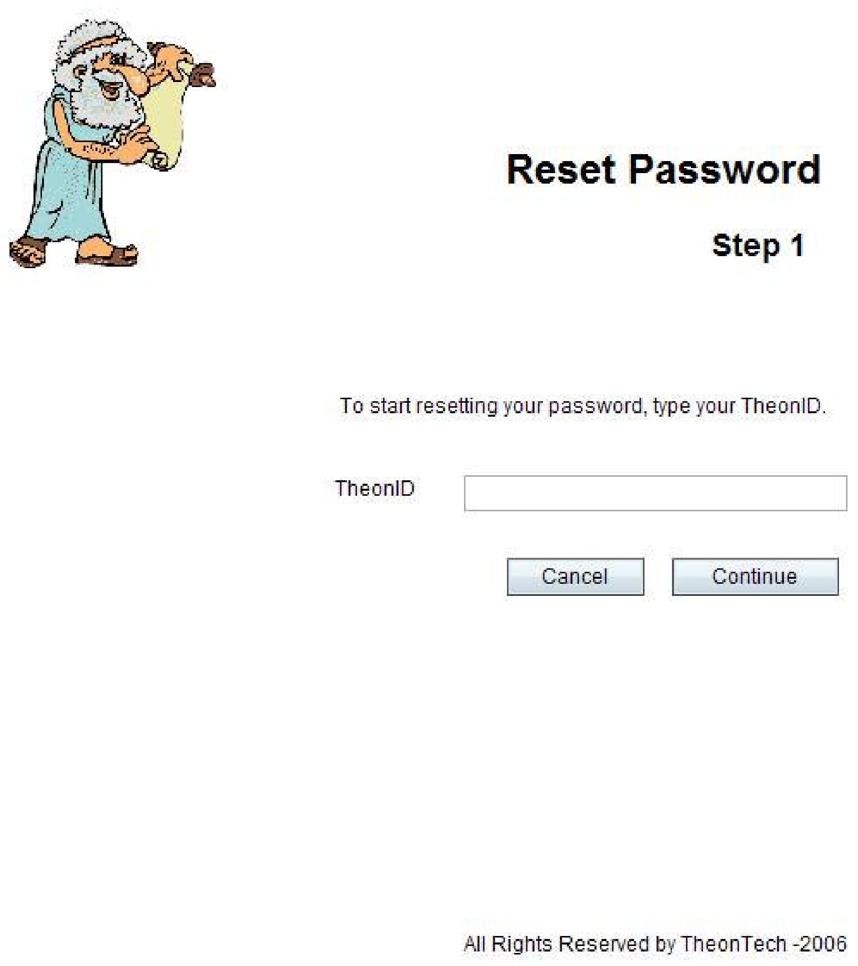

Sign in to NEWSTHEON
TheonID
Theon Password
Remember me!
Sign In
I forgot my password. | SIGN UP
All Rights Reserved by TheonTech -2006

Reset Password Page:

-User enters email.

- E-mail Address (TextBox)

- If user presses “Continue” button, user will be directed to the Reset Password Page2, else if user presses “Cancel” button user will be directed to the User Profile Login Page.
- If User does not give correct data or stay E-mail Address area as blank then system throws an error message “Error: E-mail Address are not valid!” or “Error: E-mail Address Area can not be null!”


Reset Password

Step 1

To start resetting your password, type your TheonID.

TheonID

All Rights Reserved by TheonTech -2006

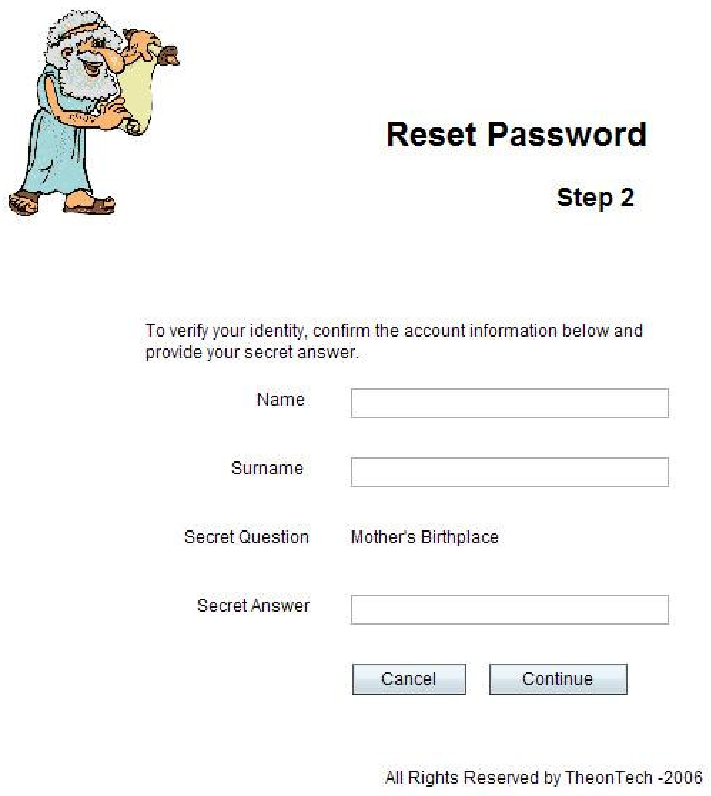
Reset Password Page2:

-There will be some account info questionnaire and secret question.

- Name (TextBox)
- Surname (TextBox)

- Secret Question (Label) (Read Only)
- Secret Answer (TextBox)

-If user presses "Continue" button, user will be directed to the Reset Password Page3, else if user presses "Cancel" button user will be directed to the User Profile Login Page.
-If user gives wrong info, system throws a message "Error: Name or Surname are not true for "…@.." email address!" or "Error: Secret Answer are not valid!"



To verify your identity, confirm the account information below and provide your secret answer.

Name	
Surname	
Secret Question	Mother's Birthplace
Secret Answer	

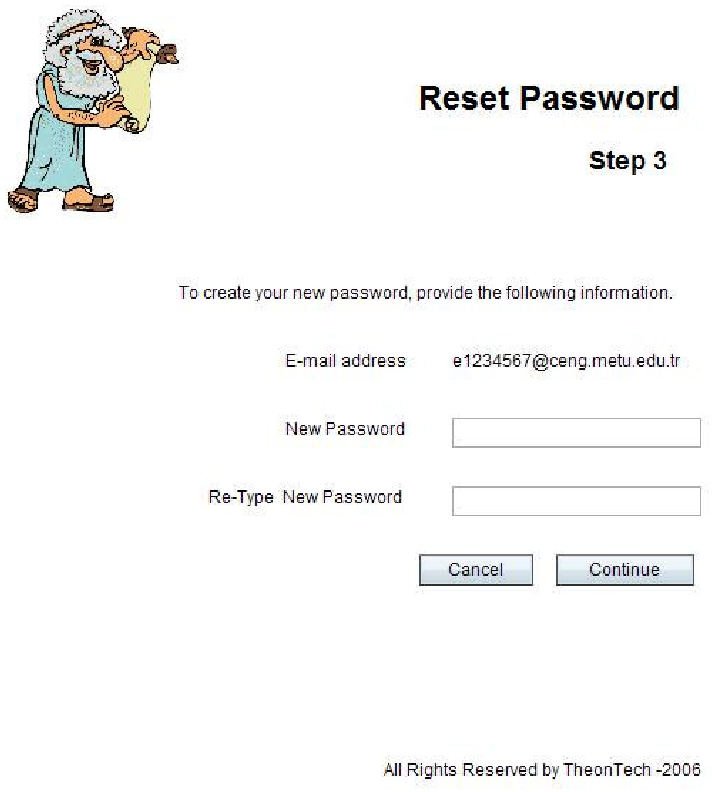
All Rights Reserved by TheonTech -2006

Reset Password Page3:

-User will see its email address and system will ask him/her to write a new password.

- E-mail Address (Label) (Read Only)
- New Password (TextBox)
- Retype Password (TextBox)

- If user presses “Continue” button, user will be directed to the User Profile Login Page, else if user presses “Cancel” button user will kill the process and be directed to the User Profile Login Page.
- If Retype Password != New Password then system throws a message “Error: Try Again. Your supplied passwords are not same!”



Reset Password

Step 3

To create your new password, provide the following information.

