1. Introduction 2
1.1 Problem Definition 2
1.2 Project Scope and Goals 3
1.3 Usage Areas 3
1.4 Design Constraints 4
1.5 Design Objectives 5
2. Data Design 7
2.1 Data Objects 7
2.2 ER Diagrams 10
2.3 Data Dictionary 11
2.4 Database Description 13
3. Architectural and Component-level Design 14
3.1 Structure Chart 14
3.2 Data Flow Diagrams 16
3.2.1 DFD Level 0 16
3.2.2 DFD Level 1 17
3.2.3 DFDs Level 2 18
3.2.4 Data Dictionary 22
3.3 State Transition Diagrams 26
3.3.1 State Diagram Adding a Rule: 26
3.3.2 Checking Logs State Diagram: 27
3.3.3 Request – Response Diagram 28
3.4 Description of Components 29
3.4.1 Plugin Architecture 29
3.4.2 Default Plugins 33
3.4.3 Capabilities of Default Plugins 37
3.4.4 Authentication 38
3.4.5 Logging 39
3.5 Use Case Diagram 41
3.6 Activity Diagrams 42
3.7 Collaboration Diagrams 47
3.8 Sequence Diagrams 51
3.9 Class Diagram 56
4 User Interface Design 57
4.1 Screenshots 57
4.1.1 Summary View 57
4.1.2 User and Group 58
4.1.3 Plugin and Rule Management 58
4.1.4 Settings 59
4.2 In Action 59
5 Requirements 61
5.1 Minimal Hardware Requirements 61
5.2 Minimal Software Requirements 61
6 Project Schedule 62
6.1 Project Task Set 62
6.2 Gantt Chart 64
7. Testing 65
7.1 Unit Testing 65
7.2 Integration Testing 65
7.3 Alpha Testing 65

1. Introduction

During the analysis phase of our project, we investigated the possible problems about web, its usage, and its control within restricted zones. We tried to identify and define those problems, and come out with creative solutions to those that are within our project's scope. We provided comprehensive information about those problems and our proposed solutions in our analysis report. After our delivery of the analysis report, we continued working on the problem set and our solutions. We came to conclusions on our project's technical aspects and infrastructure.

1.1 Problem Definition

In recent years, Internet has seen a great rise in its popularity and usage areas. People of all occupations and ages are on the internet, industries depend on it heavily, and there are completely new businesses that exist primarily in the internet. Companies depend heavily on internet usage, as a result most of the employees have access to the internet. Other than big companies, many other organizations are open to internet. Some examples are governmental bodies, schools and universities, non-profit organizations, hospitals, small offices. Examples are not limited to these, and the most common internet user is the home user.

The giant structure and variety of the internet makes it a valuable information source and communication medium. But any useful thing has some drawbacks, and internet is no exception. In fact ease of access and abundance of uncontrolled traffic makes the internet a difficult place to stay safe. And a large organization is not only concerned with outside threats, but also inside threats which stem from workers or members. The workers or members of the organizations may visit sites or download files that may bring harm to the whole network of the organization. Some sensitive information may leak from the network to the outside world, either by a careless worker or one with a malicious intent.

Companies and organizations mostly used traditional firewalls, which does packet filtering by looking at easily extractable data, like IP address or port of the communicating parties. This would be useful if the intent was to block and control some servers/protocols only. But the internet traffic around the world consists of web traffic mostly, distributed to a large number of servers in many different locations throughout the world. Traditional firewalls are not able to tackle with difficulties presented by such a gigantic network, especially in a corporate environment defined above. It is the content of the traffic, not the source/destination that should be checked to provide comprehensive control over the information flow. In a present day

environment, where nearly the whole internet traffic is web traffic, blocking ports have little influence on the information security.

Our project aims to solve the most important problems of the modern day corporate and organizational network, by providing an application layer gateway; which will enable the network manager to control web traffic of the network in many different ways. Name of our project is IronCurtain and it will not only solve problems of big corporations and organizations, but also home users and small networks. IronCurtain will provide flexible control over the whole web traffic, user and group based control, extensive logging capabilities, and a configuration interface that is accessible from web. We plan to design and build IronCurtain so flexible, strong and easy to use, such that it will answer most of the modern day internet-based concerns.

1.2 Project Scope and Goals

Our project, named IronCurtain, will implement an application level gateway for web filtering and access control. Our project's goals are as follows:

- Complete HTTP/1.1 support
- HTTPS support
- Decomposition of Entire Communication
- Plugin architecture for rules and actions
- Complete configuration over Web
- Secure User Authentication
- Logging and Accounting
- Multi-threaded (thread-pool) implementation
- Alerts (via plugins)
- Content identification (via plugins)

1.3 Usage Areas

IronCurtain will be flexible enough to satisfy needs of a great variety of entities. From large enterprises to home users, IronCurtain will have an answer to all of these entities' problems, thanks to the flexible and extendable plugin mechanism.

- **Companies:** Modern day companies use internet access extensively to maintain their

operations and profitability. It is essential for companies to access internet. If the access is uncontrolled, many adverse effects of the internet give harm to the company overall. Some of these are: Productivity loss associated to worker distraction caused by uncontrolled web site access, maliciously designed web sites, leakage of sensitive company information, etc. By using IronCurtain, companies can enable appropriate filters to take precautions about these problems. Also IronCurtain would be a security layer between company servers and the internet, checking for possible security violations and attacks and preventing according to the rules. This would require no modification of the system, only writing the appropriate rules will be sufficient. Also sensitive information about the company network may be blocked using IronCurtain. IP address of the computers can be replaced by the address of the proxy.

- **Universities:** A university environment is usually more relaxed than a corporate one. But still administration of a university would need firm control on the network. Students and academics in the university might use up all the bandwidth of the university. Or they might cause legal trouble by downloading illegal files. It would be possible to control bandwidth of the users and their internet activities using IronCurtain.
- **Other Areas:** Flexible plugin architecture of IronCurtain will enable its usage in a variety of networks, even a single computer network. Appropriate rules and plugins for the area of application will be chosen and IronCurtain will protect the network from inside or outside malicious traffic, and it will enforce other principles determined by the network's owners.

1.4 Design Constraints

Our main design constraints are as follows:

- **Time**

  Our fixed schedule is determined by our course syllabus. We have approximately six months remaining to finish the project completely. The design should be finished in one month. During the design we will also work on the prototype and it will be finished in one and a half months from the delivery of initial design report.

- **Language constraints**

  We decided to use the Python language as implementation language. It allows easy usage of plugin. In fact, python is used as a scripting language inside many applications. Most of the time we will be using Python's integrated libraries.

- **Performance and Network Latency**

IronCurtain application level gateway aims to introduce lowest levels of latency to the network communication speed. The software will use threads to be more performant than a forking implementation. Also the software will reuse existing connections and will not try to open a new connection which takes CPU time and introduces latency. GZIP compression and chunked encoding feature of HTTP 1.1 standard will be used to reduce latency.

- **Maintenance**

  The IronCurtain will require minimal maintenance. After the administrator defines the plugin, rules, general settings, per user and per group settings; there will be no need to check the operation of the gateway, other than the cases of; adding of a new rule and querying for statistics.

- **User Interface**

  IronCurtain is not a user interface intensive application. Most of its operation happens behind the communication of the members of the network. The user interface is accessed when settings are to be changed, or statistics are to be displayed. Other than these conditions, no user interface is required for normal operation. The user interface of the settings and statistics parts of the software will be easy and intuitive to use. The statistics should be displayed in a variety of easy to understand approaches.

1.5 Design Objectives

Our project's design objectives are as follows:

- **Portability**

  Our usage of Python language makes IronCurtain easily portable to any operating system that has the necessary Python run-time components; Linux, BSD, Mac OS X, Microsoft Windows, AIX, Amiga, AROS, AS/400, BeOS, OS/2, OS/390, Irix, Palm OS, Plan 9, PlayStation 2, Psion, QNX, RISC OS, Sharp Zaurus, Solaris, Symbian OS, VMS, VxWorks, Windows CE/Pocket PC, Xbox.
  Because of this portability, users with any specified OS should be able to use IronCurtain.

- **Extendibility**

The plugin architecture of IronCurtain will enable very extendible and flexible operation. If there is a need for a new limitation/control/statistics options, there may be possible affordable solutions using IronCurtain. The easiest one is the usage of existing plugin and writing of new rules that satisfy the requirements. Or a new plugin can be written using the python programming language. Using plugins any kind of behavior can be added to IronCurtain.

- **Maintainability**

  Maintainability is an important objective for IronCurtain. The plugin system is very modularly designed, plugin are independent of each other. Also functionality of IronCurtain is upon plugin. Because of this modular design, we could change the internals of any plugin without causing problems for the other untouched plugins and general operation of IronCurtain. This makes IronCurtain easy to maintain, because changes to one plugin/functionality do not require the rewriting of other plugin.

- **Reliability**

  The system should be as bug free as possible. All sub components should work asynchronously, so that any delay caused by one of the components should not block other components' work flow.

- **Security**

  There are two aspects of security. First, the internal network which clients connect to our software may not be secure. User authentication should be handled over a secure connection using SSL. Secondly, the Internet is, unfortunately, filled with people with malicious intents. IronCurtain must be safe from malicious code attacks like buffer overruns(which, we think that coding in a high-level language like Python will help enormously) itself, and also it must provide methods that allow administrators to define rules to block such known attacks.

2. Data Design

The plugins and rules will be kept in separate files on the file system. The data of users, groups and rules will be stored in the database. The logs of users and rule actions will be written to database as well. In order to store the data in a structured form, the data objects will be used. In this section, we will look at the data objects, their relationships, the ER-diagram and the data dictionary to describe the data.

2.1 Data Objects

User

The User entity will store data associated with the users of the system. When they register with the system the data they enter will be stored as an instance of the User object and they can change the information at any time. The attributes of the entity will be:

- User_ID
- Real_Name
- Passwd
- Email
- Group_ID

The *User_ID* will be a string that will be used as the primary key as it will be unique to each user in the network. The *User_ID* and password will be used to log on during authentication. *Passwd* is the MD5 hash of the user password. *Group_ID* will be the reference to *Group* entity, which the user is associated with. The user does not have to be assigned to a Group. *Email* will be the user's e-mail address. The user will be notified by e-mail if necessary.

Admin

Admin entity is just a User entity, but it will be used to identify the administrators. Admin entity will keep the *User_ID*'s of the users who has administrative rights. The users in this entity will be able to log in to administrative page of the system with their passwords.

Group

Group entity is the generalization of User entity. It will be used to group users to ease applying same rules to many users. The attributes of the entity will be:

- Group_ID
- Description

The *Group_ID* will be a string that will be used as the primary key as it will be unique to each *Group*. *Description* will be just an info string.

UserLog
UserLog entity will store the logs of the user actions.

- User_ID
- Site
- Domain
- Bandwidth
- Open_Time
- Close_Time

User_ID and the *Open_Time* together will be the primary key. *User_ID* is the foreign key and reference to *User* entity. *Site* is the string to hold the site URL that is visited. *Domain* is a string and it is the domain name of the visited site. *Bandwidth* is an integer that is the size of the data transfer. *Open_Time* and *Close_Time* are the action times. *Site, Domain, Bandwidth* will be stored associated with the user.

Plugin

The Plugin entity will store the identifiers of the plugins in the system. The plugin schemas will be saved as xml files so we will just keep the ids of the plugins to relate them with the rules. The attributes of this entity will be:

- Plugin_ID
- Plug_File
- Schema_File
- Plug_Hash
- Schema_Hash
- Description

The *Plugin_ID* will be a string that will be used as the primary key as it will be unique to each Plugin. This key will be read from the xml file. The *Plug_File* and *Schema_File* attributes will

keep the paths of the Plugin file and schema file of the Plugin, respectively. The MD5 hashes of the files will be stored in *Plug_Hash* and *Schema_Hash*, accordingly. If any change occurs in these files, it will be checked from hashes. *Description* will be just an info string.

Rule

The Rule entity will store the identifiers of the rules that are generated from Plugins. Every rule derives from a Plugin template. The attributes of the entity will be:

- Rule_ID
- Plugin_ID
- File
- File_Hash
- Description

Rule_ID is a string and primary key. *Rule_ID* is the identifier of the rule that is generated from a plugin schema and written to an xml file. The File attribute will keep the path of the xml file. *File_Hash* will be the MD5 hash of the file and make it possible to check whether any change occurs in the file. *Plugin_ID* is the foreign key and reference to *Plugin* entity. Description is just an info string.

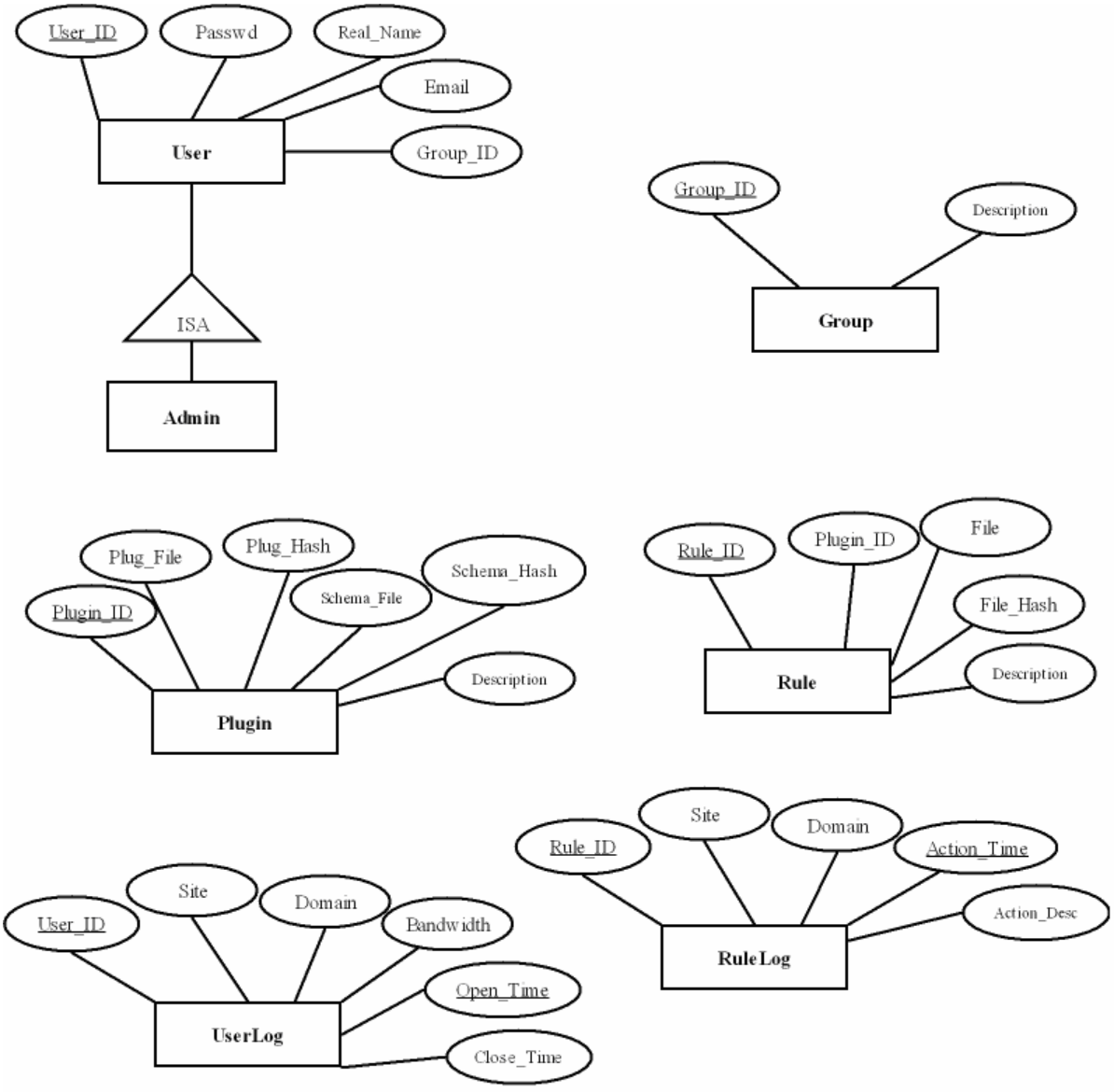
RuleLog

RuleLog entity will store the logs of rule actions. It has a similar schema to UserLog entity.

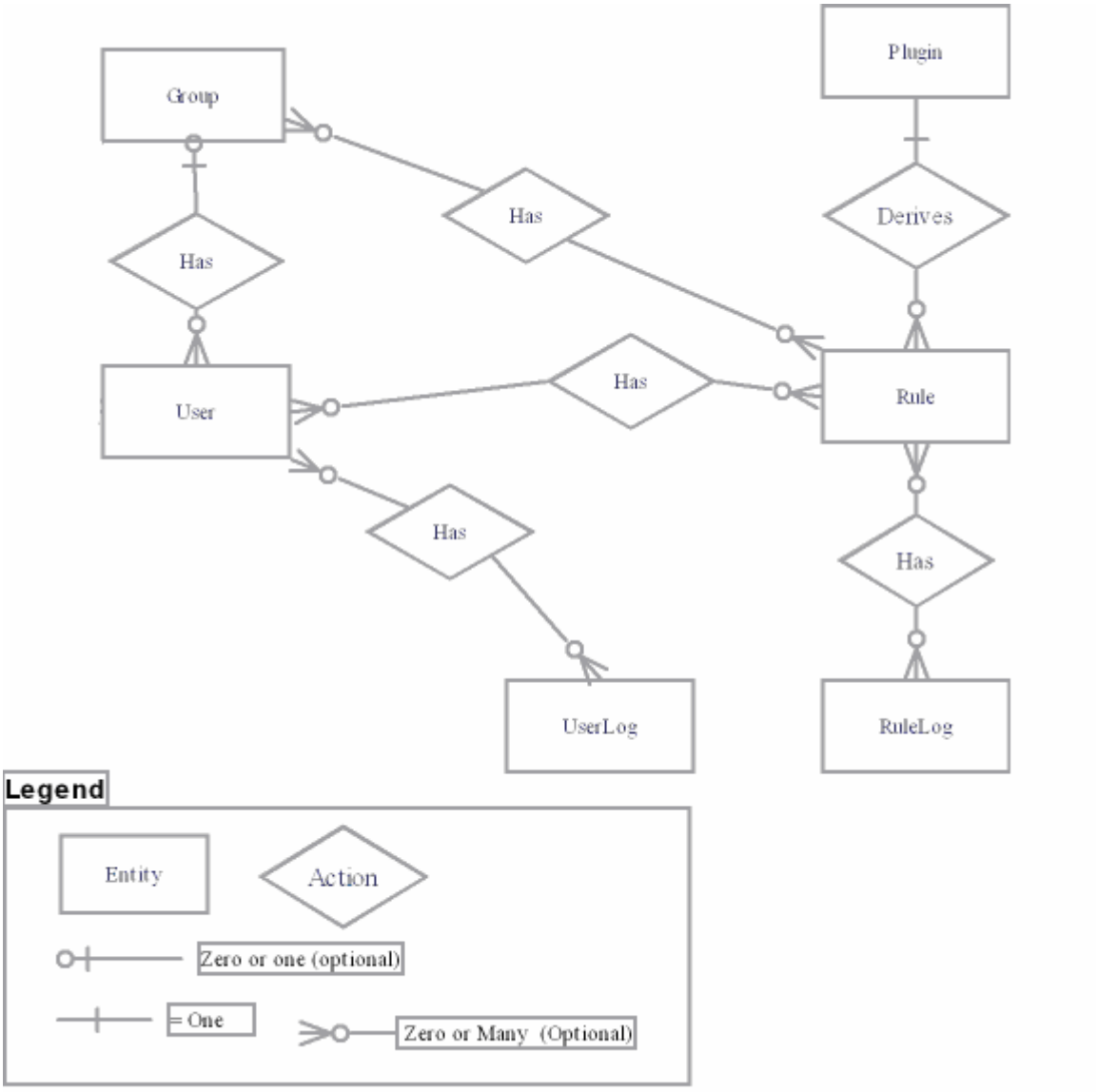
- Rule_ID
- Site
- Domain
- Action_Time
- Action_Desc

Rule_ID and the *Action_Time* together will form the primary key. *Rule_ID* is the foreign key and reference to *Rule* entity. *Site* is the string to hold the site address that is visited. *Domain* is a string and it is the domain name of the visited site. *Action_Time* is the time of the action taken. Information about the action will be kept inside the *Action_Desc* attribute.

2.2 ER Diagrams



Data Objects



ER-Diagram

2.3 Data Dictionary

User

Name	User
Alias	-
Where / How used	The people that will use the system
Description	Every actor using the system is defined to be a user.