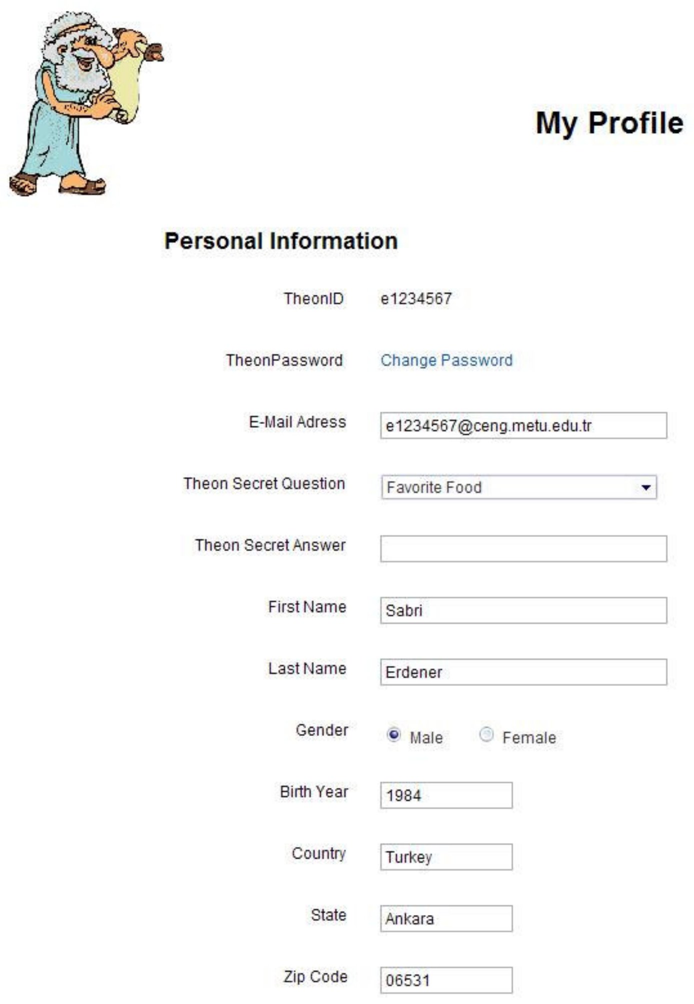
E-mail address	e1234567@ceng.metu.edu.tr
New Password	
Re-Type New Password	

All Rights Reserved by TheonTech -2006

User Profile Page:

- Client can see/update his/her personal information via a form. (As we stated above (New Registration Page))
- At the bottom of the form, there is “Save” button to submit the information on the form.

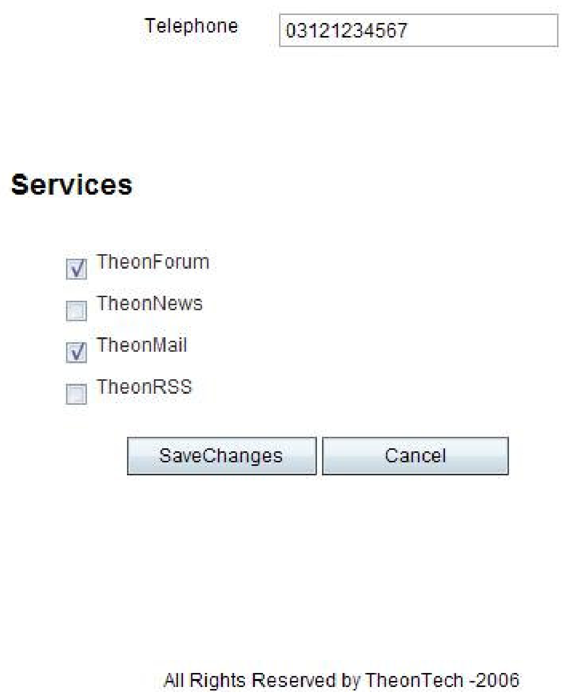
- If User gives invalid data, system throws messages as I stated above.
- Client logs out via “Logout” button and directed to the Main Page.



My Profile

Personal Information

TheonID	e1234567
TheonPassword	Change Password
E-Mail Adress	e1234567@ceng.metu.edu.tr
Theon Secret Question	Favorite Food
Theon Secret Answer	
First Name	Sabri
Last Name	Erdener
Gender	[✓ Male] Female
Birth Year	1984
Country	Turkey
State	Ankara
Zip Code	06531


Telephone: 03121234567

Services

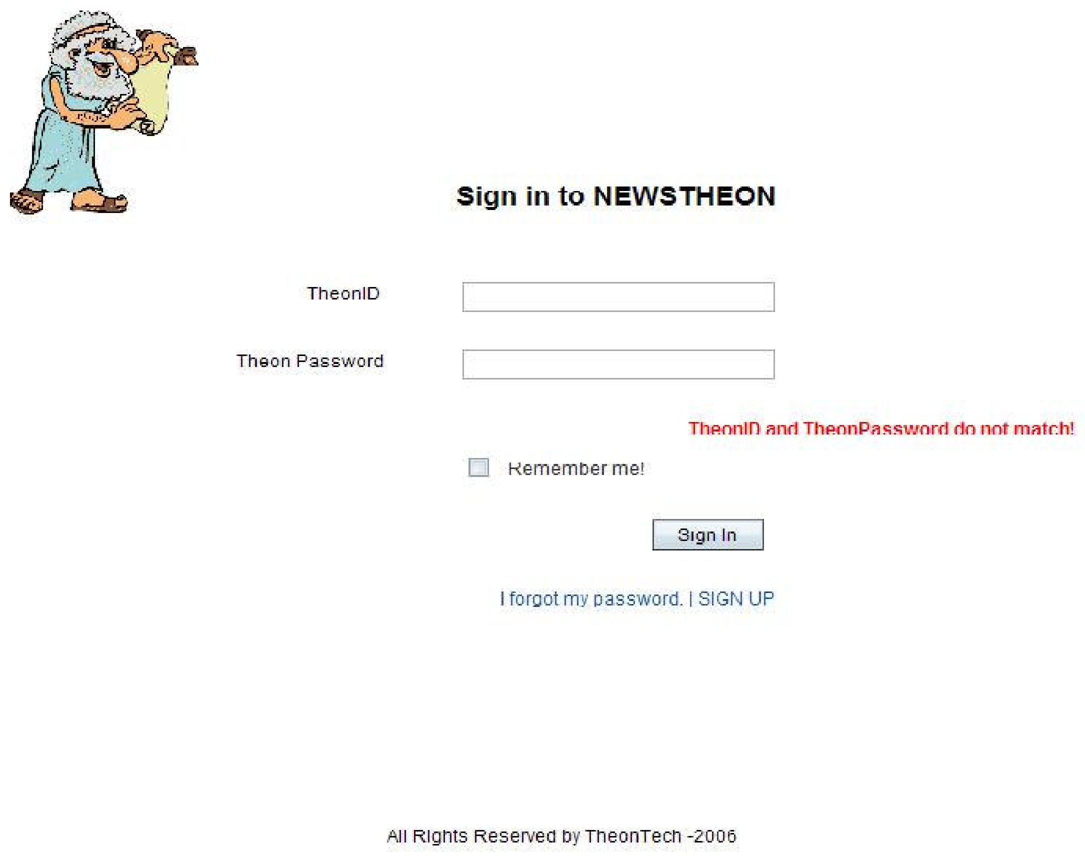
[✓] TheonForum
[ ] TheonNews
[✓] TheonMail
[ ] TheonRSS

Forum Login Page:

Client can log into forum module using the login group.

- TheonID (TextBox)
- TheonPassword (TextBox)
- Remind Me (CheckBox)
- Sign In (Button)
- I forgot my password! (HtmlLink)

- If user clicks "I forgot my password" link, user will be directed to the Reset Password Page.
- If user supplies wrong info, system throws an error message like "Invalid ID or Password!"
- If the information is valid, the client/admin will be directed to the forum main page.


Sign in to NEWSTHEON

TheonID

Theon Password

TheonID and TheonPassword do not match!

Remember me!

All Rights Reserved by TheonTech -2006

Forum Main Page:

- At the top of the page, UserName or AdminName showed.

- UserName or AdminName (Label)

- Client can access messages in any group via clicking the link of the subject of the message. After clicking, <GroupName> Page opens.

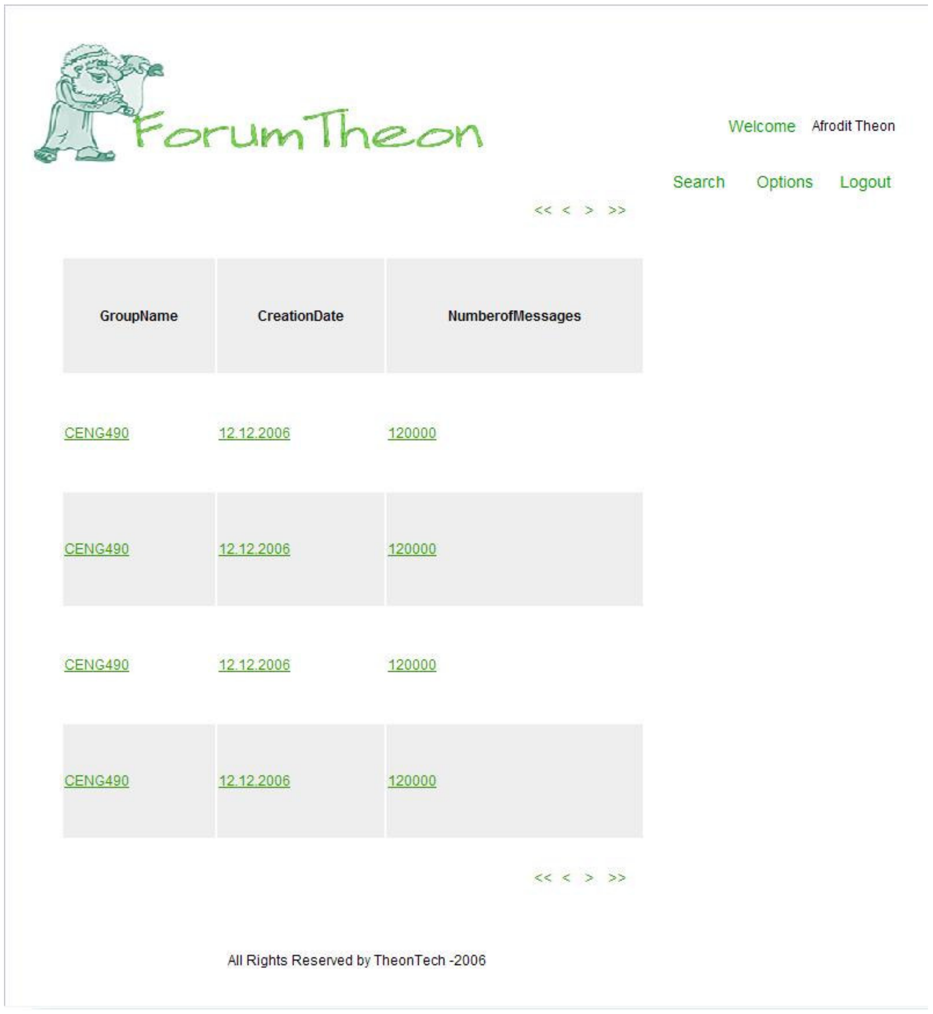
- Admin can lock any topic via the "Lock Topic" button.

- Admin can move any topic via "Move Topic" button.

- Client can make a search using "Search" button which is on the bottom of the page. Client can search the keyword in a specific group/topic or can do general search.

When user clicks search button, user will be directed to the Search Main Page.

- Client logs out via “Logout” button.



GroupName	CreationDate	NumberofMessages
CENG490	12.12.2006	120000
CENG490	12.12.2006	120000
CENG490	12.12.2006	120000
CENG490	12.12.2006	120000

<GroupName> Page:

- Messages are listed in a GridView and this GridView columns are:
    - MessageTopic (link)
    - NumberofMessages (label)

- FirstMessageClientID (label)
- LastMessageRepliedDate (label)

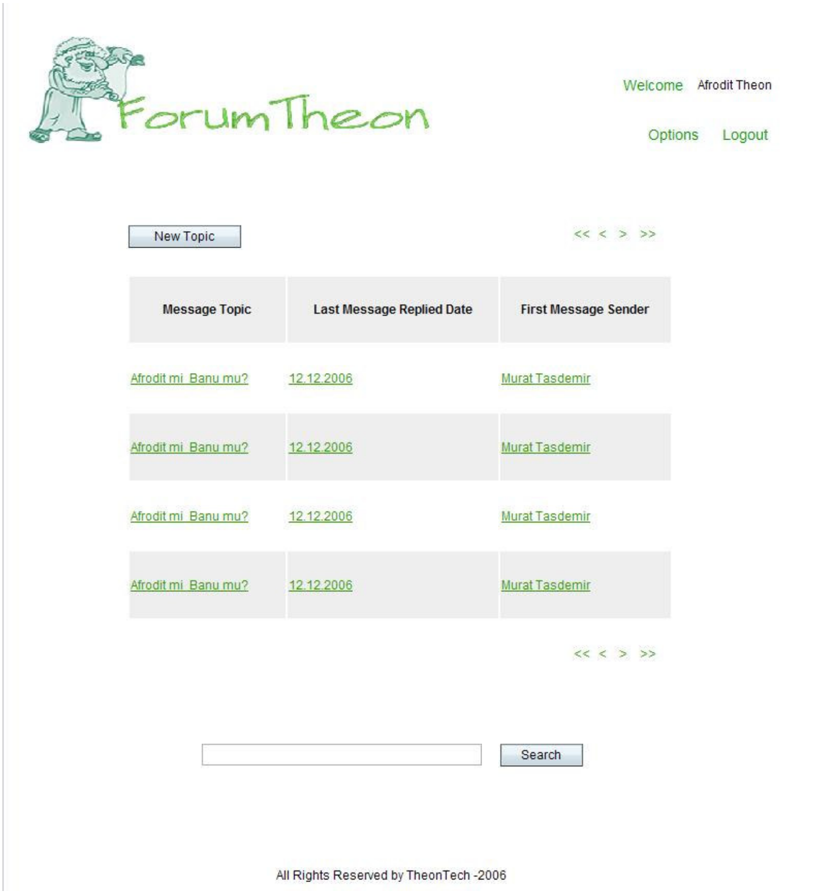
- At the bottom right of the page there are “Search” button.

- Search (button)
- Search Key (TextBox)

- GridView will be paged. (style: << < > >>).

-At the top left of GridView there is a NewMessage button. When User clicks this button, user will be directed to the Create New Message Page.

-If User clicks MessageTopic link, user will be directed to the <Message Topic> Page.



Message Topic	Last Message Replied Date	First Message Sender
Afrodit mi Banu mu?	12.12.2006	Murat Tasdemir
Afrodit mi Banu mu?	12.12.2006	Murat Tasdemir
Afrodit mi Banu mu?	12.12.2006	Murat Tasdemir
Afrodit mi Banu mu?	12.12.2006	Murat Tasdemir

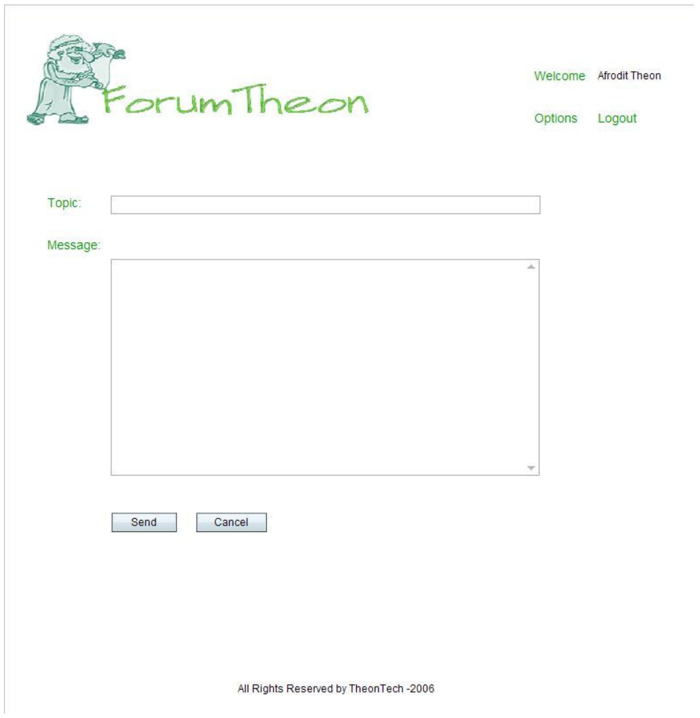
Create New Message Page:

-User can create a new topic and this page includes:

- Topic (TextBox)(if this page opened after "Reply" link this entry will automatically filled such as: Re: Replied Topic)
- Message (MultiLineTextBox) (if this page opened after "Quote" link this entry will automatically filled with Parent Message) (if this page opened after "Edit" link this entry will automatically filled with Original Message)
- Send (Button)
- Cancel (Button)