User_ID

Name	User_ID
Alias	-
Where / How used	The users will enter their User_ID together with their passwords to log into the system. Administrators will have permission to access to control panel page with their passwords.
Description	Every user has a unique User_ID.

Passwd

Name	Passwd
Alias	-
Where / How used	While logging into the system
Description	The password is to secure the system. Unauthorized users cannot access the control panel page. The user's password is converted to MD5 hash and checked with the one in the database.

Admin

Name	Admin
Alias	Administrator
Where / How used	-
Description	Admin is a special type of user who has all privileges. He/she will be able to access the control panel page. Then he/she will: - add/remove users - change users' and groups' logging and filtering rules - check logs - add/remove rules

Group

Name	Group
Alias	-
Where / How used	Create Groups and add users to groups.
Description	Groups will be created by the 'Admin' and they will be used to assign specific rules to many people at once. It will make the user management easier.

Plugin

Name	Plugin
Alias	-
Where / How used	While creating new rules.
Description	After a Plugin is written, it defines a schema and a proper template will be generated in order to create new rules from this Plugin.

Rule

Name	Rule
Alias	-
Where / How used	They are defined using an existing Plugin and assigned to users and groups.
Description	Admin defines the rule on the web control page and assign the rule to any user or group.

UserLog

Name	UserLog
Alias	-
Where / How used	While logging users actions
Description	UserLog contains log items: site as visited site's name; domain as site domain; bandwidth as data transfer size.

RuleLog

Name	RuleLog
Alias	-
Where / How used	When rules take action
Description	Rules will write their actions to database as 'RuleLog's. Visited site, site domain, action time, action description will be saved.

2.4 Database Description

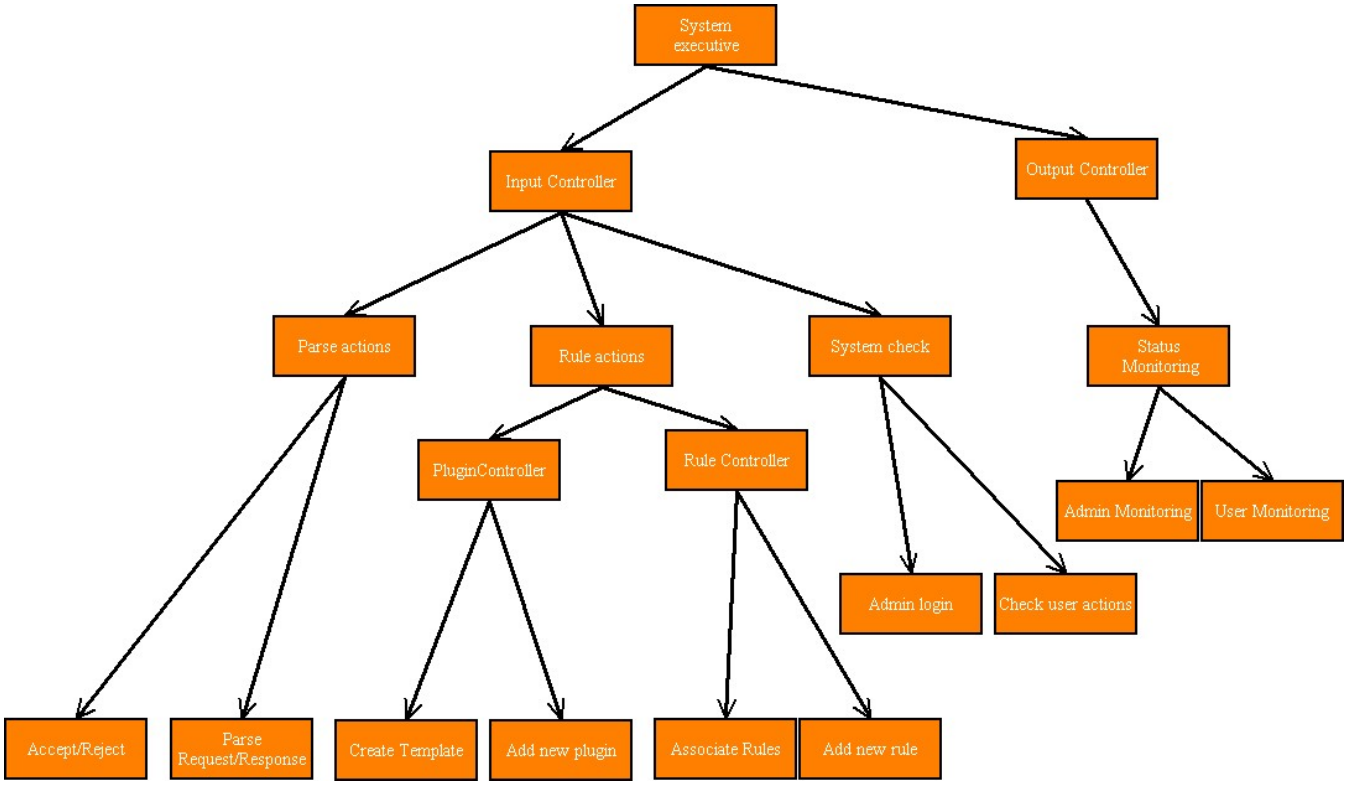
The database management system we are using for IronCurtain is SQLite. Tables will be created and be filled using Python's SQLite library. Database will store all of the information of the user's actions as logs. Also, user and rule relation are stored in the database. When the system needs retrieving data, SQL queries are used to get the necessary records.

3. Architectural and Component-level Design

This section gives details about program structure, components, and software interface.

3.1 Structure Chart

Our project has mainly into two modules which are named as input controller and output controller. Our system is also responsible of the appropriate coordination of these main parts and the system's maintainability with the ultimate updates.



The followings are the modules in our application.

- **Accept/Reject**

This module determines whether our proxy accept or reject the content. The parser can accept or reject the HTML according to the rules applied.

- **Parse Request/Response**

Every HTTP Request and Response will be parsed in this module. Parser will first check the user and then it will get the rules for this user. Then it will parse the content by using the selected rules.

- **Create Template**

This module is for creating a template of plugins so that we can easily create rules over plugins. It writes the template to an XML file. The template creation operation is only done once when we created a new plugin.

- **Add new plugin**

This module is for adding a new plugin to our proxy. A new plugin is a python code. In our parser we will import that code when we are using a user defined rule. After adding a new plugin also Create Template module creates a template for that plugin.

- **Associate Rules**

This module is used for determining which rules are applied to which users. A rule can be applied more than one user or group. This information are stored and changed in this module.

- **Add new Rule**

This module is for creating a new rule from a plugin. First the user selects a plugin from the menu. Then a template is shown on the screen and the user fills the variables and functions that he will use. Then the rule is created.

- **Admin/User login**

This module is authentication. We will use an SSL encrypted web-based authentication system. User will simply provide a username and a password, then it will be checked whether that user is an administrator.

- **Check user actions**

Admin can check user actions in this module. He can get statistical info about users. He can search for specific information in logs.

- **Admin monitoring**

This module is used for monitoring admin actions. All admin actions are recorded as logs.

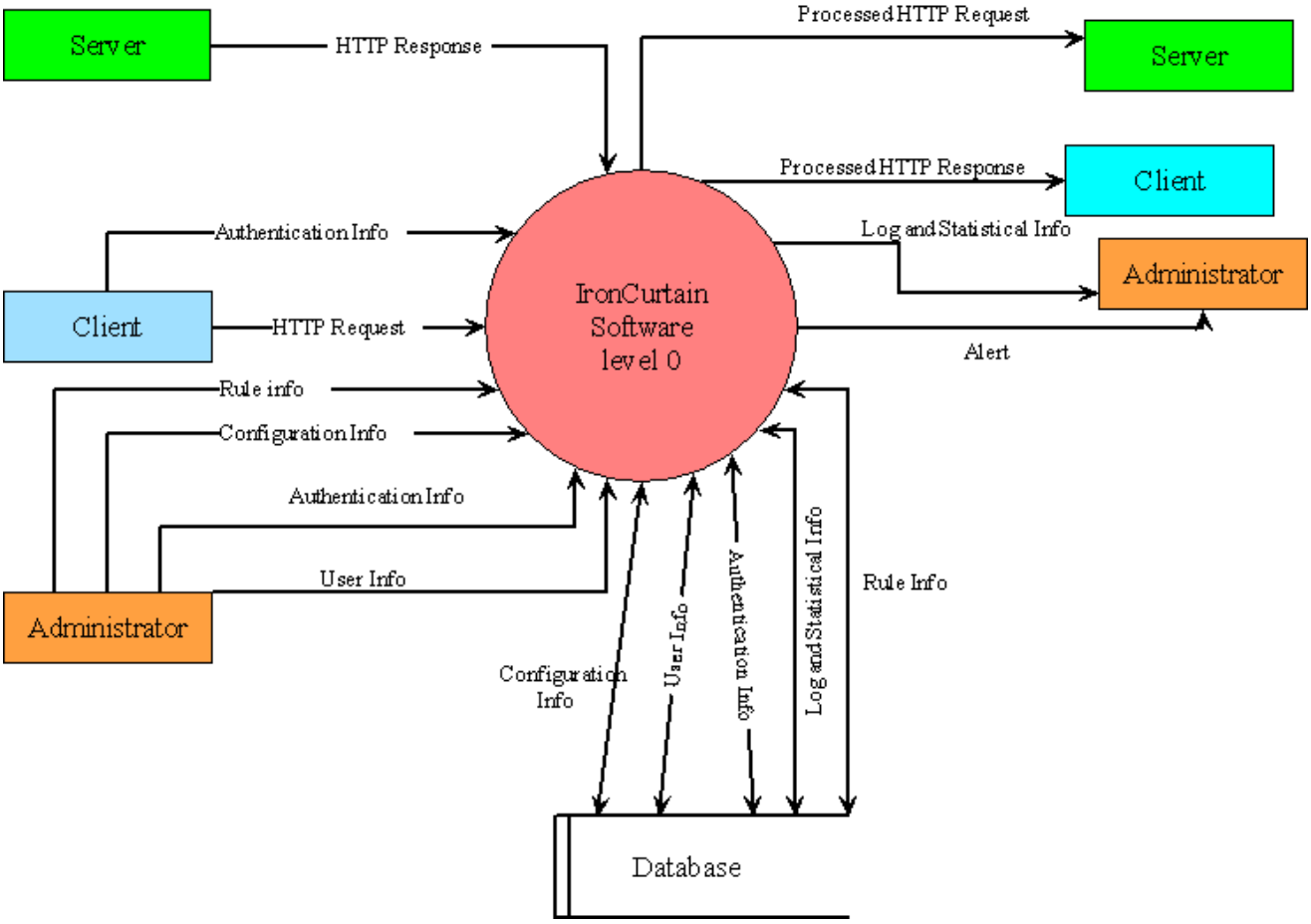
- **User monitoring**

This module is used for monitoring user actions. All user actions are recorded as logs.

3.2 Data Flow Diagrams

In this section, the functional model of IronCurtain is presented. It is composed of process specifications and DFD of the three levels (Level0, Level1 and Level2) of the system.

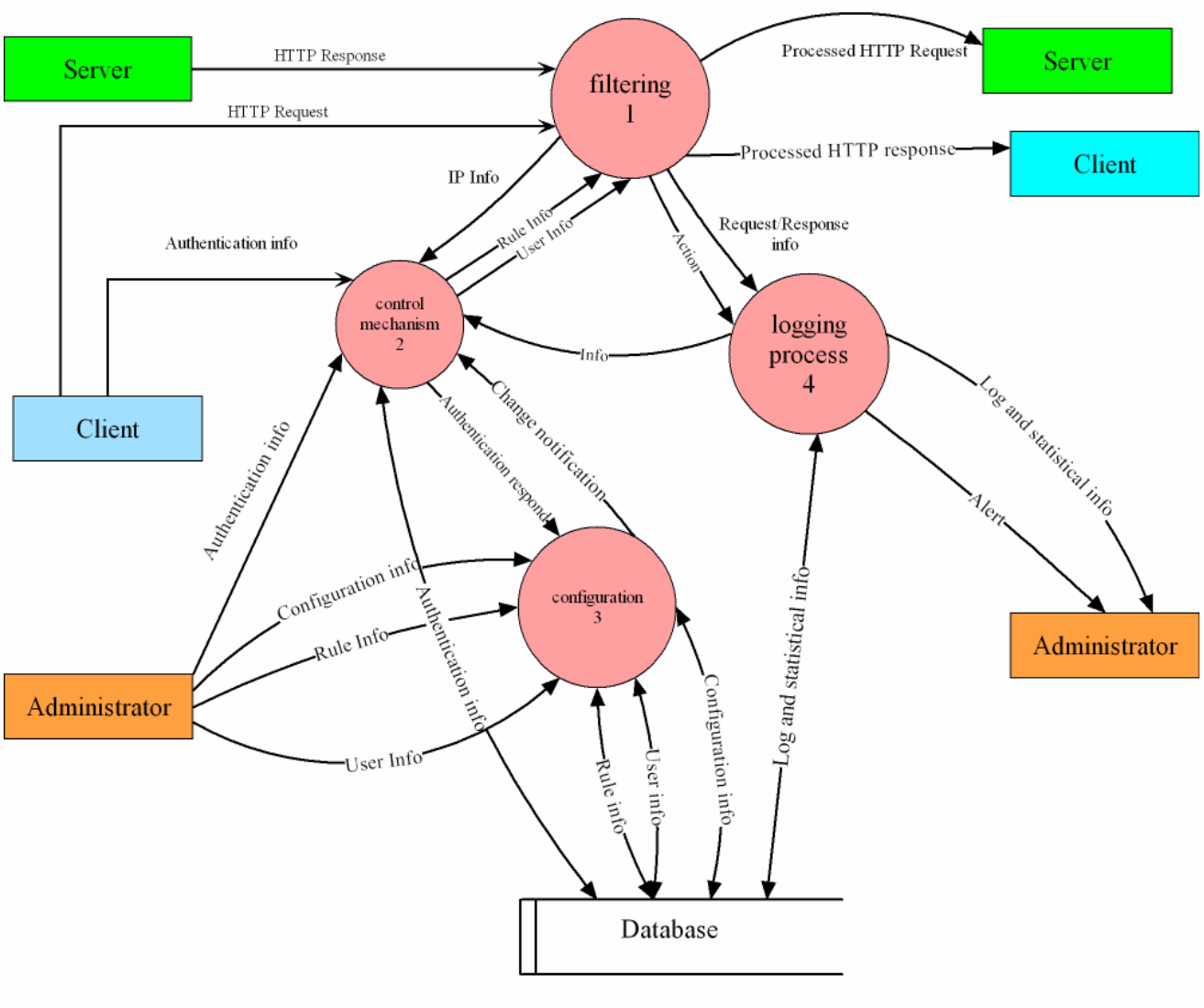
3.2.1 DFD Level 0



As seen from the diagram, we have three users: server, client and administrator. The main functionalities of the users are shown on the diagram. According to the requests IronCurtain writes some information to the database or process it and give a response to the users.

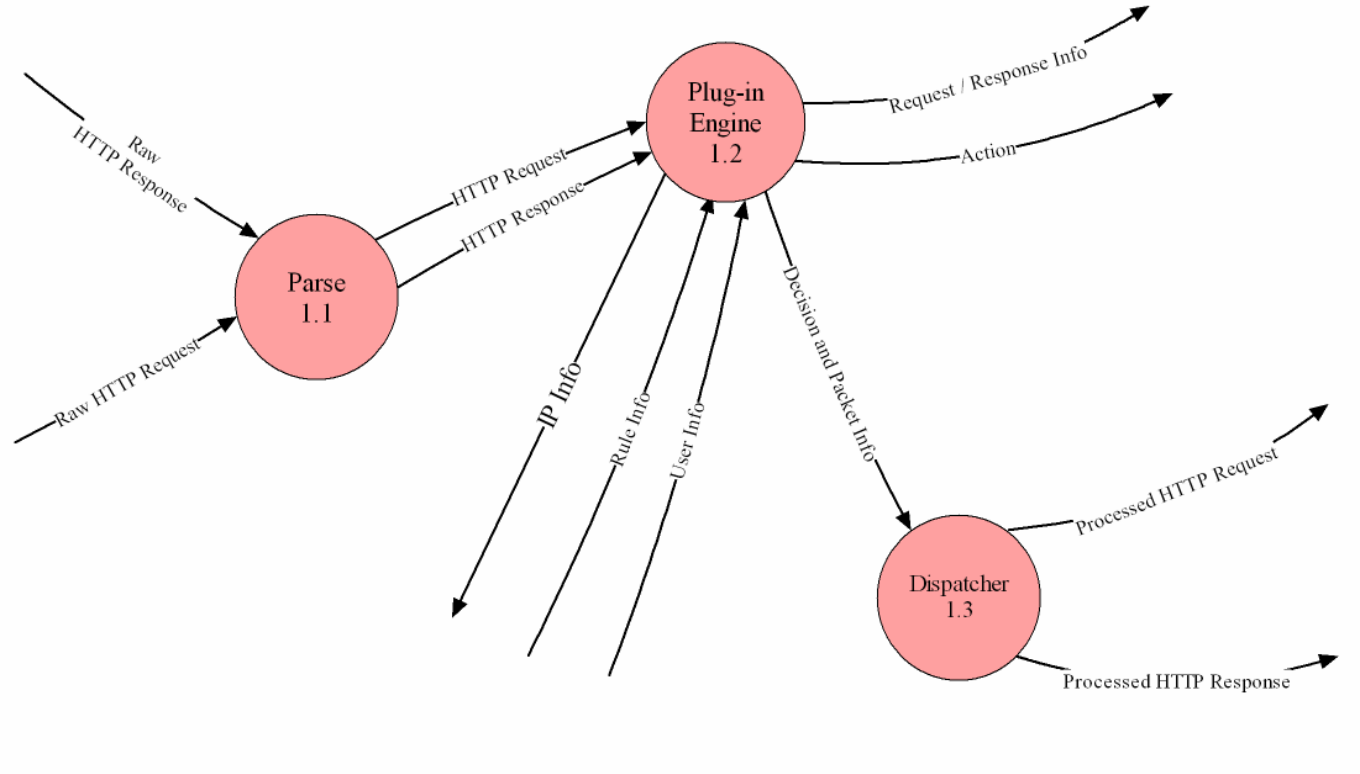
3.2.2 DFD Level 1

This is a more detailed diagram of IronCurtain. The main parts of IronCurtain are filtering, control mechanism, configuration and logging process. These parts communicate with users and each other.