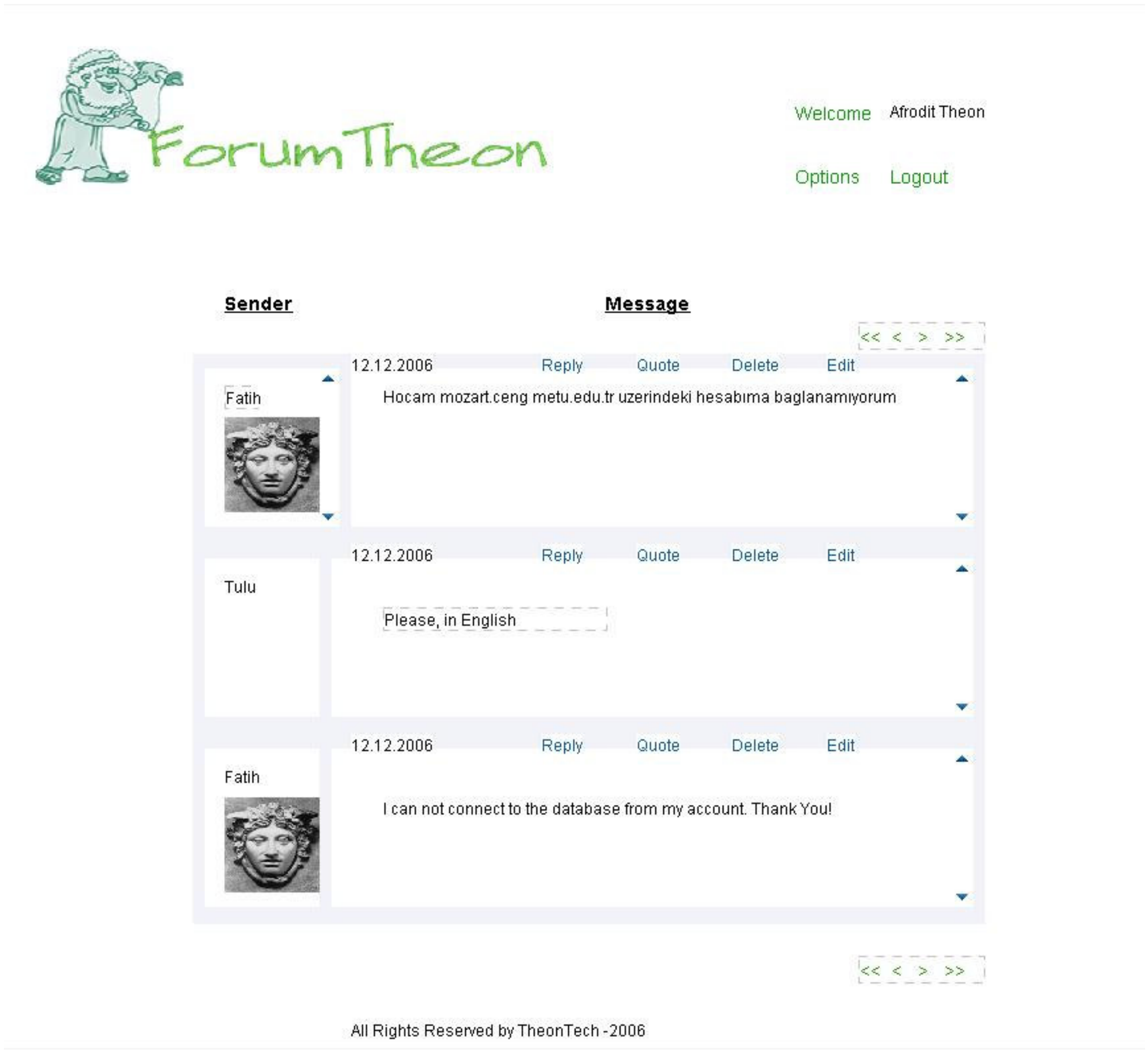
-If User clicks "Send" button, message will send to the group and user will be directed to the <GroupName> Page.

-If User clicks "Cancel" button, message will not send to the group and user will be directed to the <GroupName> Page.



<MessageTopic> Page:

- This page includes message and their replies as a GridView. This GridView’s colums are:
  - Sender
  - Message
  - MessageDate
- GridView will be paged. (style: << < > >>).
- GridView Message column’s every row has:
  - Sent datetime (label)
  - Reply (link)
  - Quote (link)
  - Delete (link)
  - Edit (link)
- When User clicks “Reply” link, user will be directed to the Create New Message Page.
- When User clicks “Quote” link, parent message will be quoted and user will be directed to the Create New Message Page.
- When User clicks “Delete” link, user deletes his/her own message and be directed to the <Message Topic> Page.
- When User clicks “Edit” link, user will be directed to the Create New Message Page.
- At the bottom right of the page there are “Search” button.
  - Search (button)
  - Search Key (TextBox)



Welcome Afrodit Theon

Options Logout

Sender	Message
Fatih	12.12.2006 Reply Quote Delete Edit Hocam mozart.ceng metu.edu.tr uzerindeki hesabıma baglanamıyorum
Tulu	12.12.2006 Reply Quote Delete Edit Please, in English
Fatih	12.12.2006 Reply Quote Delete Edit I can not connect to the database from my account. Thank You!

All Rights Reserved by TheonTech - 2006

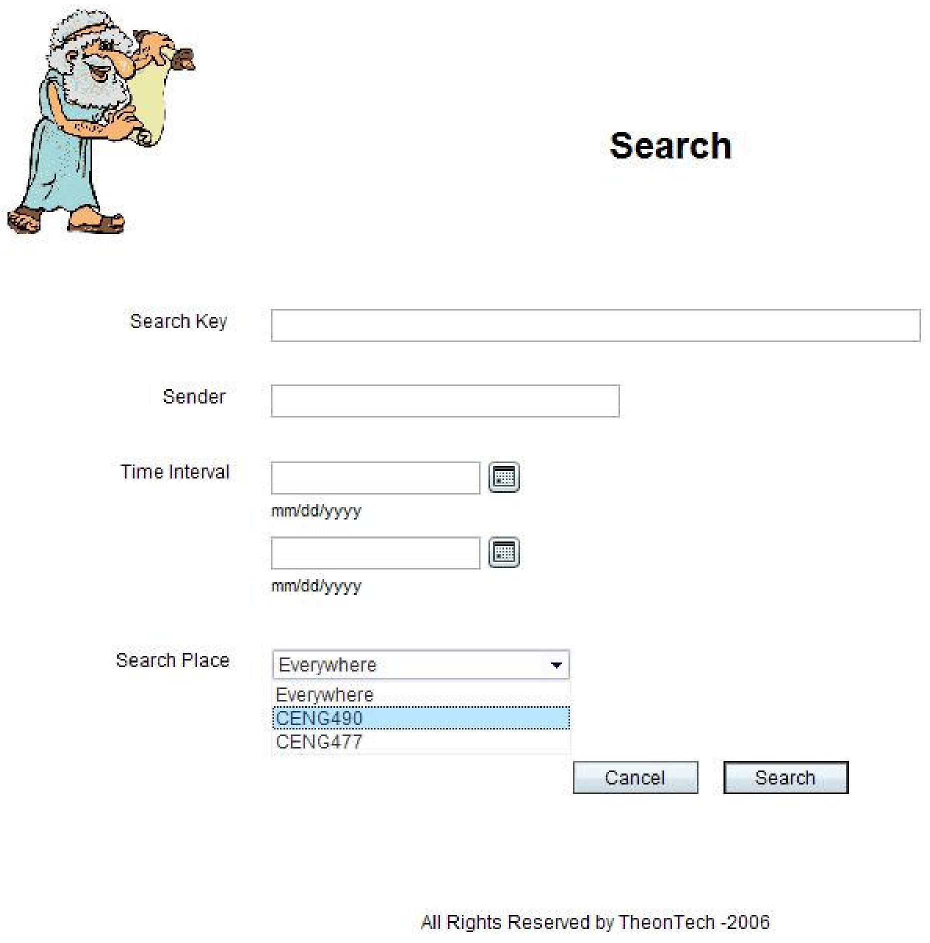
Search Main Page:

- Search key (TextBox) (you can put AND,OR between search words)
- Search according to Sender (TextBox)
- Search according to Date (TextBox)
- Search Place (MultiLineTextBox)
- Search (Button)
- Cancel (Button)

- If User clicks the “Search” Button, search results are shown in Search Result Page.
- If User clicks the “Cancel” Button, user will be directed to the ParentPage.

Search Result Page:

- Search Results are shown in a GridView according to Search parameter, GridView colums are set.