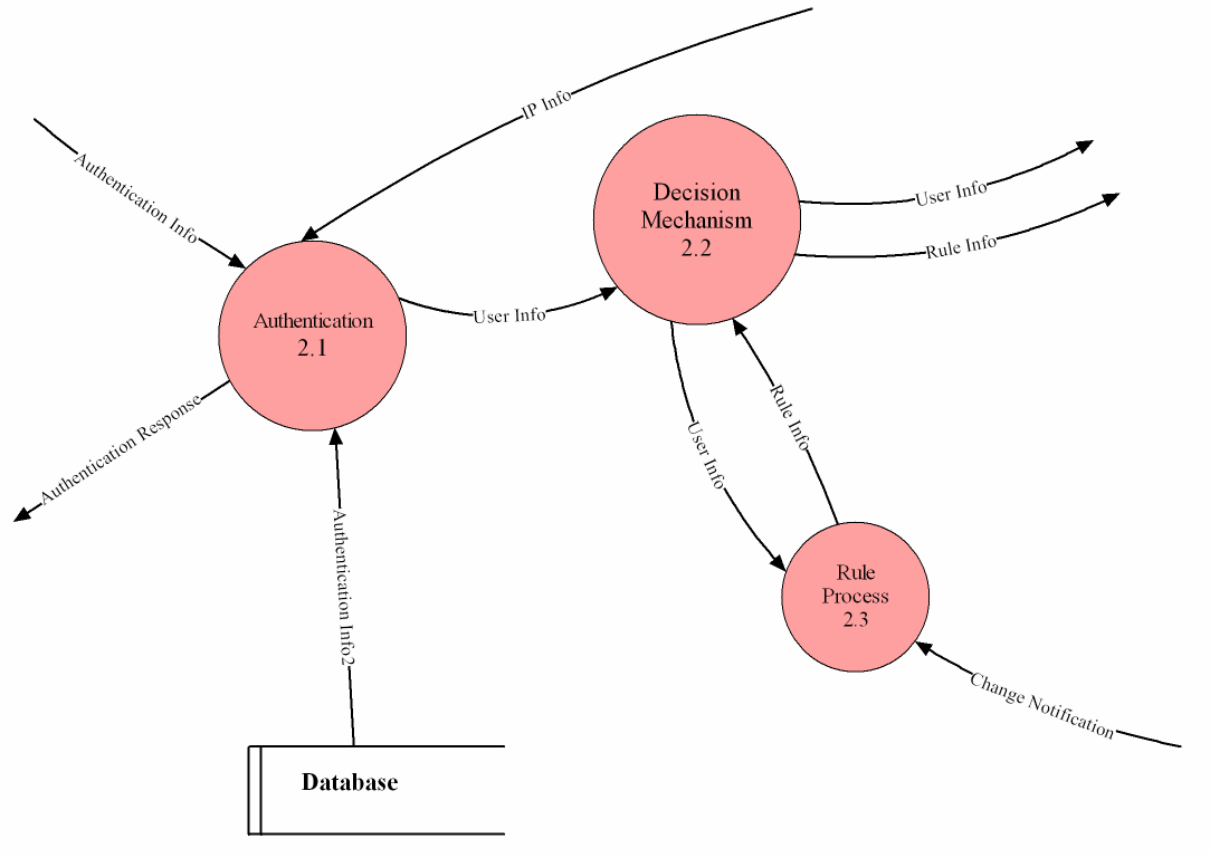
3.2.3 DFDs Level 2

Filtering



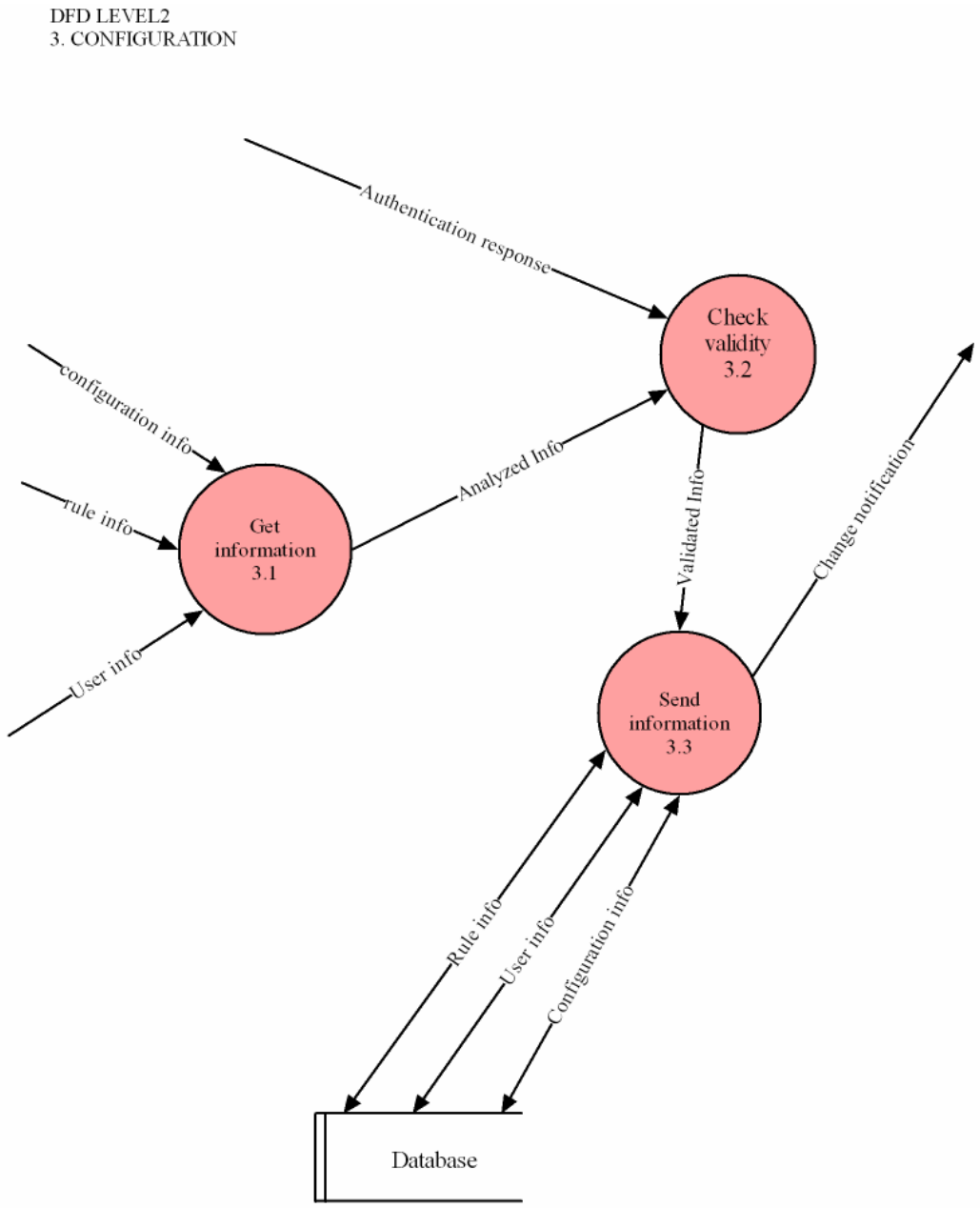
This is a more detailed diagram of the filtering part in IronCurtain. This is the most important part because all the requests and responses are processed here. First the requests and responses are parsed and then it is analyzed. In analyze part the validity of the request and response is checked by the control mechanism. After analyzing the processed request or response are sent by dispatcher to the users.

Control Mechanism



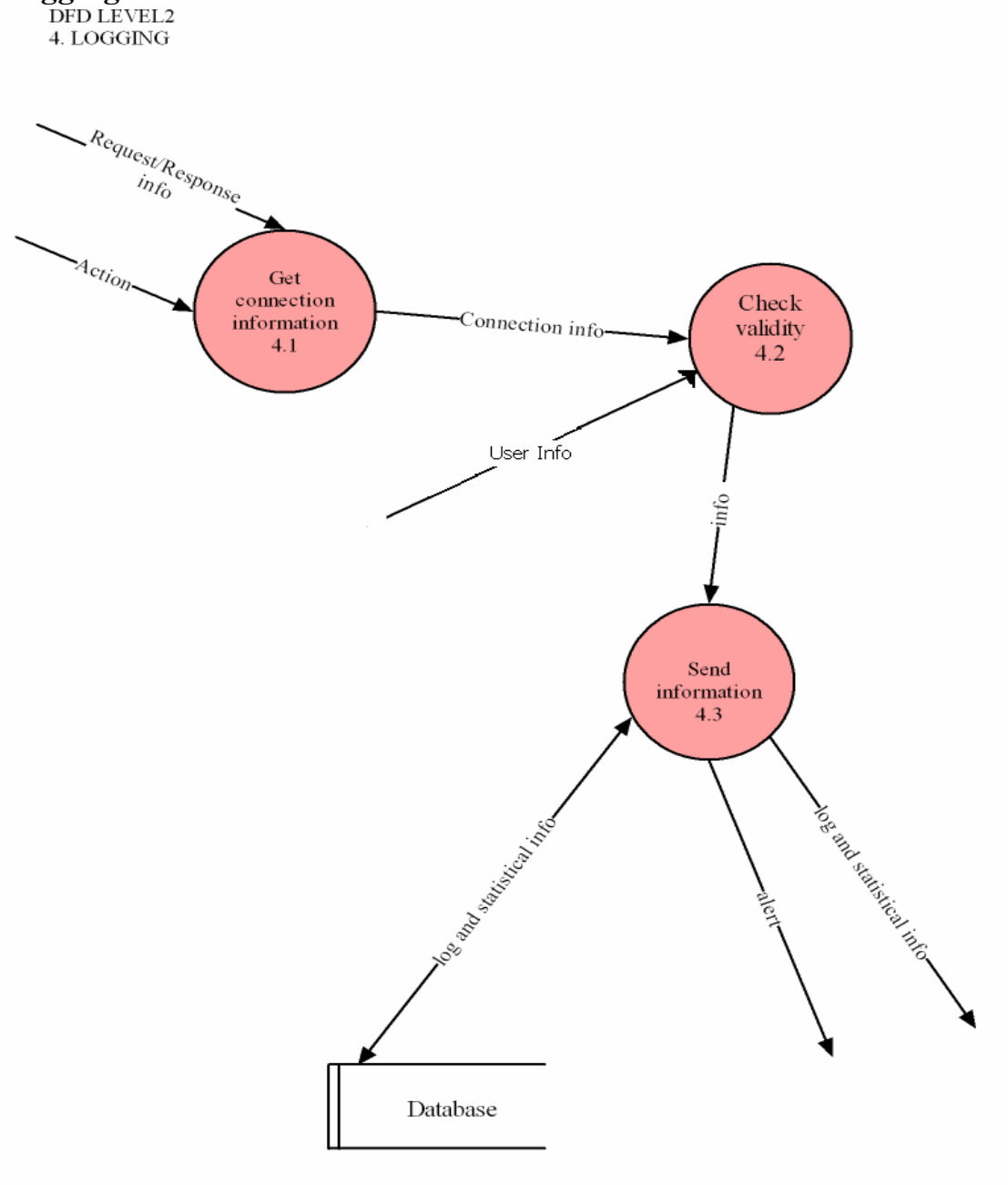
Control mechanism briefly checks the access rights and user authentication. It is connected to all the other modules. It gets the info from the other parts and checks whether this user has the right to do that action or not. According to that it sends an answer of approval/disapproval.

Configuration



Configuration part is used for changing the properties of users, rules and main configuration of the proxy. It checks the access rights of the user by using control mechanism. It writes the information to the database. Only administrator user can use that part of the IronCurtain.

Logging



Logging process is used for logging all the activities of the user and also alerting if there is a violation of a predefined condition. All the request and response info is provided to logging process by filtering part. The logging process sends log and statistical info to the database and administrator user.

3.2.4 Data Dictionary

Name	Authentication Info
Aliases	Authentication Info2 User name/Password
Where used/ How used	Control Mechanism - Authentication 2.1 (input/output) Client (output) Administrator (output) Database (input/output)
Description	This data is the username and the password.
Format	An aggregate object consisting of User_ID and Passwd of type string.

Name	Authentication Response
Aliases	None
Where used/ How used	Control Mechanism - Authentication 2.1 (output) Configuration – Check Validity 3.2 (input)
Description	Result of the username, password and administrative rights checking
Format	An aggregate object consisting of User_ID and Passwd of type string and isAdmin of type boolean.

Name	HTTP Request
Aliases	Raw HTTP Request
Where used/ How used	Filtering – Parse 1.1 (input/output) Filtering – Plugin Engine 1.2 (input) Client (output)
Description	Request coming from users is sent to Parser and Plugin Engine to be filtered.
Format	An HTTP request object consisting of Line, Header and Body

Name	Processed HTTP Request
Aliases	None
Where used/ How used	Filtering – Dispatcher 1.3 (output) Server (input)
Description	Filtered HTTP Request
Format	An HTTP request object consisting of Line, Header and Body

Name	HTTP Response
Aliases	Raw HTTP Response
Where used/ How used	Filtering – Parse 1.1 (input/output) Filtering – Plugin Engine 1.2 (input) Server (output)
Description	Response coming from server is sent to parser and plugins to be filtered.
Format	An HTTP response object consisting of Line, Header and Body

Name	Processed HTTP Response
Aliases	None

Where used/ How used	Filtering – Dispatcher 1.3 (output) Client (input)
Description	Filtered HTTP Response is sent to client.
Format	An HTTP response object consisting of Line, Header and Body.

Name	Request/Response Info
Aliases	None
Where used/ How used	Filtering – Plugin Engine 1.2 (output) Logging – Get Connection Information 4.1 (input)
Description	Information to be logged after the analysis
Format	Request or Response object

Name	IP Info
Aliases	None
Where used/ How used	Filtering – Plugin Engine 1.2 (output) Control Mechanism – Authentication 2.1 (input)
Description	To check User filtering rules Plugin sends the user IP to Control Mechanism.
Format	IP is composed of 4 integers.

Name	User Info
Aliases	User Profile Info
Where used/ How used	Filtering – Plugin Engine 1.2 (input) Control Mechanism – Decision 2.2 (input/output) Control Mechanism – Authentication 2.1 (output) Control Mechanism – Rule Process 2.3 (input) Configuration – Send information 3.3 (input/output) Configuration – Get information 3.1 (input) Logging – Check Validity 4.2 (input) Database (input/output)
Description	User's information will be passed through the modules
Format	User Object consisting of User_ID, Group_ID of type string.

Name	Rule Info
Aliases	None
Where used/ How used	Filtering – Plugin Engine 1.2 (input) Control Mechanism – Decision 2.2 (input/output) Control Mechanism – Rule Process 2.3 (output) Configuration – Send information 3.3 (input/output) Configuration – Get information 3.1 (input) Database (input/output)
Description	A new rule is added to the system over the web interface by the Administrator. Rules are stored in the DB and they are sent to 'Configuration Module' to get into action.
Format	An object containing the Rule attributes. The object may vary depending on the plugin that the rule is created from.

Name	Decision and Packet Info
Aliases	None
Where used/ How used	Filtering – Plugin Engine 1.2 (output) Filtering – Dispatcher 2.2 (input)
Description	Actions and filtered HTTP packets are sent to Dispatcher in order to deliver related client and server.
Format	HTTP Request/Response Object

Name	Change Notification
Aliases	None
Where used/ How used	Control Mechanism – Rule Process 2.3 (input) Configuration – Send Information 3.3 (output)
Description	When a new rule is added to the system, its information will be sent to the Rule Process to use the rule immediately.
Format	An object containing Rule identifier.

Name	Configuration Info
Aliases	None
Where used/ How used	Configuration – Get Information 3.1 (input) Configuration – Send Information 3.3 (output) Administrator (output) Database (input/output)
Description	Administrator makes changes to the system configuration. Admin can change group, user and rule interrelations over the web. The changes are written to DB.
Format	Configuration change object

Name	Log and Statistical Info
Aliases	None
Where used/ How used	Logging Process – Send Information 4 (input/output) Administrator (input) Database (input/output)
Description	The logs and statistical information from these logs will be stored in the database and showed to Administrator.
Format	A Log object containing the User_ID/Rule_ID, Site, Domain, Bandwidth and Action_Time (Open_Time, Close_Time), Action_Desc attributes. These are explained in Data Objects part.

Name	Alert
Aliases	None
Where used/ How used	Logging Process – Send Information 4.3 (output) Administrator (input)

Description	On predefined conditions depending on the actions and the logs, the Logging Process module will send a notification e-mail to Administrator.
Format	A text that describes the action or situation briefly.

Name	Action
Aliases	None
Where used/ How used	Filtering – Plugin Engine 1.2 (output) Logging Process – Get Connection Information 4.1 (output)
Description	One of the 4 predefined actions is returned by the Plugin according to the action performed.
Format	Enumerated value corresponds to the action taken: ACTION_BLOCK, ACTION_ANSWER, ACTION_REDIRECT, ACTION_FORWARD

Name	Analyzed Info
Aliases	None
Where used/ How used	Configuration – Get Information 3.1(output) Configuration – Check Validity 3.2(input)
Description	After the incoming information is analyzed, it is sent to be validated with the authentication info.
Format	One of the Configuration, User, Rule objects

Name	Validated Info
Aliases	None
Where used/ How used	Configuration – Send Information 3.3(input) Configuration – Check Validity 3.2(output)
Description	After the information is validated, it is transferred to Send Information module.
Format	One of the Configuration, User, Rule objects

Name	Connection Info
Aliases	None
Where used/ How used	Logging – Get Connection Information 4.1(output) Logging – Check Validity 4.2(input)
Description	Connection Information is sent to Check Validity 4.2 module
Format	Request or Response Object and User identifier.

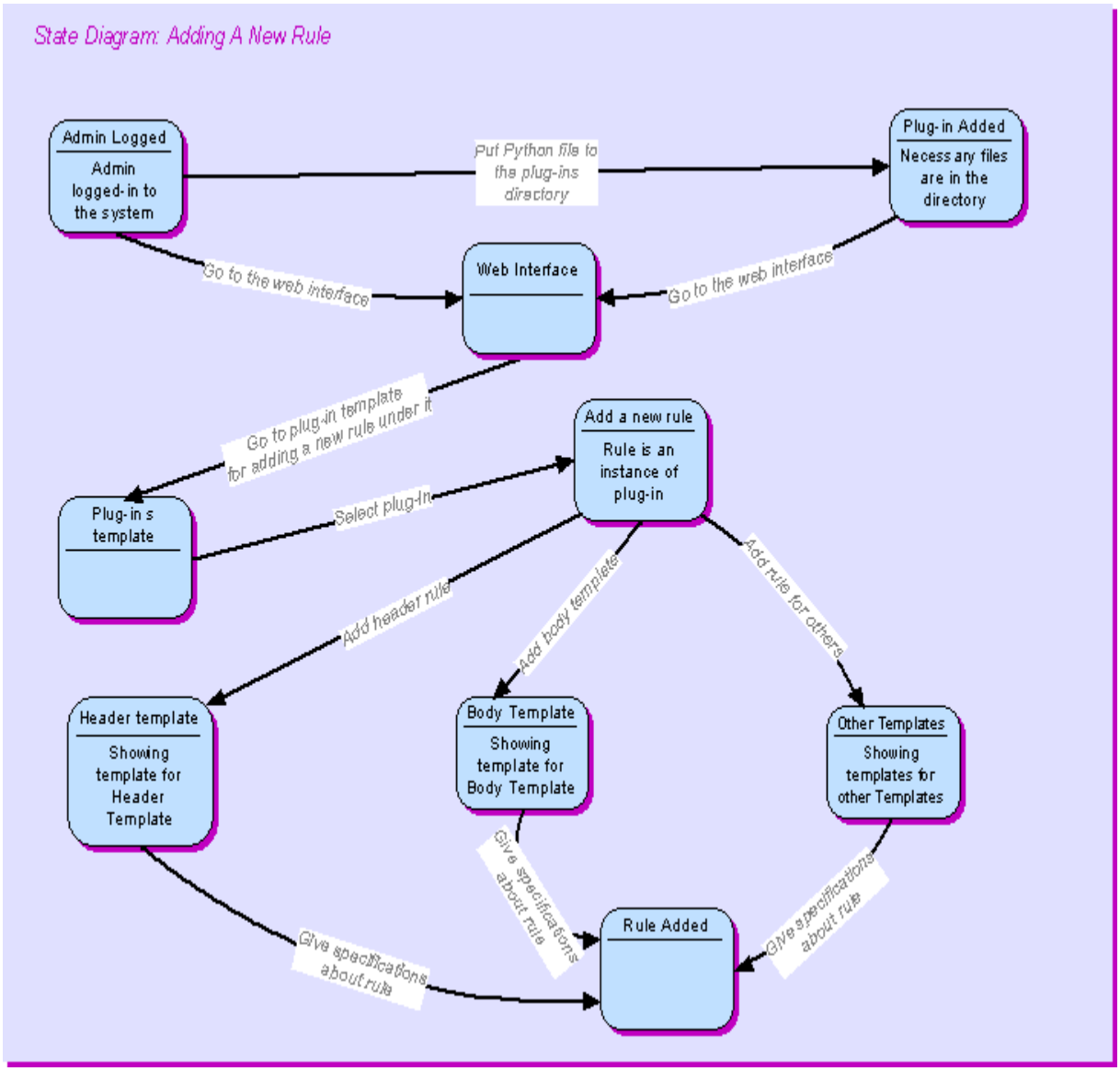
Name	Info
Aliases	None
Where used/ How used	Logging – Send Information 4.3(input) Logging – Check Validity 4.2(output)
Description	After the information is validated, it is transferred to Send Information module.
Format	Request or Response Object and User identifier.

3.3 State Transition Diagrams

3.3.1 State Diagram Adding a Rule:

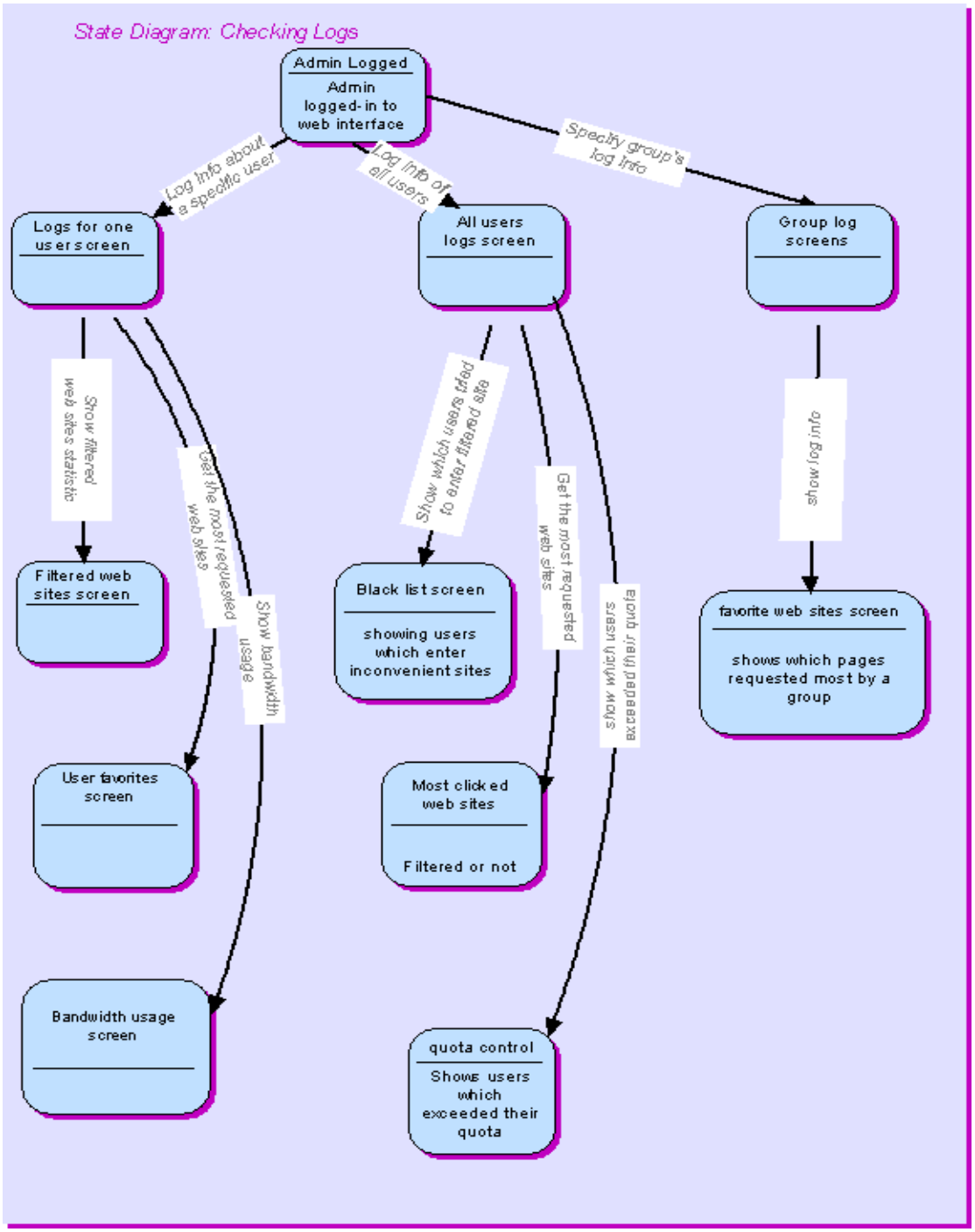
The figure given below explains how a rule is added to the system by the administrator by using web interface. Adding a plugin is represented below, too.

A rule is a particular instantiation of a plugin

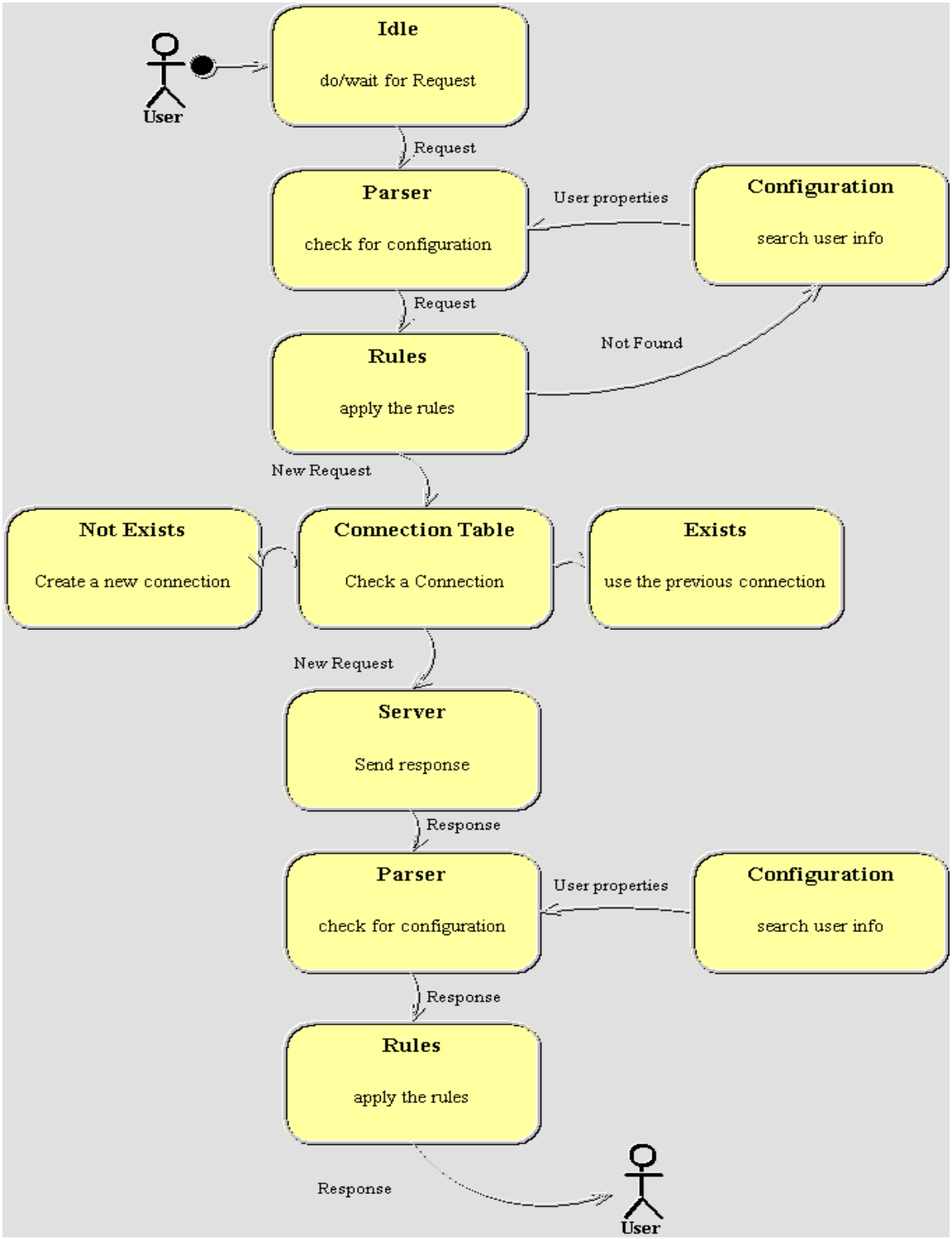


3.3.2 Checking Logs State Diagram:

The diagram below gives the state diagram for checking logs, starting from the administrator web interface.



3.3.3 Request – Response Diagram



- When a new Request comes, IronCurtain will create a new thread to our thread pool.
- Then it will send request to our parser.
- Parser will get information from configuration module. Information is about which rules to apply for this user? The bandwidth limit of the user. The time limit of the user.
- After getting information about user parser sends these to Rules. In rules module it will select the proper rules and apply them to the request in parallel.
- Then Rules will send the updated request to the connection table.
- There, the connection will be checked if it is opened before, it will use the previous connection. (HTTP 1.1 persistent connection property) If not it will create a new connection.
- Then the request will be sent to the server.
- Server will send the response.
- The response will be taken by Parser. It gets the information from configuration module.
- Then it will send the response to Rules.
- Rules will apply the proper operations to the response and then send the new response to the user.

3.4 Description of Components

3.4.1 Plugin Architecture

3.4.1.1 Structure of a Plugin

A plugin is a Python file with a few quirks. In the file, there will be one class with the same name as the file (case-sensitive), which defines at least one function act_on with no arguments, and a static variable called Args. Also, the constructor of the class must take one argument, a dictionary (also known as, associative array).

act_on: is the function that is called when the rule works. It can optionally return a constant value or a tuple to indicate the course of action for IronCurtain.
The constants are:

ACTION_BLOCK: A blocked request is rejected; the browser's connection is closed and the proxy returns an error page or a "broken image" icon.

ACTION_ANSWER: An answered request is handled directly by the plugin. In a sense, the proxy acts as a web server, and sends the web page that it created to the client by itself. If the action is ACTION_ANSWER, web page is returned as the second argument of a tuple.

ACTION_REDIRECT: A redirected request is sent to a location other than for what location it was originally intended. In this case, the second argument of the tuple will be the new location to be redirected.

ACTION_FORWARD: A forwarded request is sent to the web server for which it was originally intended.

If act_on function does not return a value, the action taken is ACTION_FORWARD.

Args: should be declared as a list of tuples that lists the arguments to the rule.

Args is a mapping between argument names and their types. An element in the list is a 3-tuple or a 4-tuple.
The first is the actual name of the argument.
The second is the text as it will look on the HTML Page.
The 3rd and 4th are indicators of data types. Only 4 structures are allowed: int, string, enum, and array. The first two (int and string) are obvious.
The third, enum, is an n-tuple of strings. It represents a group of choices of which only one can be selected.
Last, array is a list of strings. Two numbers follow array declaration to indicate its size. The first is the minimum size of array, the second is the maximum. If the second is omitted, it is assumed that maximum is infinity.

The constructor of the class: The constructor should be written to expect only one argument: a dictionary. It will contain the instances of arguments defined by Args.

3.4.1.2 Activating the Plugin

After the coding is completed, we upload the plugin over IronCurtain's web interface. This creates a new directory under plugins/ directory with the name of the plugin. Under that directory,

IronCurtain will put the plugin file and a hidden XML file with the name <plugin_name>_$schema$.xml. The schema file is generated from the Args variable.

Once this is done, the new plugin will be available in the web interface. The administrator will be able to instantiate new rules from this plugin. All rules are stored under the rules/ directory. Rule is stored as a XML file. The name of the XML file will be "<plugin_id>_<rule_id>.xml". At the next startup, IronCurtain will scan the rules/ directory and start all active rules.

3.4.1.3 An Example Plugin and Rule

Let's go over the process of creating a new rule.

We want to write a rule that changes the value of the "User-Agent" header in an HTTP request to "Protoxy/0.0.1" if the site is in a German or French domain. Let's assume that no plugin has the necessary infrastructure for such a task (of course, in Protoxy there will be such an infrastructure), so we will also write a plugin.

Args = [("MatchURLs", "Match URLs", "array", 1), ("Actions", "Actions", "enum", ("Add", "Replace", "Delete")), ("HeaderName", "Header Name", "string"), ("HeaderValue", "Header Value", "string")]

(Assuming such a declaration) def __init__(self, Params):

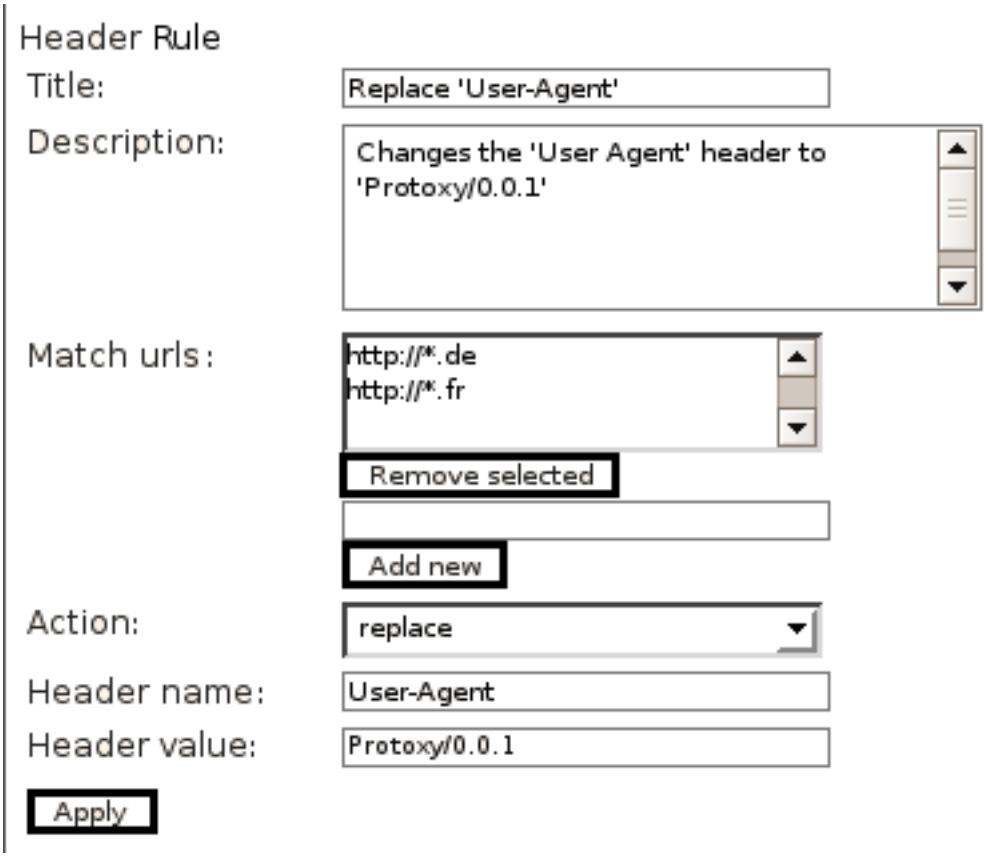
Params["Actions"] -> "Replace"
Params["HeaderName"] -> "User-Agent"
Params["HeaderValue"] -> "Protoxy/0.0.1"
len(Params["MatchURLs"]) -> 2
Params["MatchURLs"][0] -> "http://*.de"
Params["MatchURLs"][1] -> "http://*.fr"

Once we are happy with our code, we upload it as a new plugin over IronCurtain's web interface. This creates a new directory under plugins/ directory with the name of the plugin. Let's assume we named it HeaderPlugin. Under that directory
it puts the file HeaderPlugin.py and a hidden xml file with the name HeaderPlugin_$schema$.xml. The schema file is an explicit from the Args variable. In our case, it would look like this;

```
<Plugin ID="1">
   <MatchURLs displayname="Match urls" type="array" minsize="1"/>
   <Actions displayname="Actions" type="enum" choices="3">
      <choice>
```

```
        Add
      </choice>
      <choice>
        Replace
      </choice>
      <choice>
        Delete
      </choice>
  <HeaderName displayname="Header Name" type=string/>
  <HeaderValue displayname="Header Value" type=string/>
</Plugin>
```

Once this file is generated, the web interface will show the new plugin immediately. To add the rule we select the ‘Add New Rule’ tab and click on our previously defined plugin as the template.



When we click the apply button, the web interface generates an XML document to store the relevant data. The document is put in under rules/. Our rule would look like this:

```
<Rule pluginID=”1” ID=”1” active=”true”>
   <Title type="string">
      Replace 'User-Agent'
   </Title>
   <Description type="string">
      Replaces 'User-Agent' header to 'Protoxy/0.0.1
   </Description>
   <MatchURLs type="array" size="2">
      <element type="string">
         http://*.de
      </element>
      <element type="string">
         http://*.fr
      </element>
    </MatchURLs>
    <Actions type="string">
      Replace
    </Actions>
    <HeaderName type="string">
      User-Agent
    </HeaderName>
    <HeaderValue type="string">
      Protoxy/0.0.1
    </HeaderValue>
</Rule>
```

3.4.2 Default Plugins

IronCurtain's flexible plugin architecture allows easy addition of new capabilities. This way, IronCurtain will have the flexibility of having new functions, but there will be some plugins that will be shipped along the IronCurtain package by default.

The plugins themselves will have default fields that will be present in every plugin. These fields are as follows:

- "Title" : The title of the created rule.
- "Description" : The description of the created rule, in plain text. This can be empty.
- "URLs to Apply" : The list of the URLs to which the rule will be applied. If this field is empty; any URL not specified by “URLs not to Apply” field will be matched. This behaviour may be replicated by putting a “*” in the field.
- "URLs not to Apply" : The rule will not apply to the URLs in this list.

Only one of these two fields can be active in a rule. Either the rule is applied to specific web sites and no other; or the rule is applied to all web sites, excluding those in the list.

- “Tags to Apply” : When defining a rule, the administrator can select a number of tags from this list. Then the rule will apply to sites having all of those tags, thus we define an 'and' relation between selected tags.
- “Tags not to Apply” : By selecting a number of tags, the administrator may choose not to apply a rule to sites having one or more of the selected tags, thus we define an 'or' relation between selected tags.

Like URLs, only one of these tag related fields can be active.

- “UserIDs” : The user ids that this rule will apply to. The field will be a list to choose user Ids from, and there will be an “invert” button to invert the selected users of the list.
- “GroupIDs” : The group ids that this rule will apply to. Its behaviour will be like “UserIDs” field.

If there are both GroupIDs and UserIDs present in the rule, there will be an ‘or’ operation between them. The rule will apply to members of given groups, also it will apply to given users.

- “Alert”: This will be a checkbox and if it is checked, actions of the plugin will be sent to the administrator by e-mail. The administrator e-mail address can be changed from web configuration interface.

The followings are the default plugins in our proxy.

3.4.2.1 Tagging(Categorization)

This plugin will allow human-guided categorization of web sites. Using this plugin the administrator will be able to add any number of tags to any site.(For example, an administrator can tag www.ntvmsnbc.com as “news”, and www.trgamer.com as “games” and “news”.) Tagging is part of the core proxy, and every rule may use tagging facility, this plugin provides an easy to use interface to the tagging facility.

3.4.2.2 Modify Header

This plugin is used to change HTTP request/response headers. This plugin may be used to instantiate rules that add a new header, remove an existing header or change the content of a

header. Its fields are:

- “Header Name”: This is the name of the header to act on.
- “Action”: There are three possible actions:
    - “Add/Change”: This is used to add a new header to the request/response. If the given header is present, then its content is changed to the string specified in “Header Value”.
    - “Remove”: This is used to remove an existing header.
    - “Block”: If the header name and the header value matches, deny access to this site.
- “Header Value”: This is the value of the header to add or change. This field is not used if the “Remove” action is selected.

3.4.2.3 Modify Content

This plugin is used to change the body of the HTTP request/response(The body of a request is generally only useful for the HTTP POST method. It is used to send things like login data. The body of a HTTP response is generally the HTML web page). Its fields are:

- “Tag Name” : The HTML tag
- “Attribute Name” : The attribute of the HTML tag
- “Attribute Value” : The value of the attribute
- “Enclosing Block” : The tag name of the block that encloses the given tag
- “Action” : There are three possible actions.
    - “Remove” : Removes the field completely.
    - “Replace” : Changes the field with the new given value.
    - “Block”: If the selected field (Tag Name, Attribute Name, Attribute Value, Enclosing Block) matches it will deny access.
- “Action on Part” : Plugin will act on one of the Tag, Attribute Name, Attribute Value, Enclosing Block fields.
- “Replace Value” : New value to modify the selected field.

3.4.2.4 Bandwidth Limiter

This plugin is used to adjust per-user quotas. Looking at the Content-Length in HTTP Header and the Header data size, the bandwidth usage will be calculated. A tag based or a global quota can be set for users and groups. Its fields will be:

- “Quota” : The total web traffic quota in MBs.
- “Period”: This is used to set the period of the quota usage, in days.

3.4.2.5 Image

This plugin is used as a general image blocker and modifier. It can block animated gifs or images according to user defined size.

Plugin's fields:

- "Image size" : The plug-in will block/replace images of given size. If present the size will be extracted from "img" tag, otherwise the image will be downloaded and its size will be taken from the file.
- "Image URL" : URL of the image.
- “Actions” : There are different actions that can be performed on the image.
    - Block: The image will be replaced by a 1x1 transparent image.
    - Hide: The image will be replaced with another image of same size stating that there is a blocked image. This image will be a link and when clicked the page will be reloaded, with that image being shown.
    - BlockAnimation: Only the first frame of animated gif image will be displayed.
    - Optimize: The image file size is reduced by compressing the image.

3.4.2.6 Blocker

This plugin allows general purpose blocking of web sites, based on URL or tags. It will have no additional field besides those already provided as default. This plugin always returns ACTION_BLOCK.

3.4.2.7 JavaScript Filtering

This plugin is used for filtering the JavaScript code that is present in the web pages. It will be possible to remove certain functions and calls to those functions. This plugin will not parse the

JavaScript code in a web page. Instead, this plugin will use a pattern matching technique to remove or alter parts of the JavaScript code. Its fields are:

- "Pattern to Match": This is a regular expression to match in the JavaScript code.
- "Replacement Text": This is the text to replace the matched pattern.