Search

Search Key	
Sender	
Time Interval	mm/dd/yyyy
	mm/dd/yyyy
Search Place	Everywhere

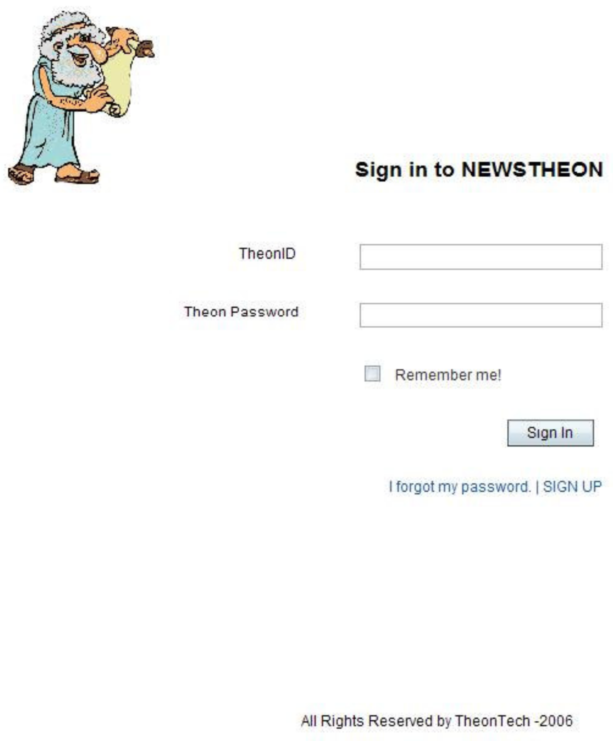
All Rights Reserved by TheonTech -2006

News Login Page:

Client can log into news module using the login group.

- TheonID (TextBox)
- TheonPassword (TextBox)
- Remind Me (CheckBox)
- Sign In (Button)
- I forgot my password! (HtmlLink)

- If user clicks "I forgot my password" link, user will be directed to the Reset Password Page.
- If user supplies wrong info, system throws an error message like "Invalid ID or Password!"
- If the information is valid, the client will be directed to the news main page.


Sign in to NEWSTHEON
TheonID
Theon Password
Remember me!
Sign In
I forgot my password. | SIGN UP
All Rights Reserved by TheonTech -2006

News Main Page:

- Client can (un)subscribe to a group or groups which she/he wants to follow via “Options” button which leads to a list of groups which can be selected via checkboxes. Client can also see the description under the group name and number of messages in the groups near the group name. At the bottom of the group list, there is “Save” button which saves the client preferences. **(Option Page)**
- There is a ‘selected groups’ menu on the left of the page which includes links to the groups. Client can access messages via clicking these links. When user clicks this links, <GroupName> Page opens.
- Clientlogs out via “Logout” button.

<GroupName> Page:

-Header of this page Message Date, Subject and Author columns exist. Besides, number of pages will be shown.
- Every row of this page includes message links. Client reads messages in groups which he/she subscribed via clicking the link of the subject of the message. When user clicks this button, message details will shown under this page in <MessageSubject> Page.
- Client creates a new thread in a group which he/she subscribed via “New Thread” which is on the top of the page. When User clicks this button, Create New Message Page is seen under this page.

Create New Message Page:

- Subject (TextBox)(if this page opened after "Reply" link this entry will automatically filled such as: Re: Replied Topic)
- CC (TextBox) (if this page opened after "Reply" link this entry will automatically filled)
- ID of Sender (Label) (if this page opened after "Reply" link this entry will automatically filled)
- MessageArea (MultiLineTextBox)
- Quote (Button)
- Post (Button)

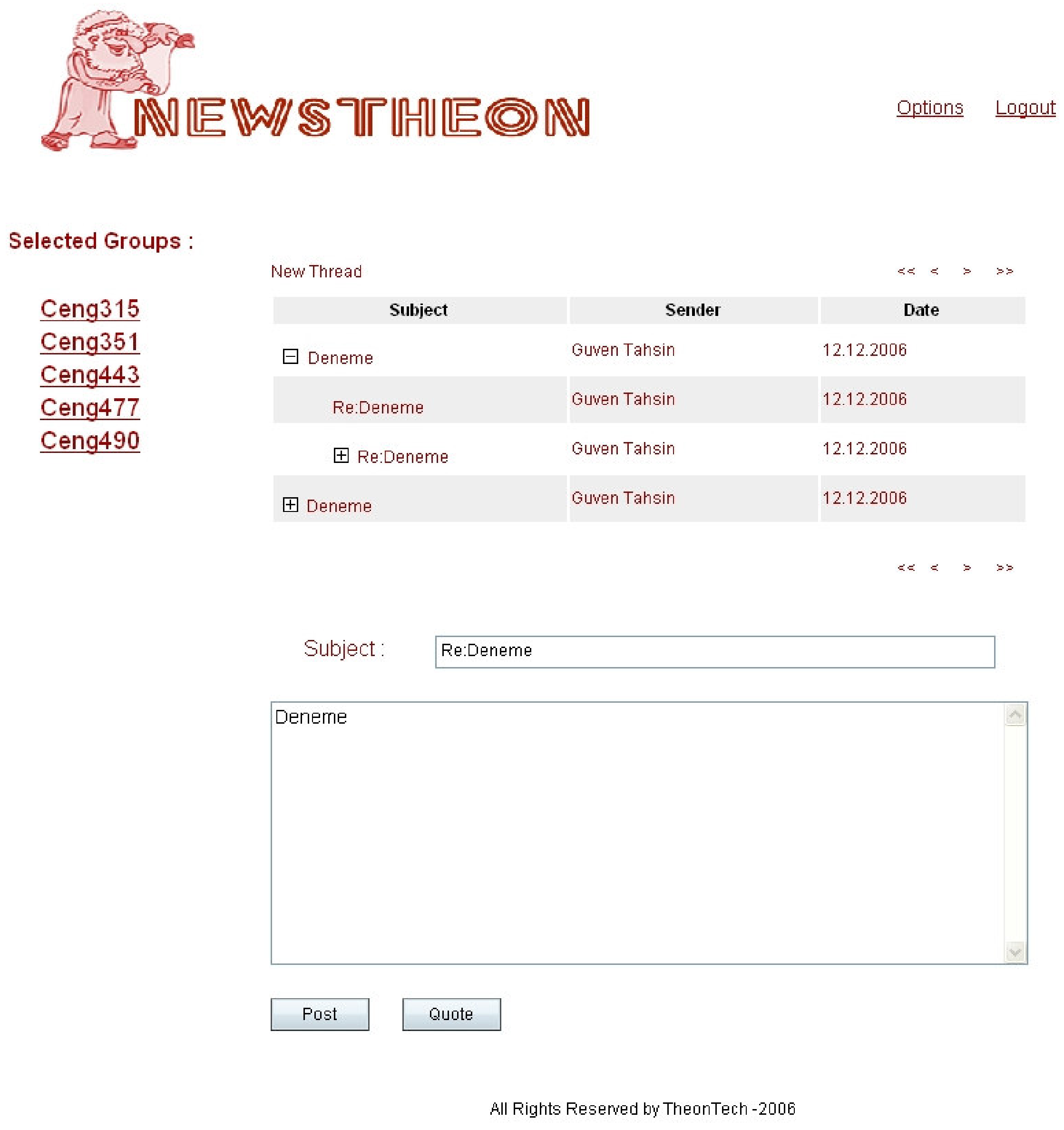
-If user clicks "Quote" button, parent message will be quoted to the MessageArea.
-If user clicks "Post" button, message will send to the group and user will notified via a message like "To see your message please click "Refresh" button".

<Message Subject> Page:

- User will see message details.

- Subject (label)
- From (label)
- Date (label)

- User can reply a message with clicking the link "Reply". When user clicks this page, Create New Message Page will appear instead of <Message Subject> Page.



Subject	Sender	Date
⊟ Deneme	Guven Tahsin	12.12.2006
Re:Deneme	Guven Tahsin	12.12.2006
⊞ Re:Deneme	Guven Tahsin	12.12.2006
⊞ Deneme	Guven Tahsin	12.12.2006

Re:Deneme

Deneme

All Rights Reserved by TheonTech -2006

Admin Login Page:

- Admin can log into system using the login group (AdminID, AdminPassword, and Remind Me).

- If admin supplies wrong info, system sends an error message like “Invalid ID or Password!”

- If the information is valid, the admin will be directed to the admin main page.

Admin Main Page:

- If Admin wants to edit/delete a user, firstly admin search the user and search results are shown in a gridview and Admin can Edit or Delete users via this gridview. If admin wants to search all users, there is a Search All Users Button.