3.4.2.8 Keyword Filtering

It is similar to the Modify Content plugin. It will modify all leaves under the "body" tag except the ones under the "script" tag in the HTML DOM tree. Its fields are:

- "Pattern to Match" : This is a regular expression to match in the text of the HTML page.
- "Replacement Text" : This is the text to replace the matched pattern.
- "Action" : There are two possible actions.
    - "Replace" : It will replace the matching pattern with the replacement text.
    - "Block" : If there is a matching pattern, it will deny the access.

3.4.3 Capabilities of Default Plugins

These set of default plugins provide IronCurtain with the following capabilities.

3.4.3.1 Ad Blocking

There are three main kinds of ads on the internet; image ads, text ads and flash ads. Image ads usually consist of images of jpg, gif or png format of a fixed size (120x120, 125x125, 234x60, 468x60, 600x120) IronCurtain's "Image" plugin can be used to block images of these sizes. This will eliminate many of the image ads.

Also, URL based filtering will be applied by regular expressions of previously discovered URLs that the ads com from. Links that point to those URLs will be removed from the page using "Modify Content" plugin. This will prevent both text and Image ads from displaying. Flash ads will be blocked similarly, flash files coming from URLs that match the regular expression will not be shown. Also the administrator may choose to display a "Blocked" image instead and show the flash file upon clicking on this image.

3.4.3.2 Pop-up Blocking

Pop-ups sometimes may be annoying, especially the ads. These are generally displayed using

JavaScript(**window.open(url, ...)** function). Using our JavaScript filtering plugin, IronCurtain can catch and remove these kinds of pop-ups.

3.4.3.3 Blacklisting

Blacklisting can be done easily using the Blocker plugin. Simply choose the tags or URLs you want to block.

3.4.3.4 Java Blocking

Modify Content can be instantiated to block Java applets. The technique that we will use is; if there is class file in the “object” or “embed” tags, it will delete that tag from HTML code, effectively blocking the Java applet from executing.

3.4.3.5 Flash Blocking

Flash files need to have “swf” extension to run on the browser. If there is swf file in the “object” or “embed” tags, Modify content will delete that tag from the HTML code, thus the flash objects present in the page will be blocked.

3.4.4 Authentication

IronCurtain will allow rules to be applied for some users/groups only. This brings the requirement that there should be a mechanism to identify which user's web traffic is being analyzed. Our plans consisted of using NTLM authentication, which Internet Explorer and Firefox browsers support. But our research revealed that it would not be an appropriate way of authenticating users, because NTLM support of Firefox is not geared towards proxy authentication use, and we would need an Active Directory server.

We may use standard proxy authentication scheme of HTTP (“Authorization” header) to get user name and password or we would get authentication info from the user by using a secure web page. The first method required the user to enter information every time they reopened the browser, also password is transmitted in clear text, which is insecure. So we decided to take the second approach, presenting the user with a login screen and associating the user by his/her IP address.

Upon a connection attempt, IronCurtain will check the source IP address of the connection. If the address is already associated with a user, rules that apply to that user will be activated and the logs will recorded on that user. If the address is not associated with any user, we will redirect the user to the login screen, where they would be able to enter their user name and passwords. This

page will be an SSL enabled page, to prevent clear text transmission of passwords. The password entered into the page will be hashed, and the hash value will be compared to the stored password of the user in the database. If two hashes match, the user has successfully logged into the system. After a successful login, the IP address will be associated with that user for a fixed period of time; its default value will be 8 hours, and it can be changed via configuration interface. The associated IP-user information will be held in memory, using a python dictionary. If the user enters a wrong user name or password, he/she will be requested to enter both the username and the password again. This will prevent a malicious person from discovering a valid user name and conducting a brute force attack on the proxy.

3.4.5 Logging

Logging is the part of the core proxy, and every plugin will have the logging capability. The administrator will have extensive choices what to show from the logs. It will be possible to use the logs to track user activities and plugin actions for technical reviews. Different log viewing schemas will be used per user and group. In the Logging configuration panel, administrator will select what to show for specified users' and groups' logs.

The followings will be selectable to be displayed from user logs:

- Site: This is the full address of the visited web page.
- Site domain: This is the domain of the visited web page.
- UserID: The user name
- Open_Time/Close_Time: These are the action begin and end times.
- DataSize: The data transfer size during the action time.

The followings will be stored for rules themselves:

- RuleID: The identifier of the rule.
- Site: This is the full address of the site that the rule acts on.
- Domain: This is the domain of the Site.
- Action_Time: This is the time of the action.
- Action_Desc: Information about the action taken.

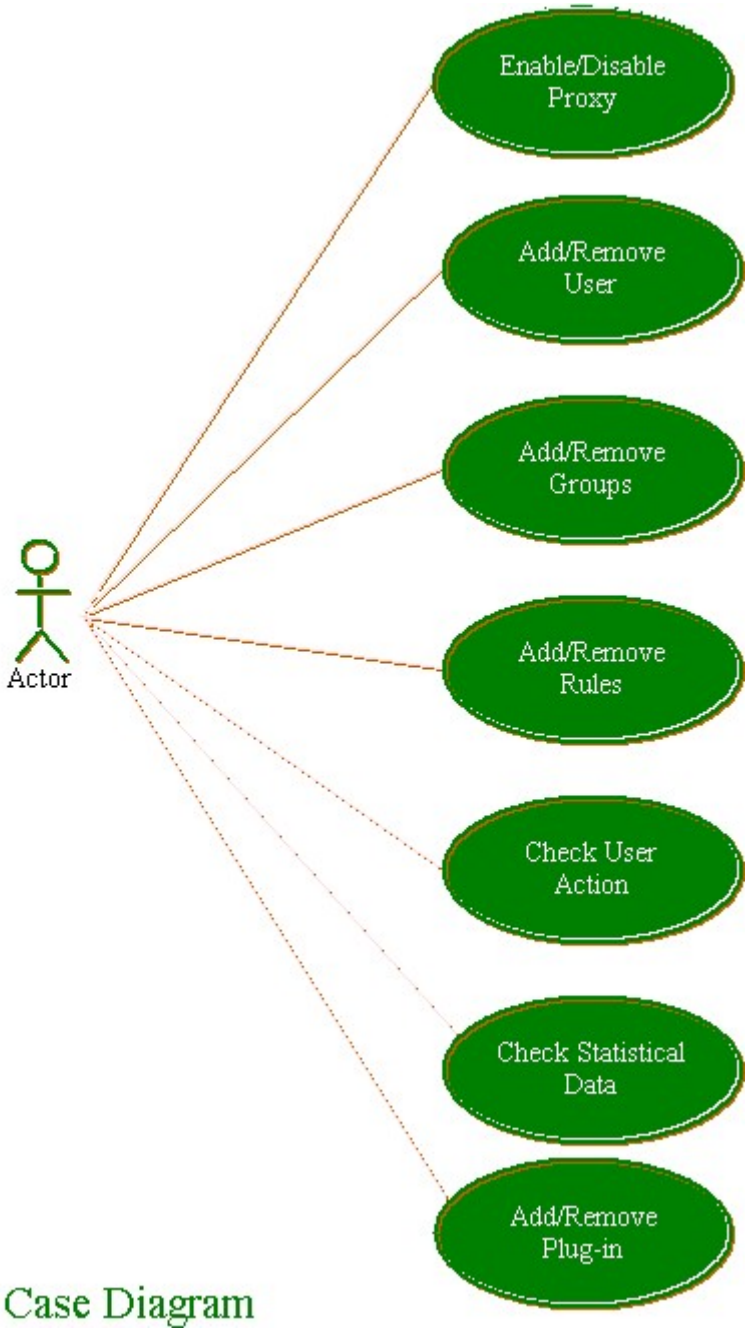
How Logging Works:

Upon arrival of the request and the response the core proxy will set the following fields in the logs:

- UserID: By looking at the request's source IP address and consulting the IP user table that is kept in the memory shared between all threads, IronCurtain will be able to extract user name and log it.
- Site: Request header contains the whole URL of the page to be retrieved. It will be logged along the user name and open time.
- Site Domain: The domain part of the URL will be recorded for easy statistic analysis.
- Open_Time/Close_Time: These are the arrival time of the request to the proxy and the time when the client closes the connection. By using this information, the administrator can get a rough estimate of the duration of the user's visit to a particular site.
- DataSize: The total size of the document(headers and content) which is calculated in a similar way like the bandwidth limiter will be recorded.

The rule will have access to a log function, which will take the description of the action as parameter. The other information that will be logged (RuleID, Site, Domain, Action_Time) is available to the core proxy and will be recorded to the log database.

3.5 Use Case Diagram

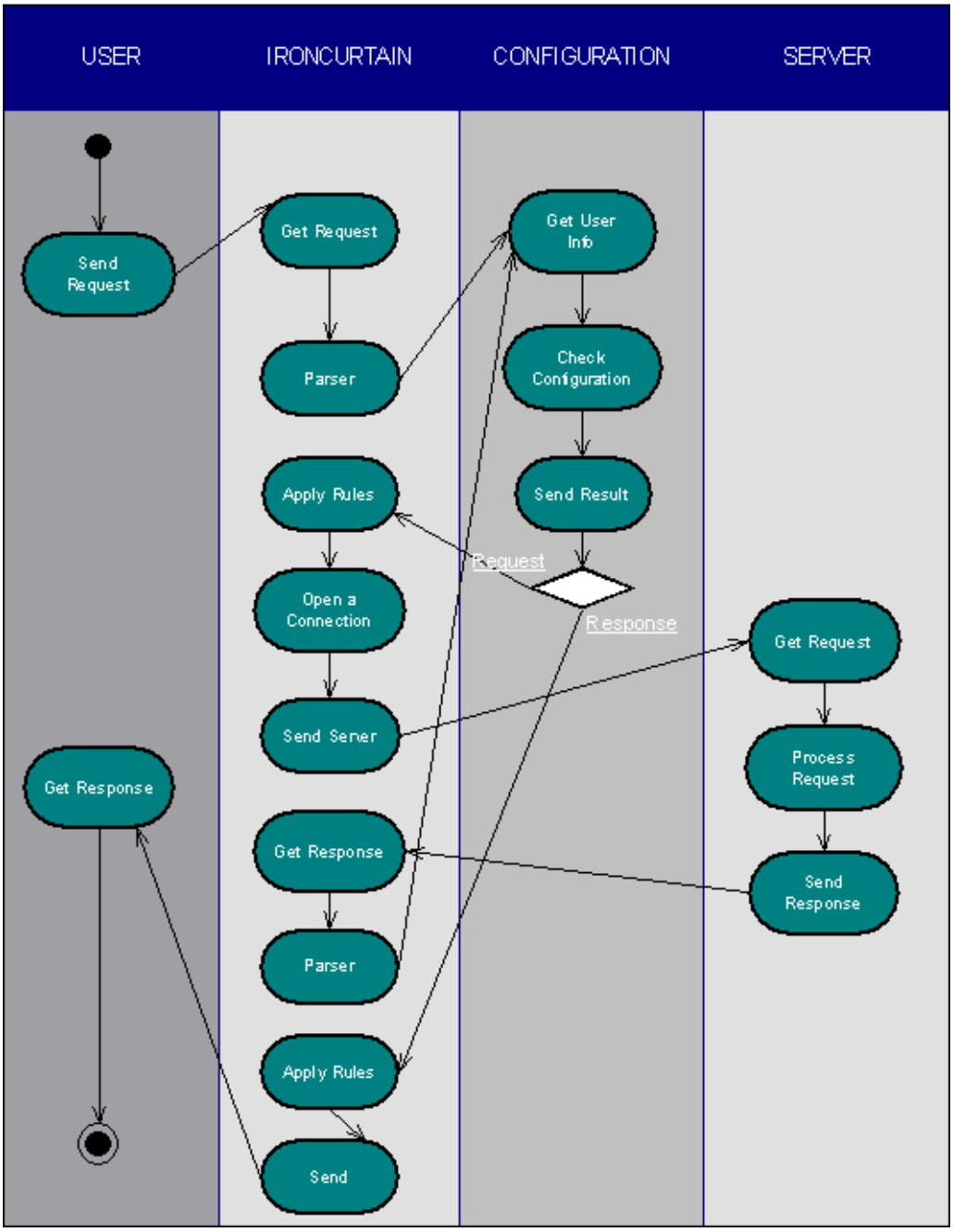


Use Case Diagram

3.6 Activity Diagrams

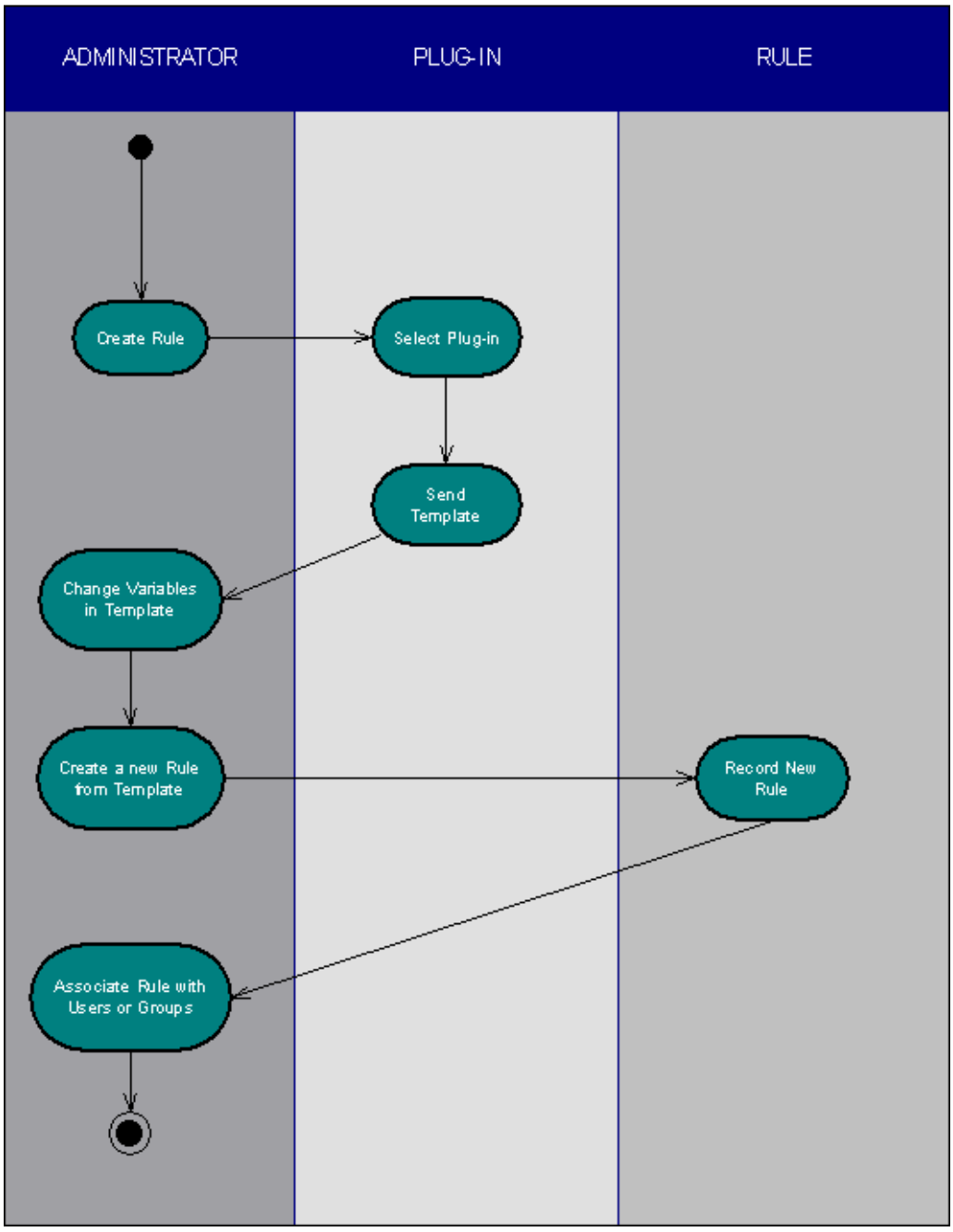
Connection Request-Response

UML Activity Diagram: Connection Request-Response



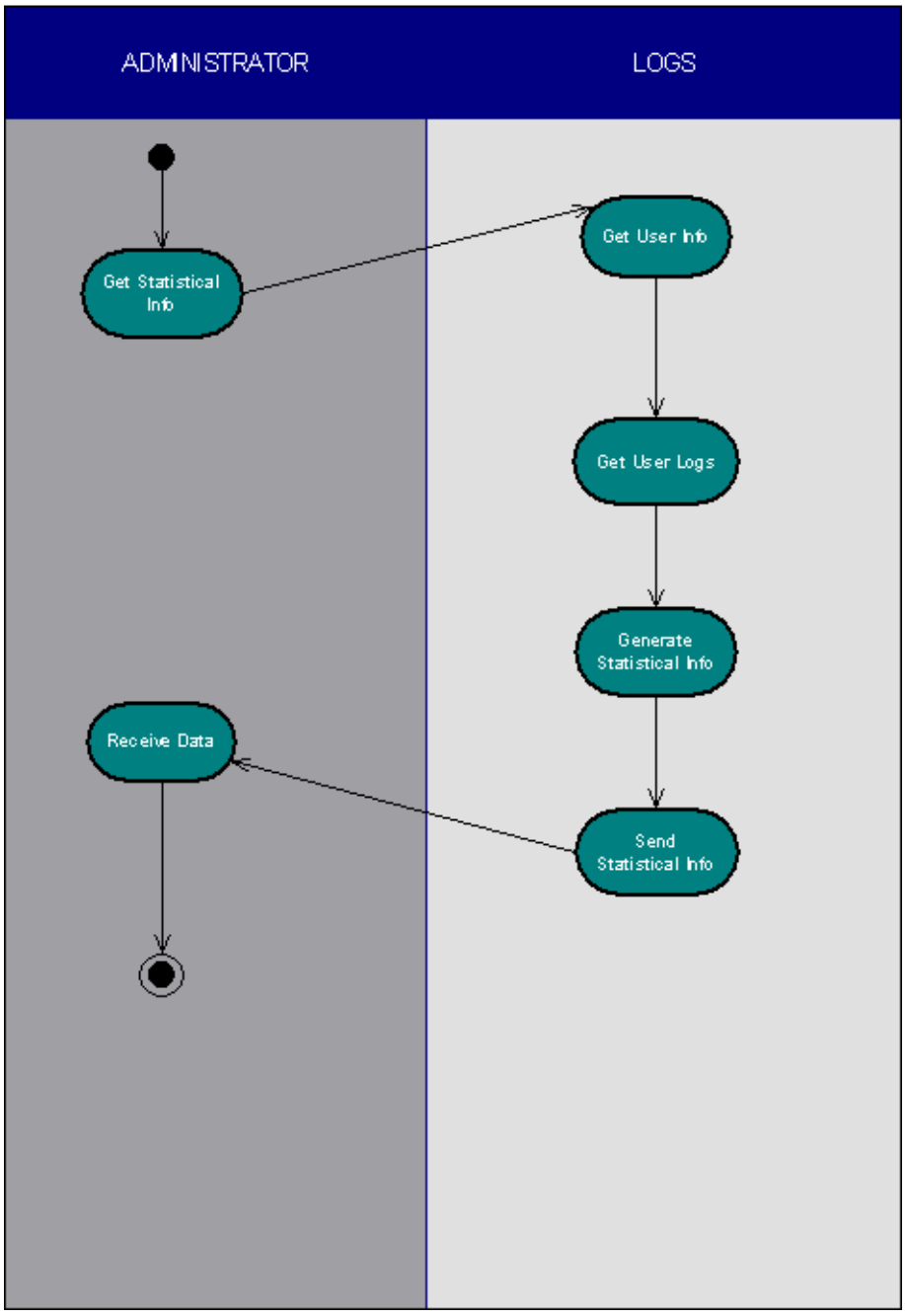
Adding a Rule

UML Activity Diagram: Adding a Rule



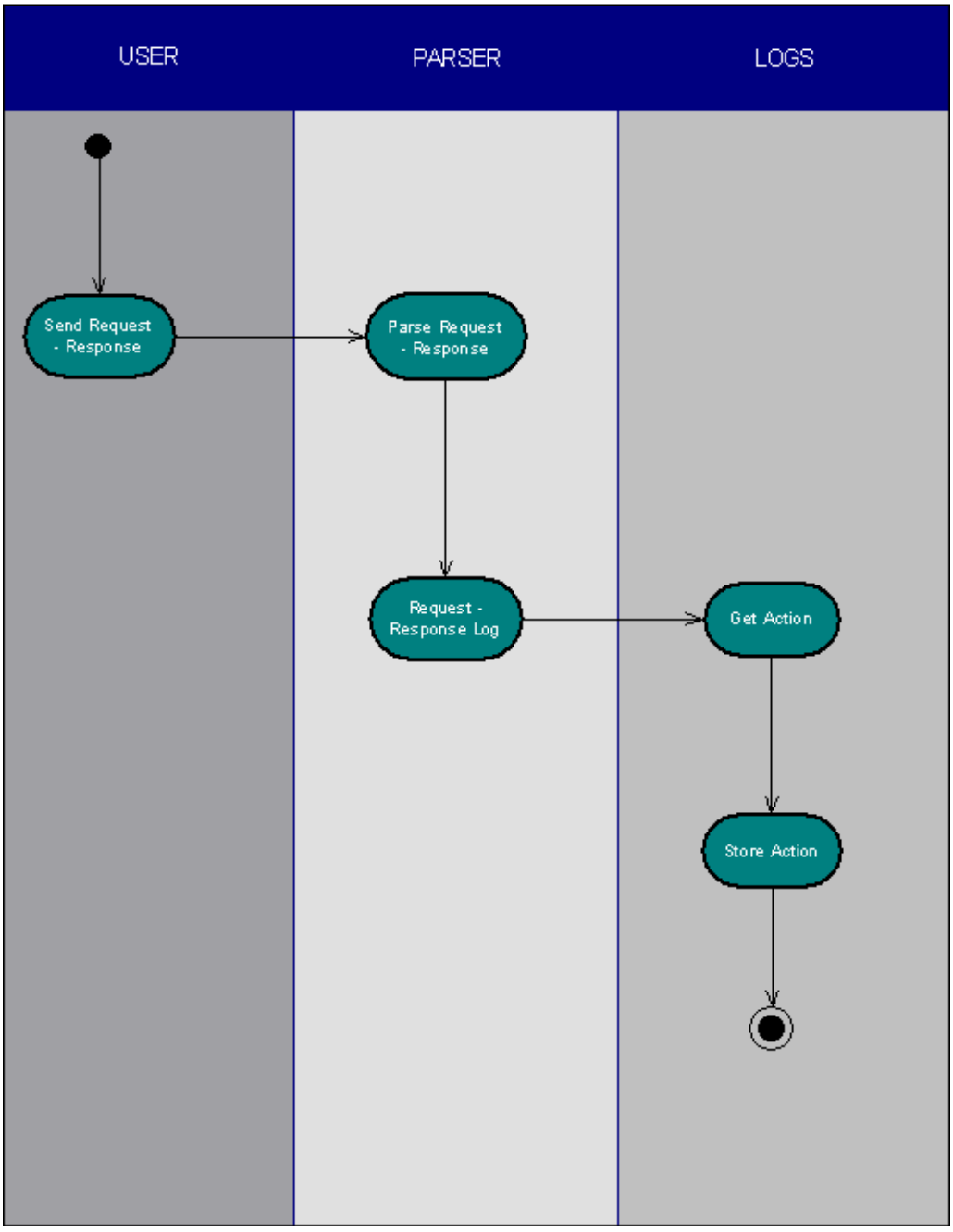
Get Statistical info

UML Activity Diagram: Get Statistical Info

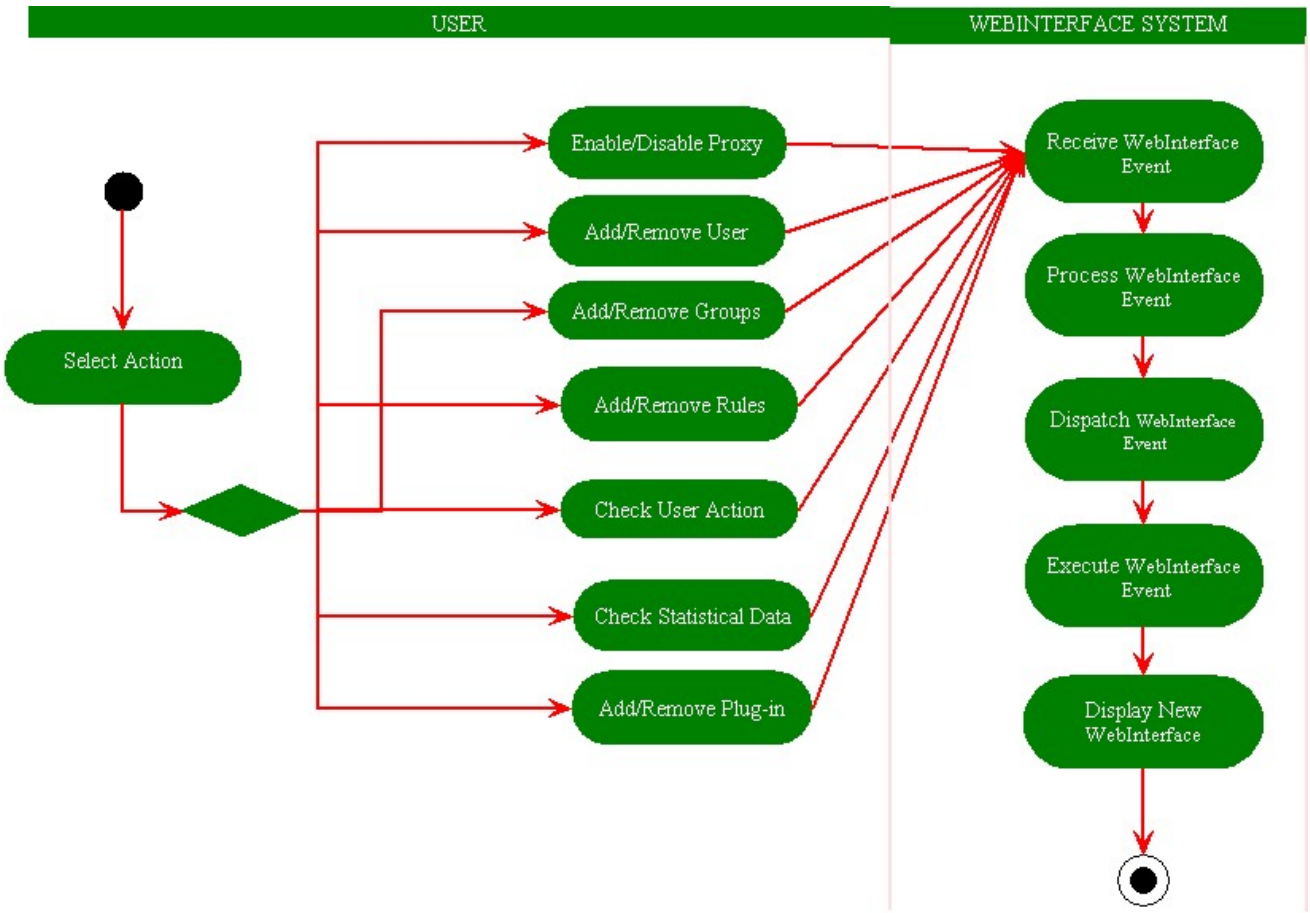


Write Statistical Info

UML Activity Diagram: Store Statistical Information



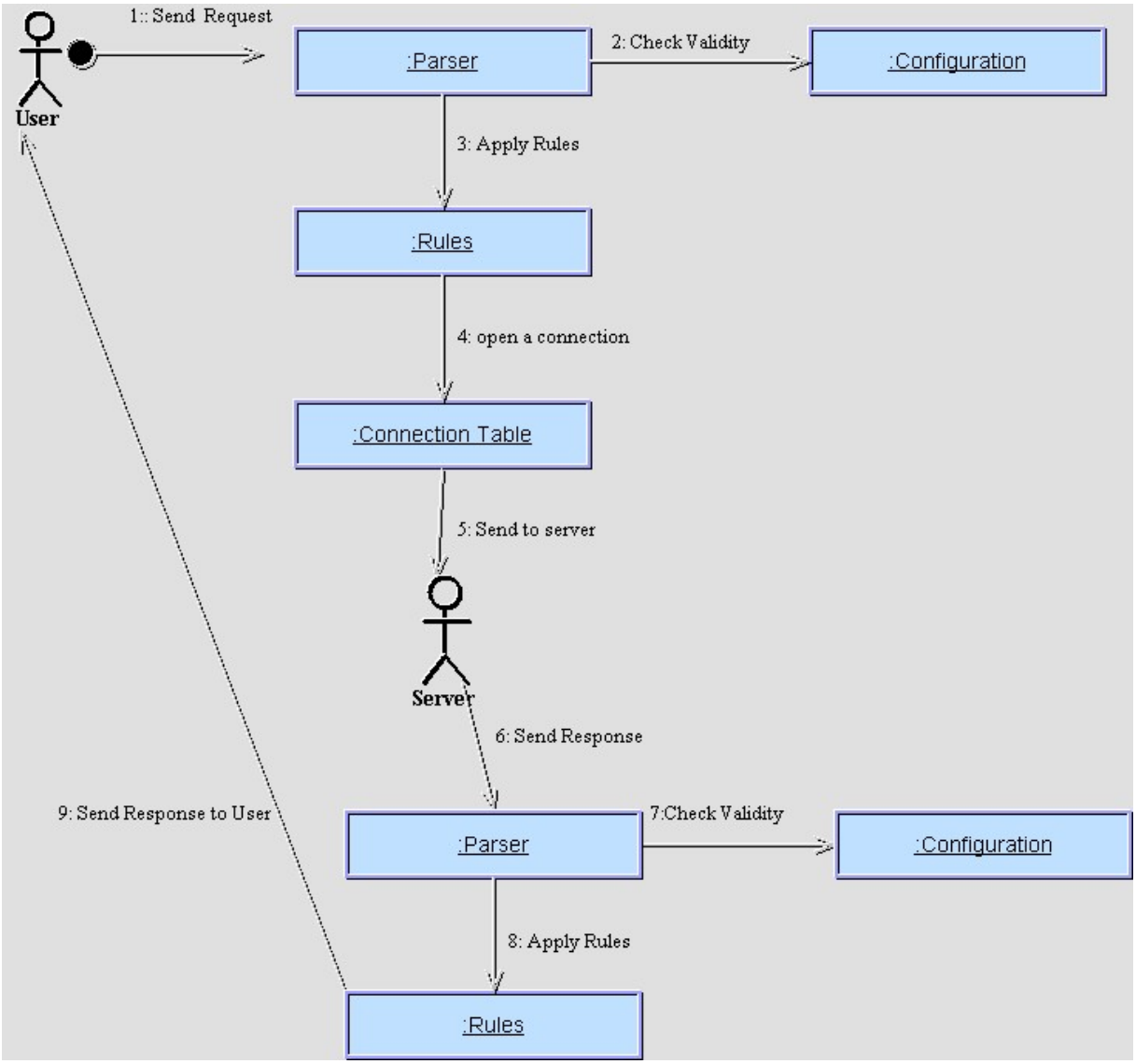
Web Interface Activity Diagram



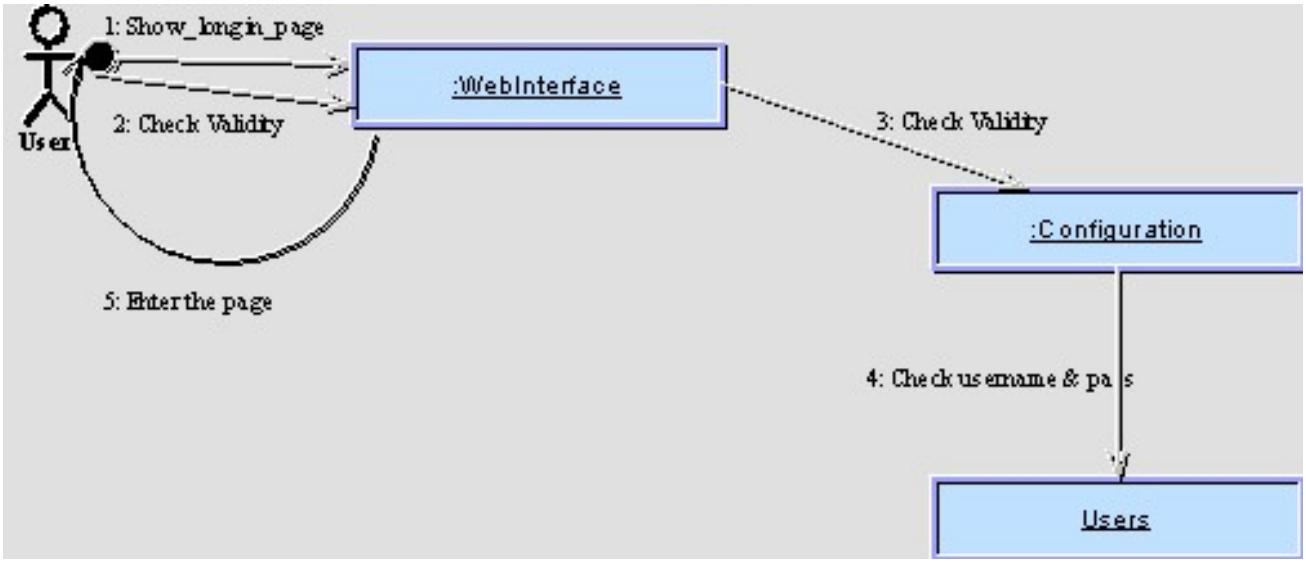
Activity Diagram

3.7 Collaboration Diagrams

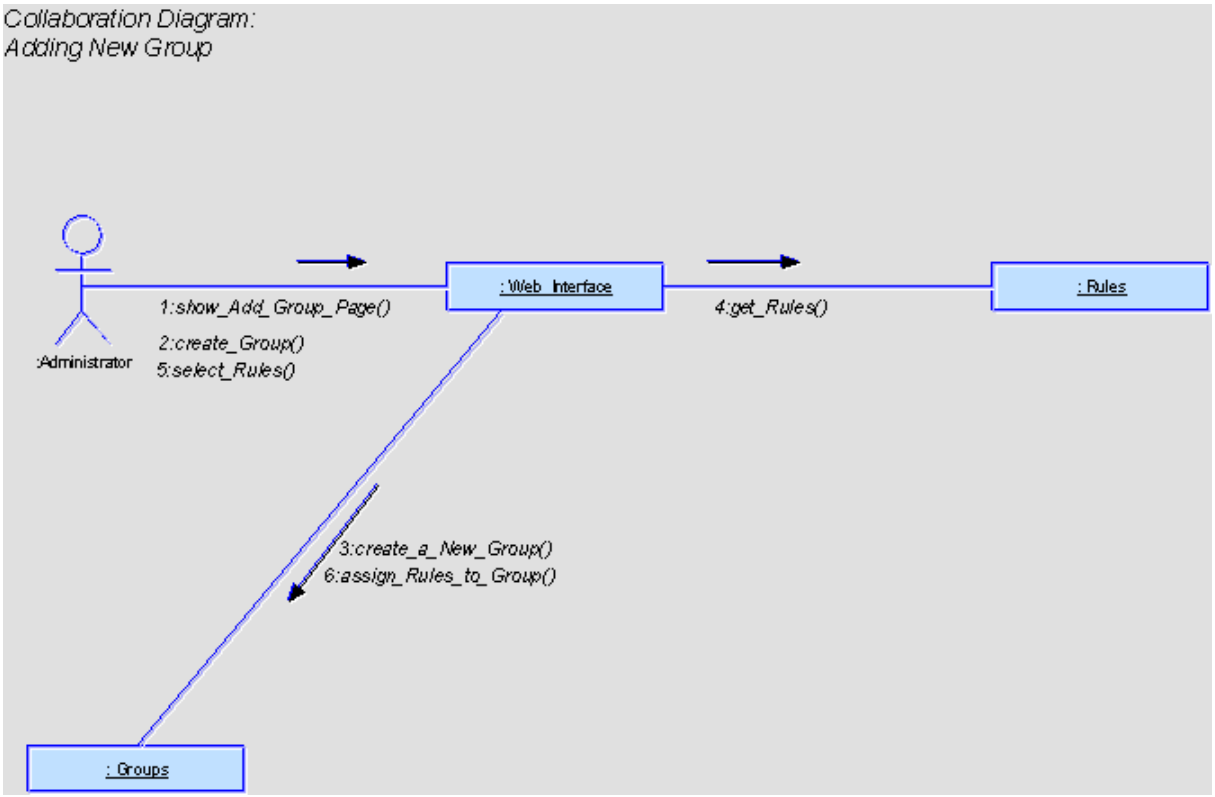
Request / Response Collaboration Diagram