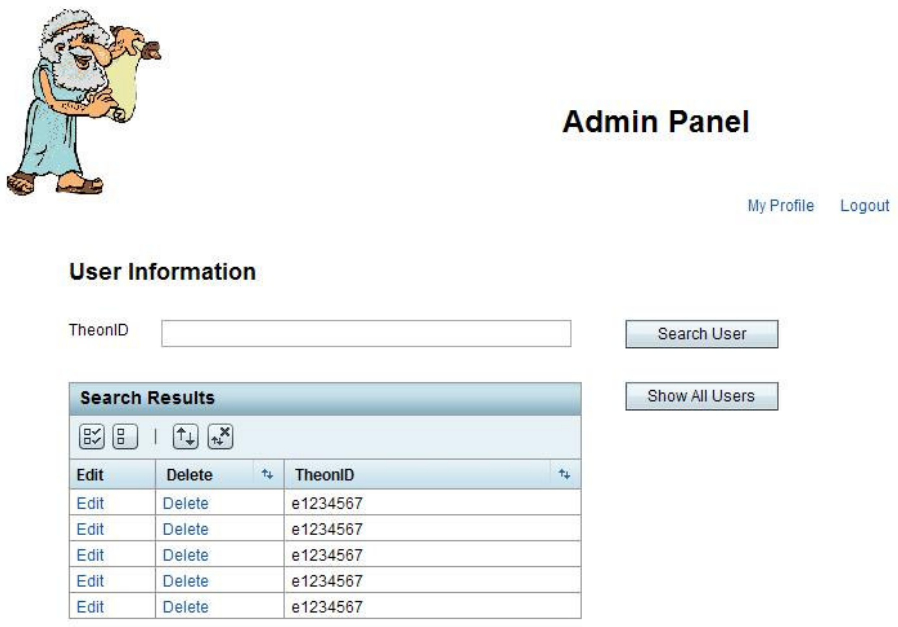
-Admin can delete a group or groups. via a ComboBox and DeleteGroup button.

-Admin can edit a group or groups. Firstly admin select a group and group properties (Group Name, Group Description) are showed and change these values via Edit Group button.

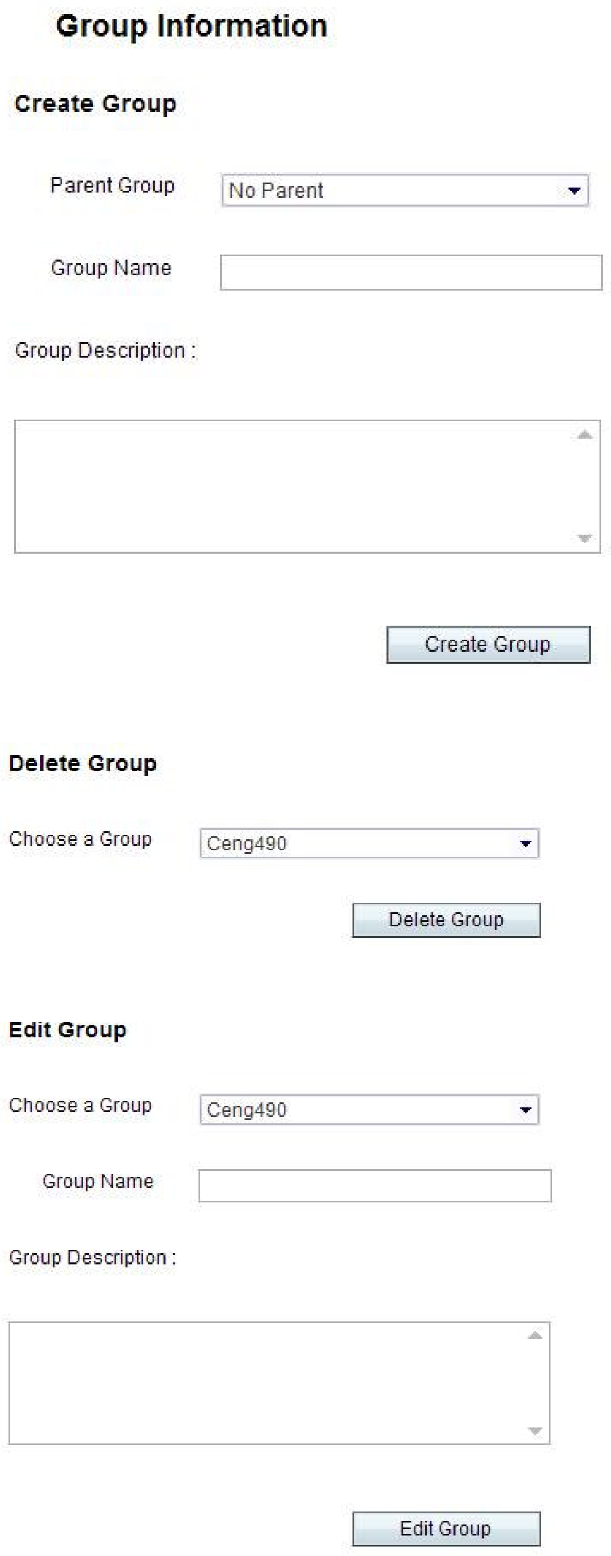
-Admin can create group, To create group admin firstly select ParentGroup (If no parent group want to be selected then admin use No Parent option) and then GroupName via a TextBox and GroupDescription via Multiline TextBox and finally add a group via using the Create Button.

-Admin can view his/her Priofile via MyProfile Page.

- Admin logs out via “Logout” button.



Edit	Delete	TheonID
Edit	Delete	e1234567
Edit	Delete	e1234567
Edit	Delete	e1234567
Edit	Delete	e1234567
Edit	Delete	e1234567


Group Information

Create Group

Parent Group: No Parent

Group Name:

Group Description :

Delete Group

Choose a Group: Ceng490

Edit Group

Choose a Group: Ceng490

Group Name:

Group Description :

7.0 Hardware and Software Requirements

7.1 Software Specifications

Server Side	- Unix / Linux - MySQL Server - Web Browser
Client Side	- Java Virtual Machine - Unix / Linux / Windows - Web Browser for forum access (optional) - News Reader Application for newsgroup access (optional) - RSS Aggregator for RSS access (optional) - Mail Client Application for mailing list access (optional)
Developers Side	- Eclipse Java IDE - MySQL DBMS

7.2 Hardware Specifications

Server Side	- Minimum 1024 MB RAM - Pentium IV Processor - Minimum 10 GB Hard disk
Client Side	- Minimum 256 MB RAM - Pentium IV Processor - Minimum 5 GB Hard disk

Developers Side	- Minimum 512 MB RAM - Pentium IV Processor - Minimum 5 GB Hard disk

7.3 Development Tools

We will make use of the following tools during the development period of the project:

- Java EE SDK, JSP and Apache TomCat
- Eclipse Java IDE
- MySQL Database Management System (DBMS)
- Java Virtual Environment
- Apache Web Server, James NNTP and SMTP Server

8.0 Testing Provisions

8. Testing Provisions:

To be able to deliver an error free server, we are going to identify and fix possible bugs by using specific testing techniques. By this way, we will be able to release an error free final product. We are planning to use the following testing techniques during our testing period:

8.1 Unit Testing

Unit testing is a procedure used to validate that individual modules of source code are working properly. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. Unit testing helps to eliminate uncertainty in the units themselves and can be used in a bottom-up testing style approach. By using this method, we can identify the functionality of our individual modules. However, it will not catch integration errors. Therefore, we need another testing method to identify integration errors.