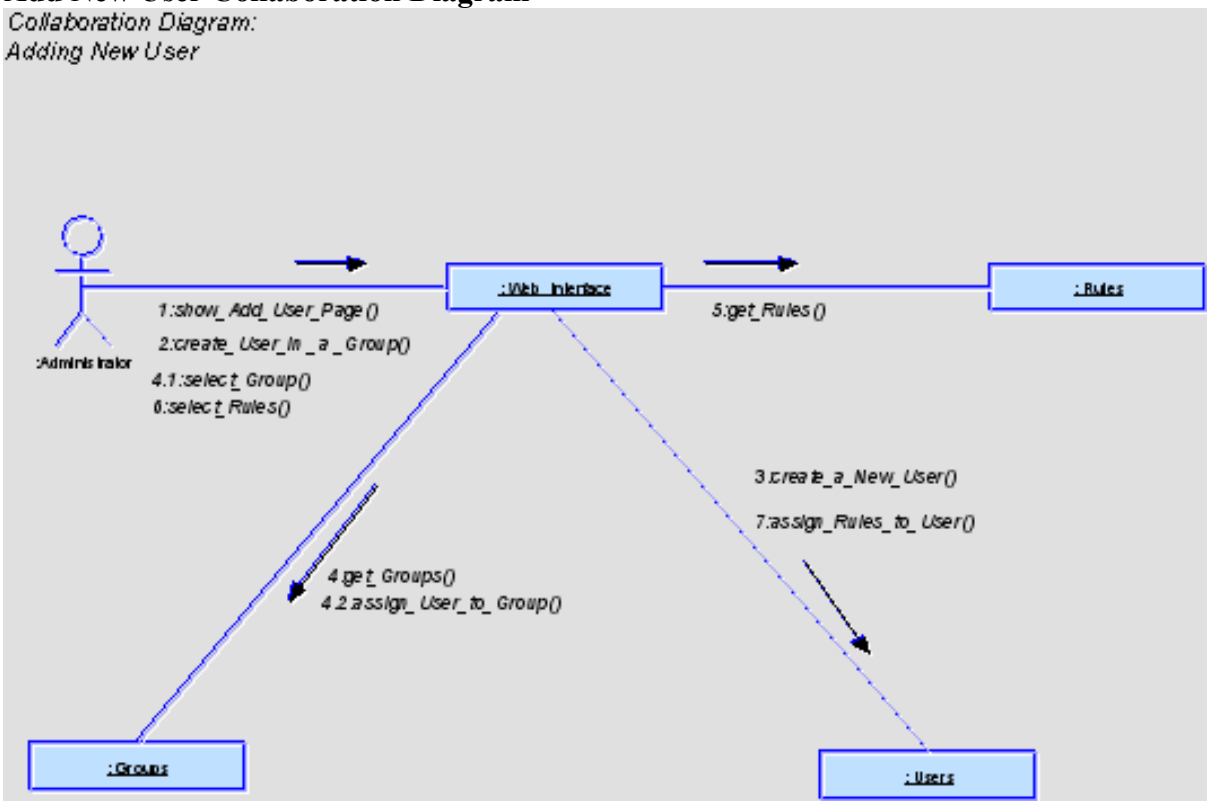
Login Collaboration Diagram



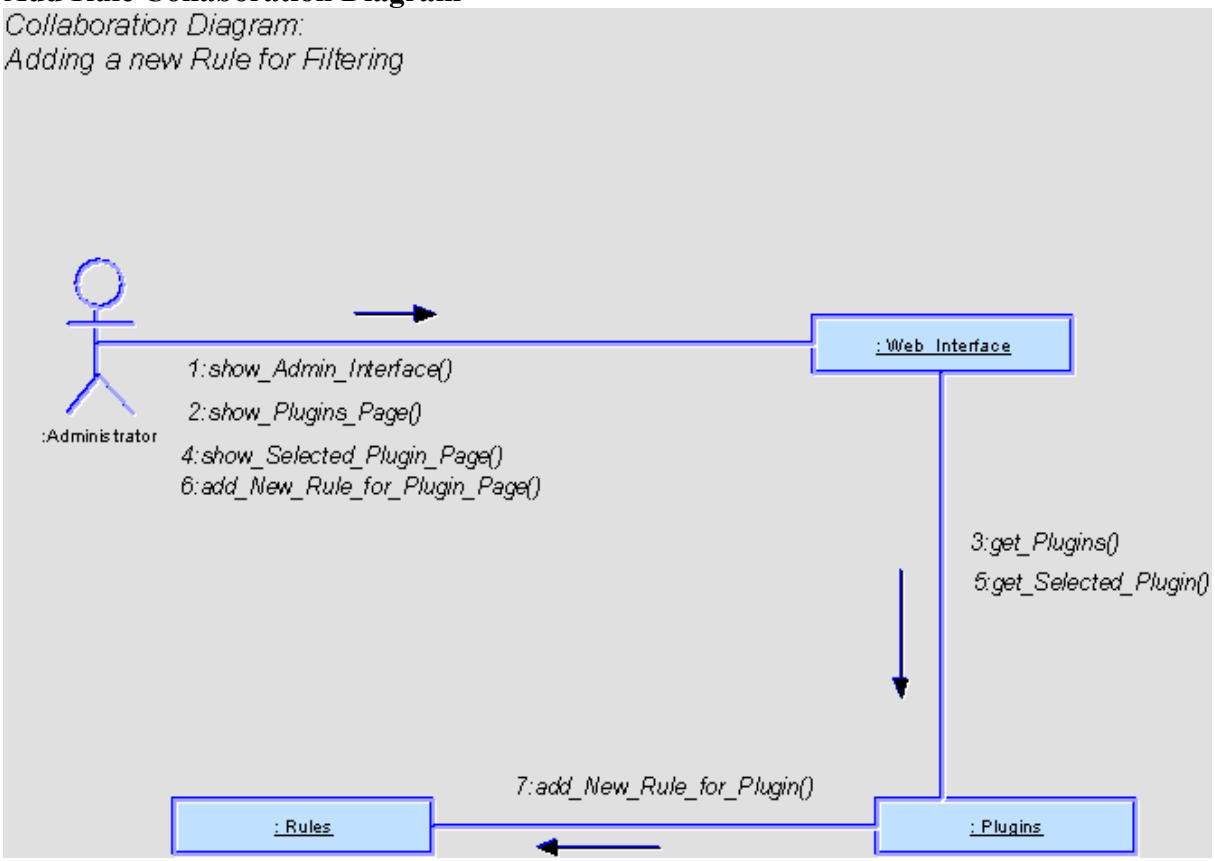
Add New Group Collaboration Diagram



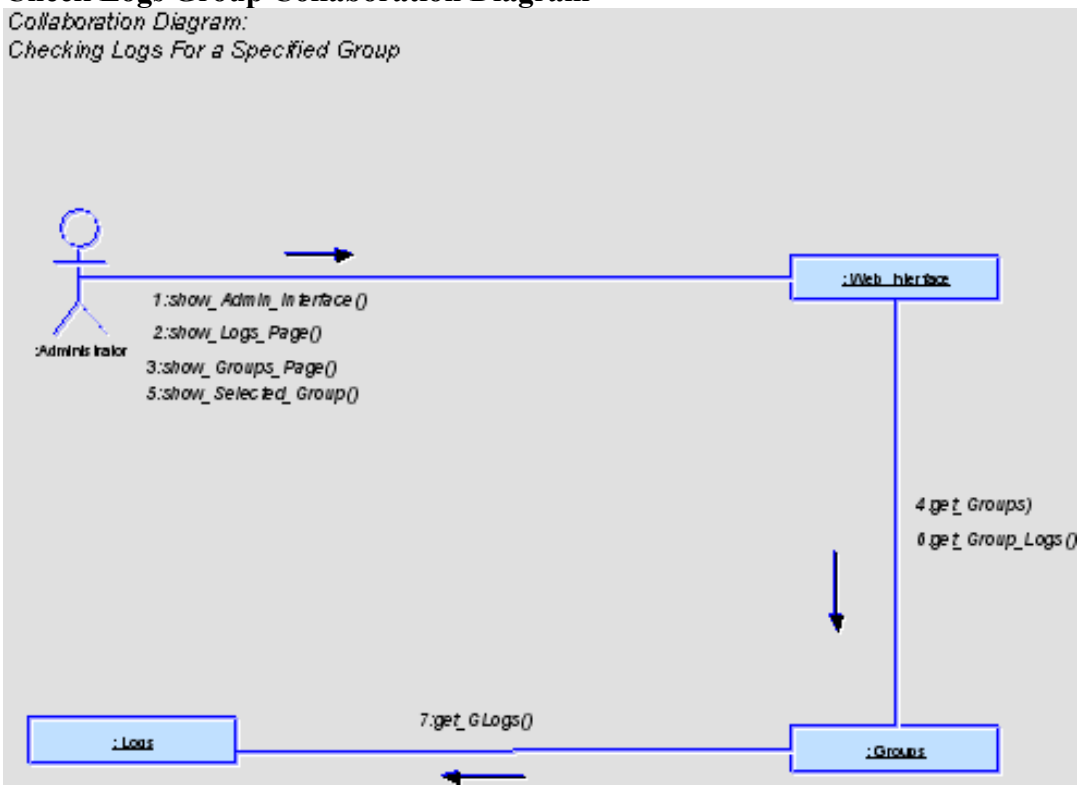
Add New User Collaboration Diagram



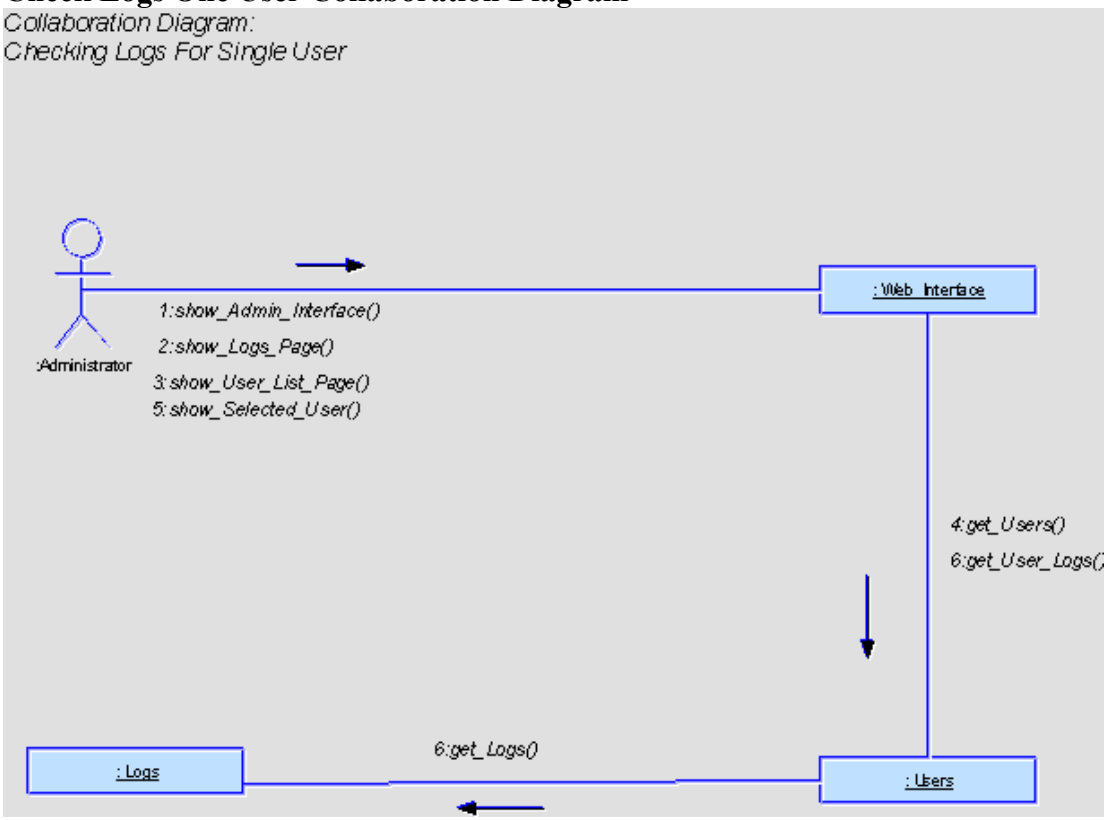
Add Rule Collaboration Diagram



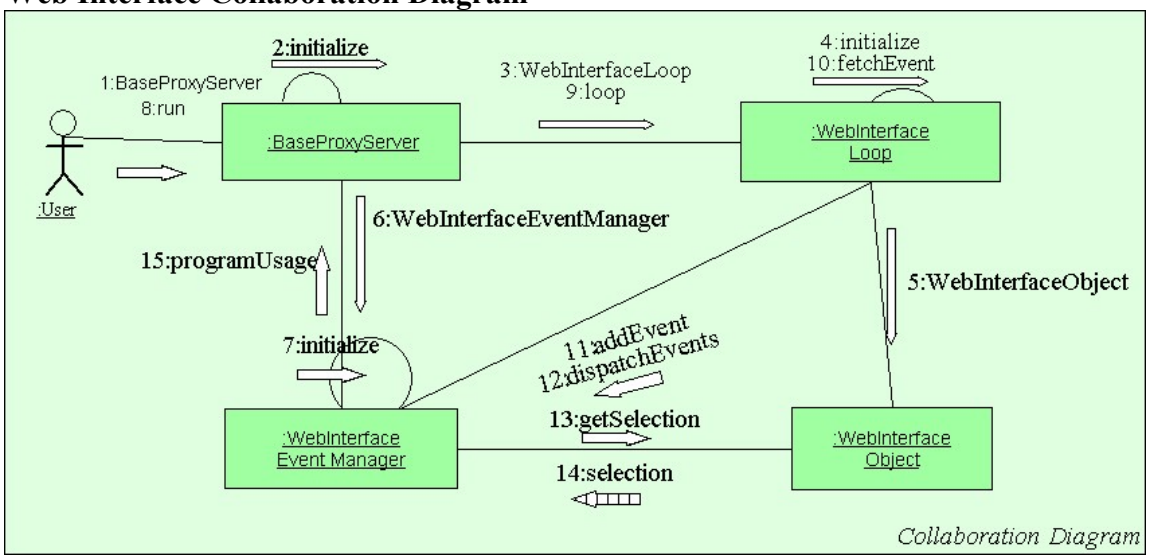
Check Logs Group Collaboration Diagram



Check Logs One User Collaboration Diagram

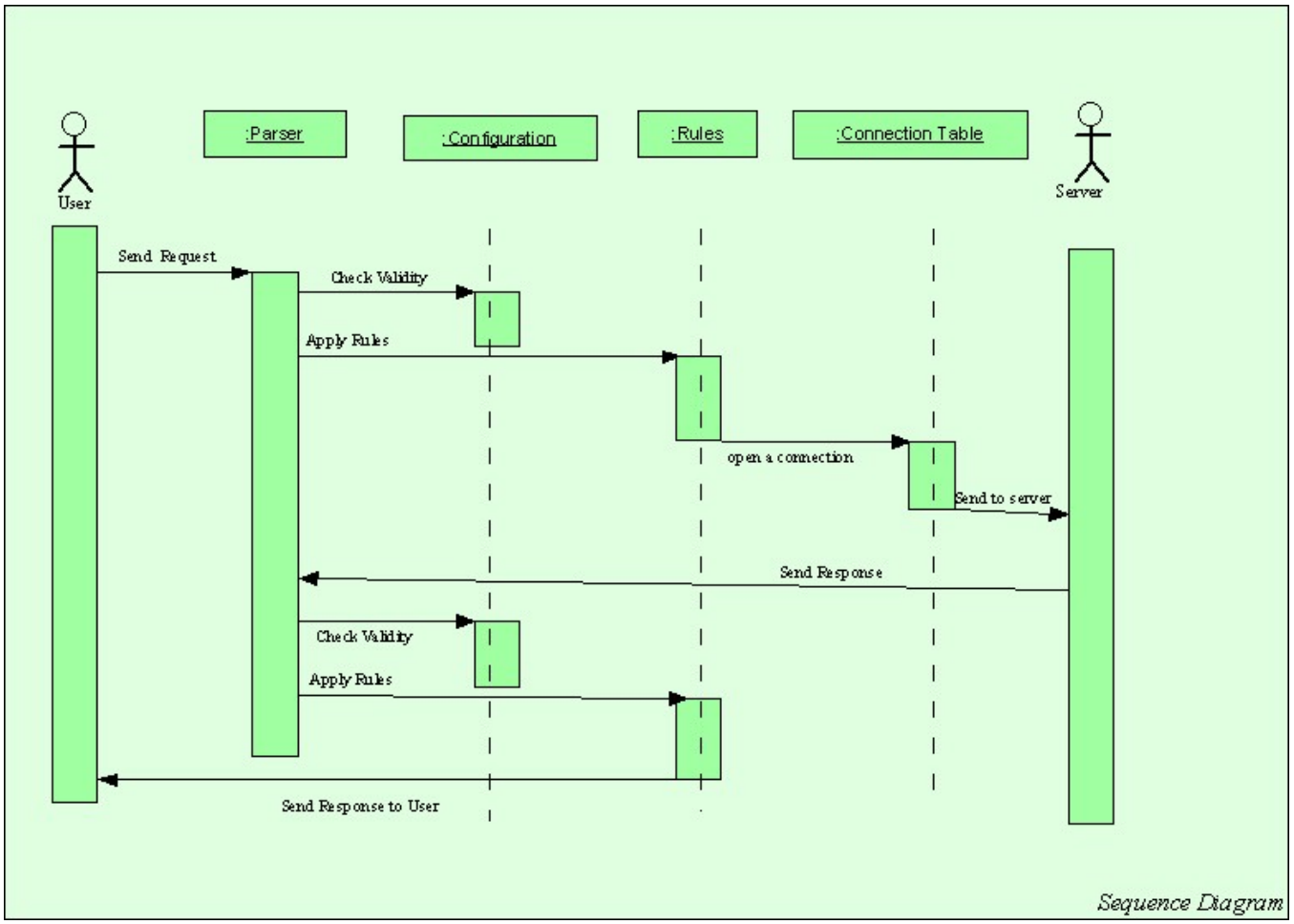


Web Interface Collaboration Diagram

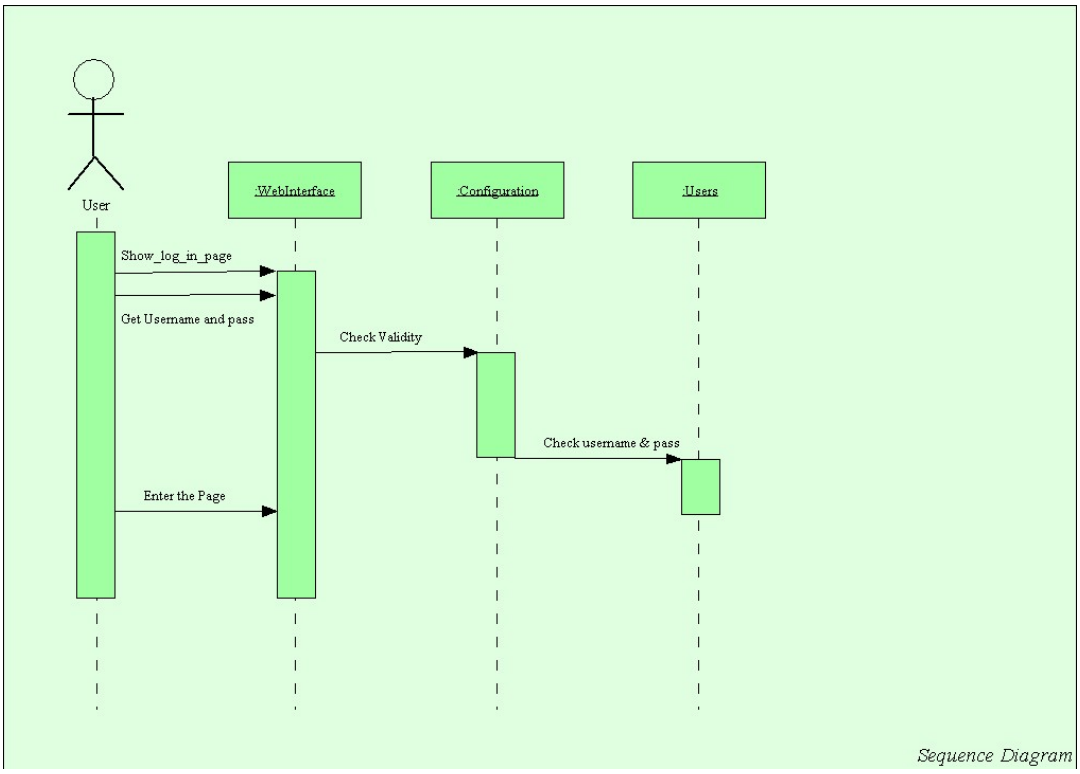


3.8 Sequence Diagrams

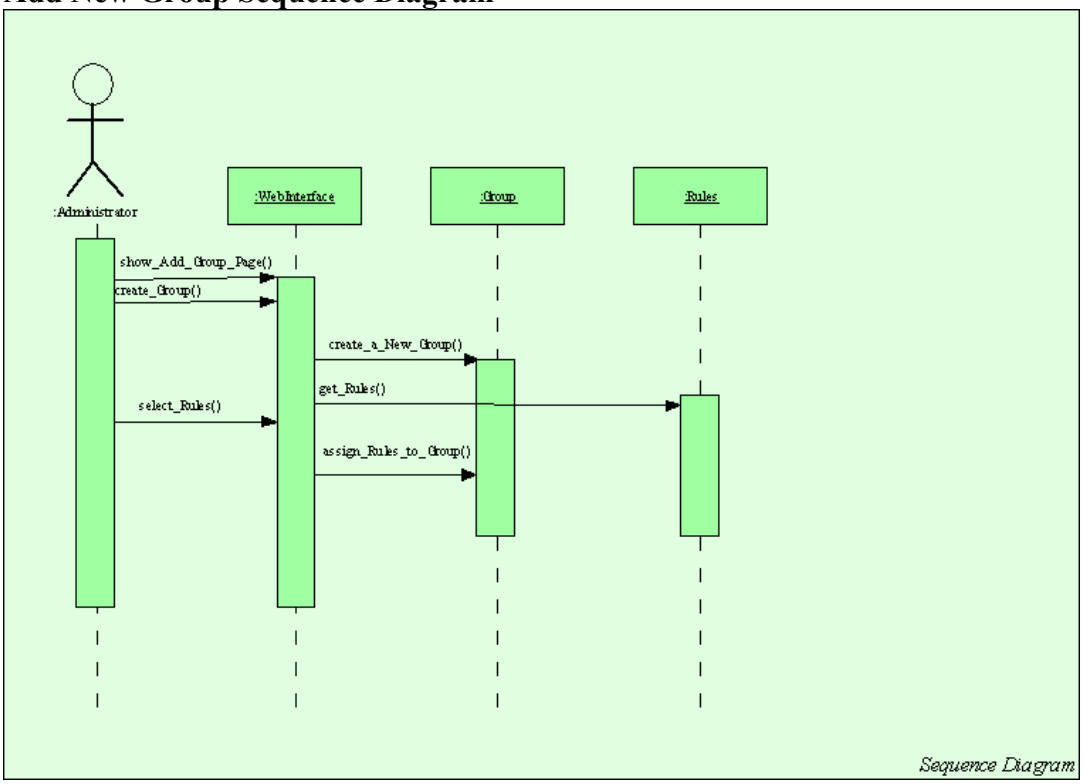
Request / Response Sequence Diagram



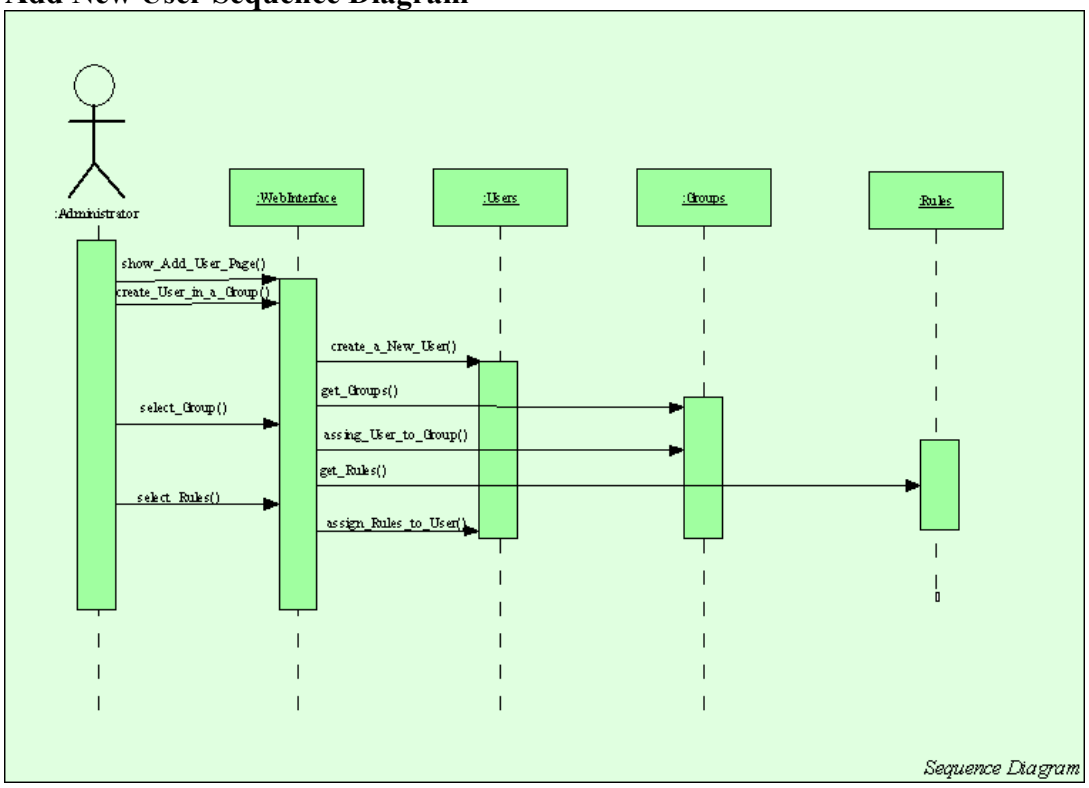
Login Sequence Diagram



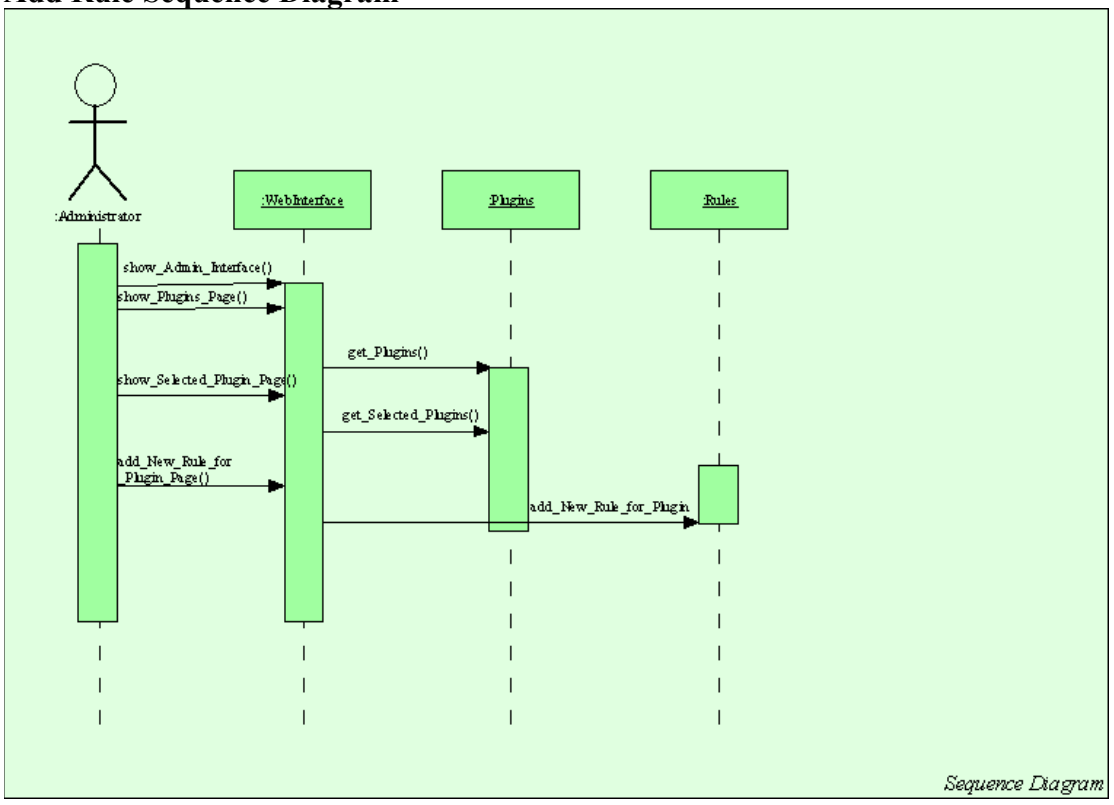
Add New Group Sequence Diagram