8.2 Integration Testing

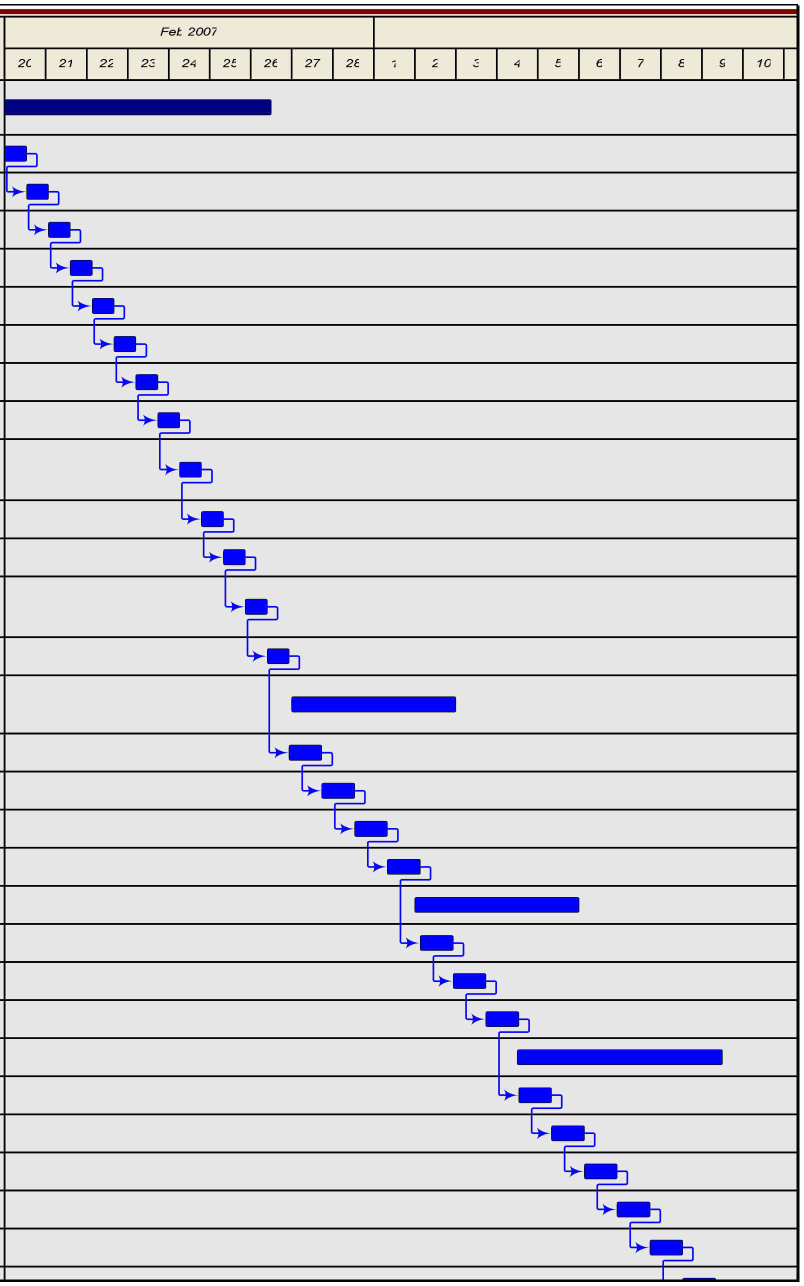
Integration testing is the phase of software testing in which individual software modules are combined and tested as a group. To identify our integration errors, we need integration testing. It follows unit testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

8.3 System Testing

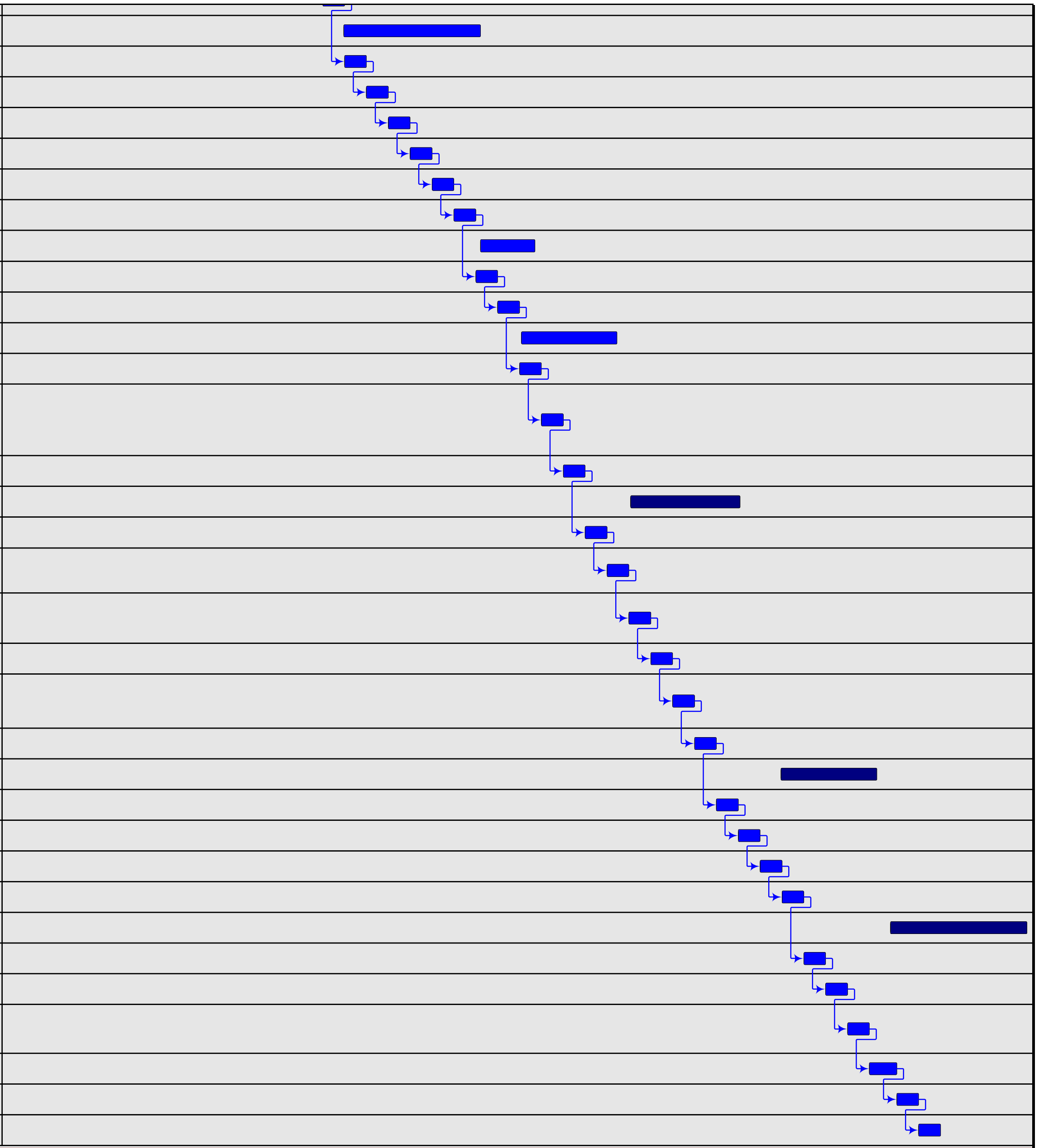
System testing is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing takes, as its input, all of the "integrated" software components that have successfully passed Integration testing. The purpose of Integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware.

9.0 Appendix-A (Gantt Chart for Implementation)

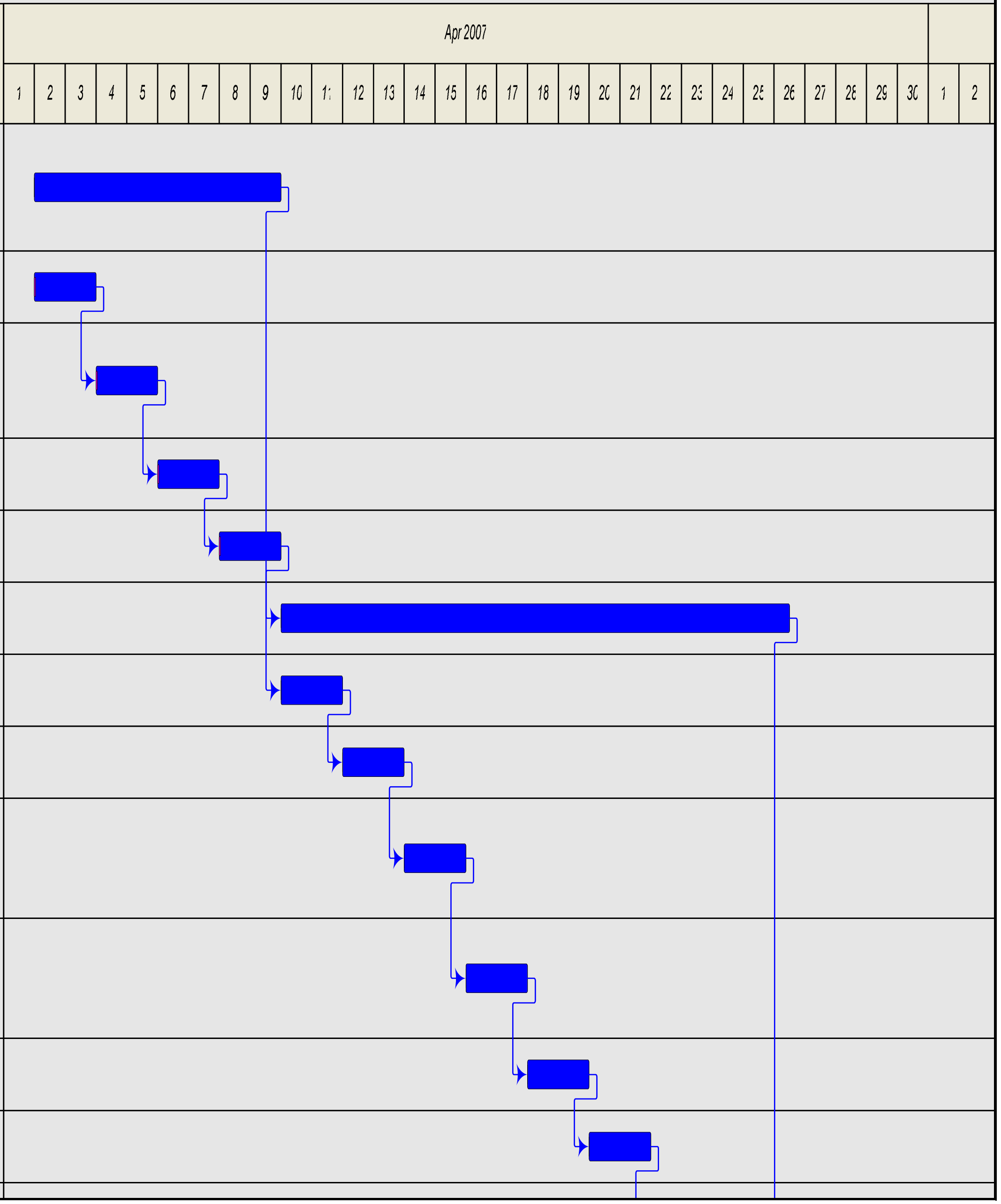
ID	Task Name	Start	Finish	Duration
1	**Client Side GUI Implementations**	2/20/2007	2/26/2007	6d 7 5h
2	Authenticaton Implementation	2/20/2007	2/20/2007	8h
3	User Profiler Implementation	2/20/2007	2/21/2007	8h
4	Service Selection Implementation	2/21/2007	2/21/2007	8h
5	Password Operations Implementation	2/21/2007	2/22/2007	8h
6	NewsTheon Viewer Implementation	2/22/2007	2/22/2007	8h
7	NewsTheon Sender Implementation	2/22/2007	2/23/2007	8h
8	ForumTheon Viewer Implementation	2/23/2007	2/23/2007	8h
9	ForumTheon Sender Implementation	2/23/2007	2/24/2007	8h
10	RSS Comment Viewer/Sender Implementation	2/24/2007	2/24/2007	8h
11	Message Searcher Implementation	2/24/2007	2/25/2007	8h
12	Administrator Panel Implementation	2/25/2007	2/25/2007	8h
13	Administrator Operations Implementation	2/25/2007	2/26/2007	8h
14	Integration of the Implementations	2/26/2007	2/26/2007	8h
15	**Implementation of the Common Classes**	2/27/2007	3/2/2007	4d
16	Authentication Class Implementation	2/26/2007	2/27/2007	12h
17	Database Connector Class Implementation	2/27/2007	2/28/2007	12h
18	Listener Class Implementation	2/28/2007	3/1/2007	12h
19	Message Converter Class Implementation	3/1/2007	3/2/2007	12h
20	**News Module Implementation**	3/2/2007	3/5/2007	4d
21	News Class Implementation	3/2/2007	3/2/2007	12h
22	News Thread Class Implementation	3/2/2007	3/3/2007	12h
23	Integration of the Classes	3/3/2007	3/4/2007	12h
24	**MailTheon Module Implementation**	3/4/2007	3/9/2007	5d
25	Webmail Class Implementation	3/4/2007	3/5/2007	12h
26	Webmail Operations Class Implementation	3/5/2007	3/6/2007	12h
27	Webmail Composer Class Implementation	3/6/2007	3/6/2007	12h
28	Webmail Reader Class Implementation	3/6/2007	3/7/2007	12h
29	Webmail Subscription Class Impletation	3/7/2007	3/8/2007	12h



24	**MailTheon Module Implementation**	3/4/2007	3/9/2007	5d
25	Webmail Class Implementation	3/4/2007	3/5/2007	12h
26	Webmail Operations Class Implementation	3/5/2007	3/6/2007	12h
27	Webmail Composer Class Implementation	3/6/2007	3/6/2007	12h
28	Webmail Reader Class Implementation	3/6/2007	3/7/2007	12h
29	Webmail Subscription Class Impletation	3/7/2007	3/8/2007	12h
30	Integration of the Classes	3/8/2007	3/9/2007	12h
31	**Mail Module Implementation**	3/9/2007	3/11/2007	2d
32	Mail Class Implementation	3/9/2007	3/10/2007	12h
33	Integration of the Classes	3/10/2007	3/10/2007	12h
34	**RSS Comment Module Implementation**	3/11/2007	3/14/2007	3d 7.5h
35	RSS Comment Class Implementation	3/10/2007	3/11/2007	12h
36	RSS Comment Viewer / Sender Class Implementation	3/11/2007	3/12/2007	12h
37	Integration of the Classes	3/12/2007	3/13/2007	12h
38	**Web Forum Module Implementation**	3/15/2007	3/18/2007	4d
39	Web Forum Class Implementation	3/13/2007	3/14/2007	12h
40	Web Forum Main Operations Class	3/14/2007	3/14/2007	12h
41	Web Forum Subscription Class Implementation	3/14/2007	3/15/2007	12h
42	Web Forum Search Class Implementation	3/15/2007	3/16/2007	12h
43	Web Forum Group Operations Class Implementation	3/16/2007	3/17/2007	12h
44	Integration of the Classes	3/17/2007	3/18/2007	12h
45	**Web Module Implementation**	3/20/2007	3/23/2007	3d 7.5h
46	RSS Handler Class Implementation	3/18/2007	3/18/2007	12h
47	Forum Handler Class Implementation	3/18/2007	3/19/2007	12h
48	Admin Handler Class Implementation	3/19/2007	3/20/2007	12h
49	Integration of the Classes	3/20/2007	3/21/2007	12h
50	**NewsTheon Module Implementation**	3/24/2007	3/29/2007	5d
51	NewsTheon Class Implementation	3/21/2007	3/22/2007	12h
52	News Operations Class Implementation	3/22/2007	3/22/2007	12h
53	News Sender Class Implementation	3/22/2007	3/23/2007	12h
54	News Reader Class Implementation	3/23/2007	3/24/2007	1d
55	News Subscription Class Impletation	3/24/2007	3/25/2007	12h
56	Integration of the Classes	3/25/2007	3/26/2007	12h



ID	Task Name	Start	Finish	Duration
1	**ServerTheon Core Module Implementation**	4/2/2007	4/9/2007	8c
2	Core Service Class Implementation	4/2/2007	4/3/2007	2c
3	Service Message Class Implementation	4/4/2007	4/5/2007	2c
4	Service Response Class Implementation	4/6/2007	4/7/2007	2c
5	Integration of the Classes	4/8/2007	4/9/2007	2c
6	**Admin Module Implementation**	4/10/2007	4/26/2007	16c 7h
7	Admin Module Implementation	4/10/2007	4/11/2007	2c
8	Admin Display Class Implementation	4/12/2007	4/13/2007	2c
9	Admin Group Operations Class Implementation	4/14/2007	4/15/2007	2c
10	Admin Message Operations Class Implementation	4/16/2007	4/17/2007	2c
11	Admin Sender Class Implementation	4/18/2007	4/19/2007	2c
12	Admin Reader Class Implementation	4/20/2007	4/21/2007	2c



13	Admin Client Operations Class Implementation	4/22/2007	4/23/2007	2d
14	Integration of the Classes	4/24/2007	4/25/2007	2d
15	**Integration of the Modules**	4/26/2007	5/10/2007	14d
16	**Testing**	5/10/2007	5/24/2007	14d
17	Unit Testing	5/10/2007	5/16/2007	6d
18	Integration Testing	5/16/2007	5/20/2007	4d
19	Debugging & Fixing	5/20/2007	5/28/2007	8d 7h
20	**Documentation**	5/24/2007	5/30/2007	6d
21	User-Manuel Preparation	5/29/2007	6/2/2007	4d 7h
22	Installation Manuel Preparation	6/2/2007	6/6/2007	4d
23	**Packaging and Releasing**	6/6/2007	6/10/2007	4d 7h
24	Packaging Whole Project	6/6/2007	6/10/2007	4d 7h