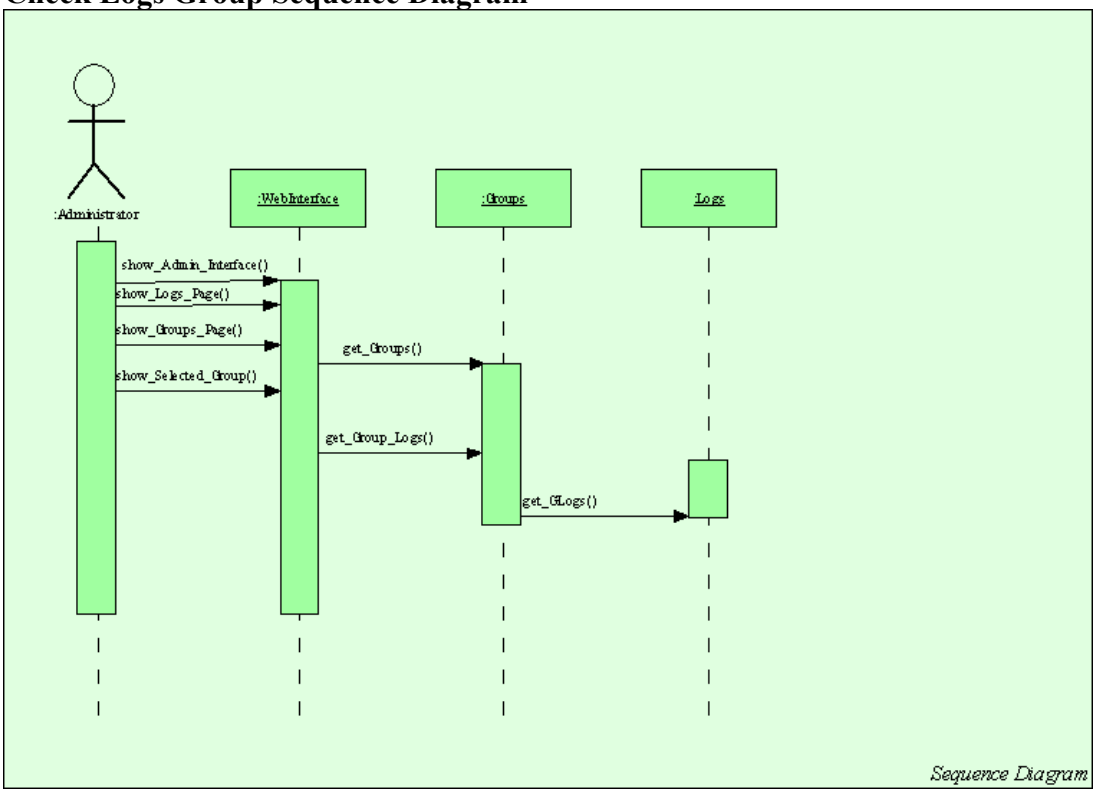
Add New User Sequence Diagram



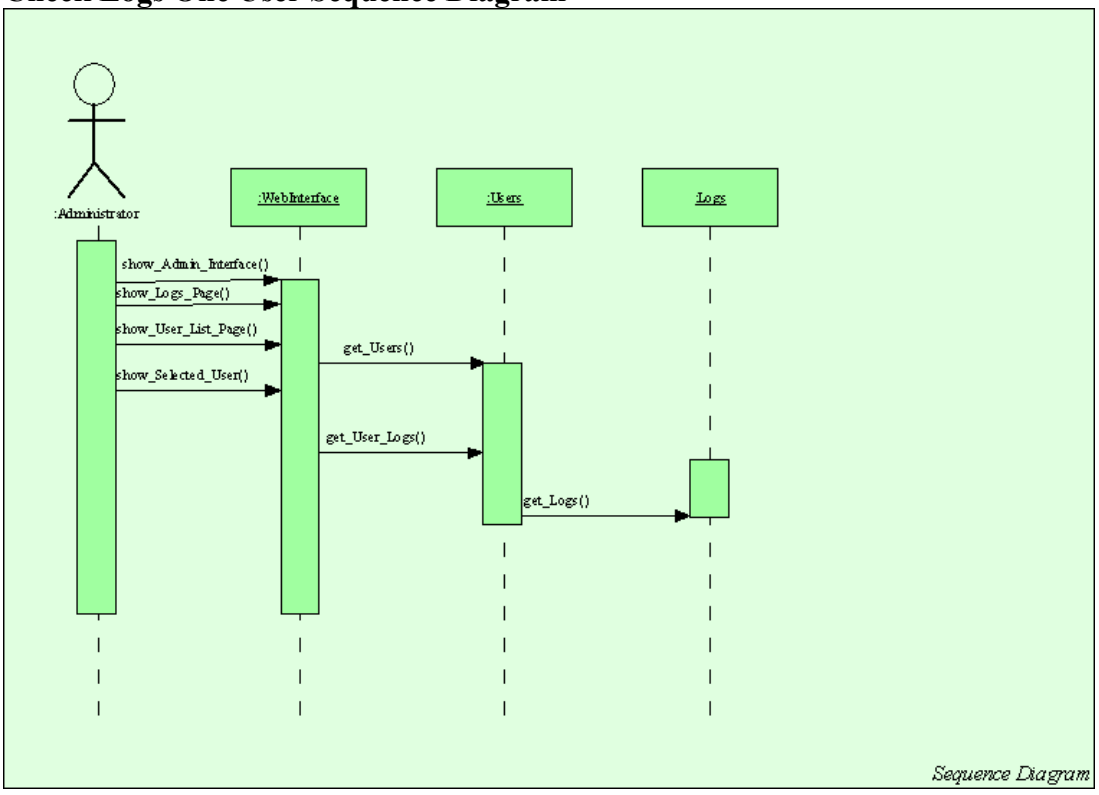
Add Rule Sequence Diagram



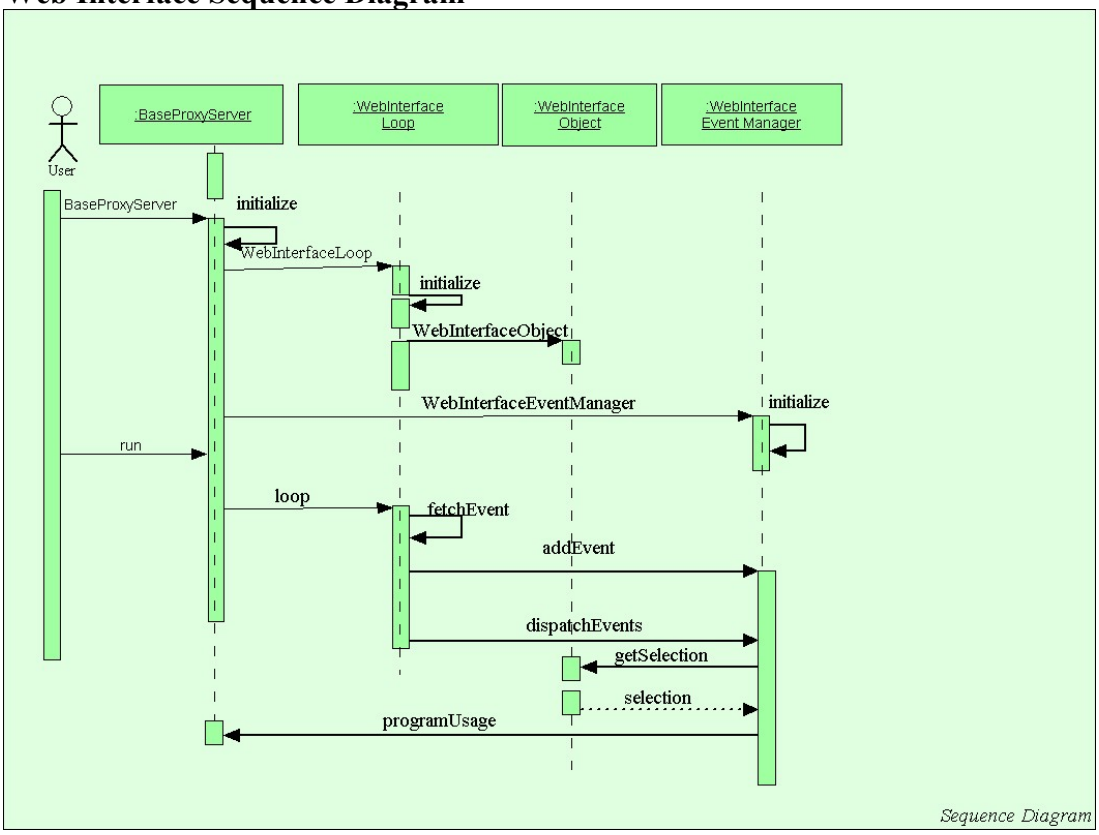
Check Logs Group Sequence Diagram



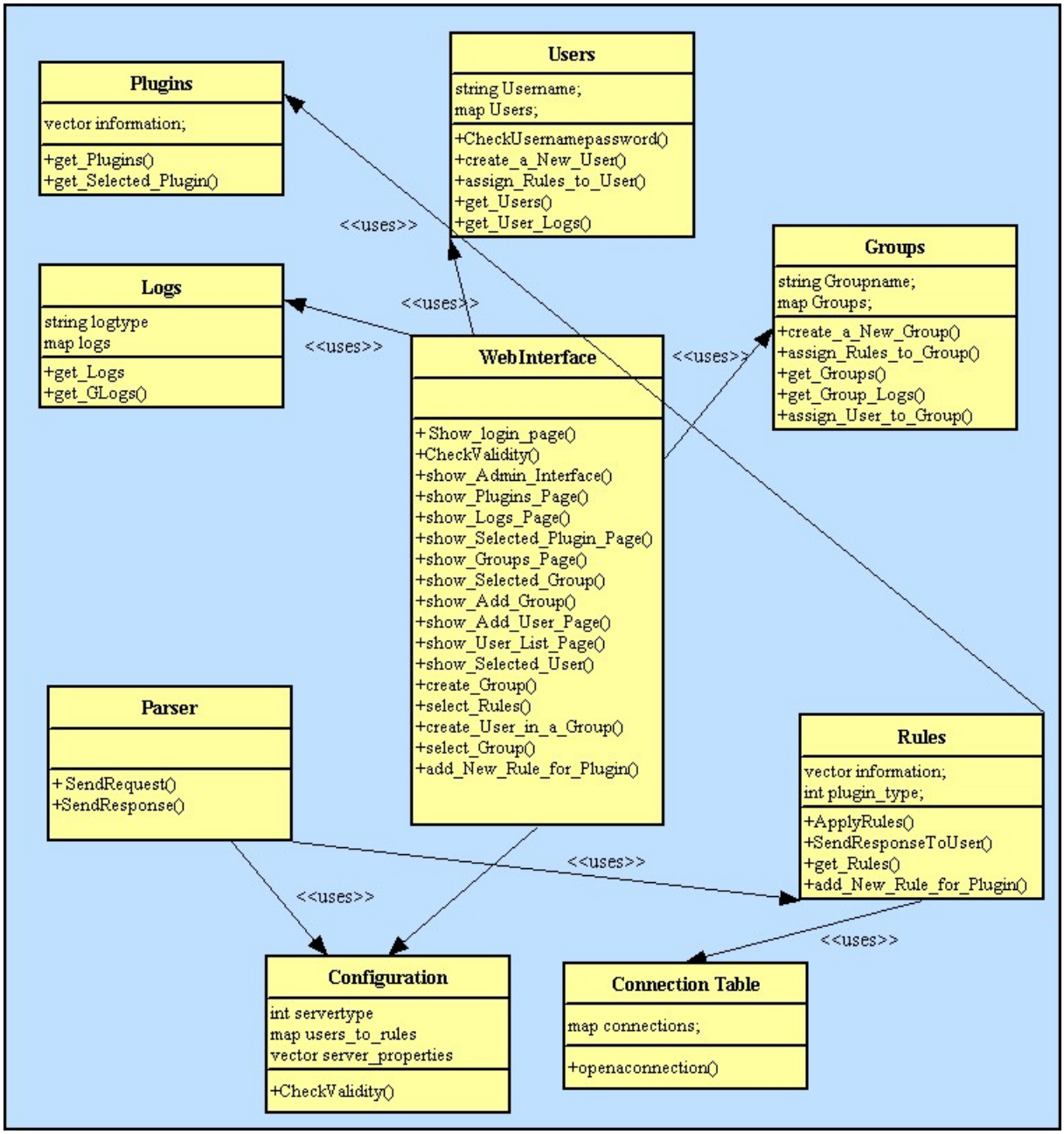
Check Logs One User Sequence Diagram



Web Interface Sequence Diagram



3.9 Class Diagram



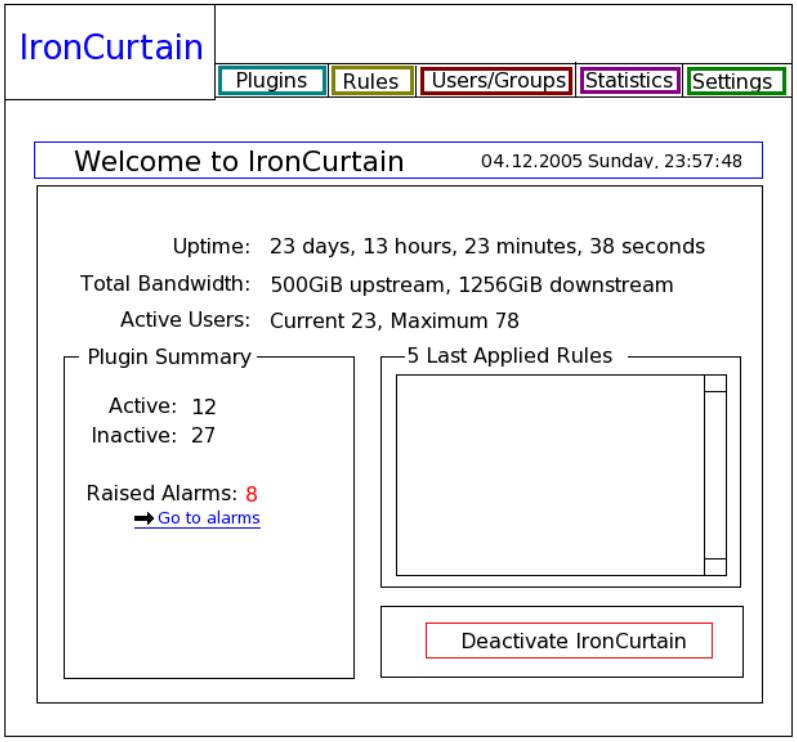
4 User Interface Design

IronCurtain's user interface will only be visible to the administrator. The interface will allow the administrator to change many aspects of IronCurtain, install new plugins and create new rules, view statistics and logs. The clients of IronCurtain will only use the user authentication interface.

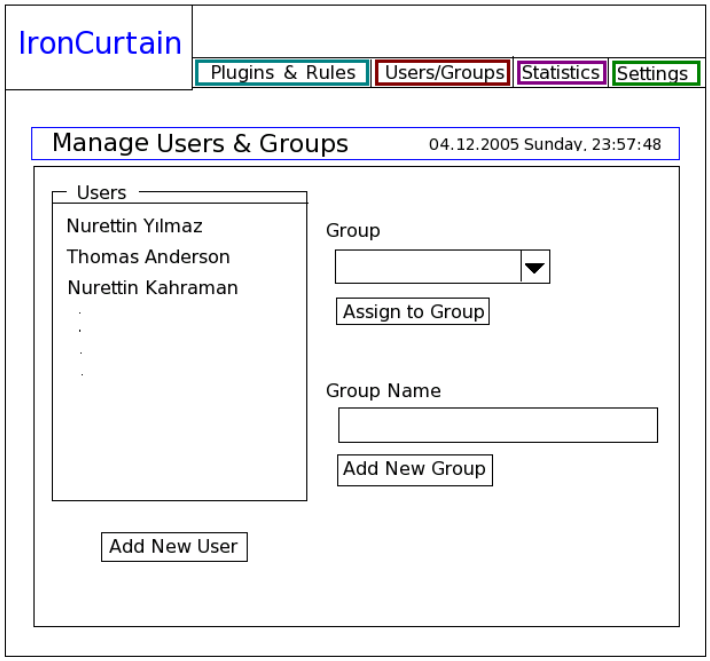
4.1 Screenshots

Our interface mockups are as follows.

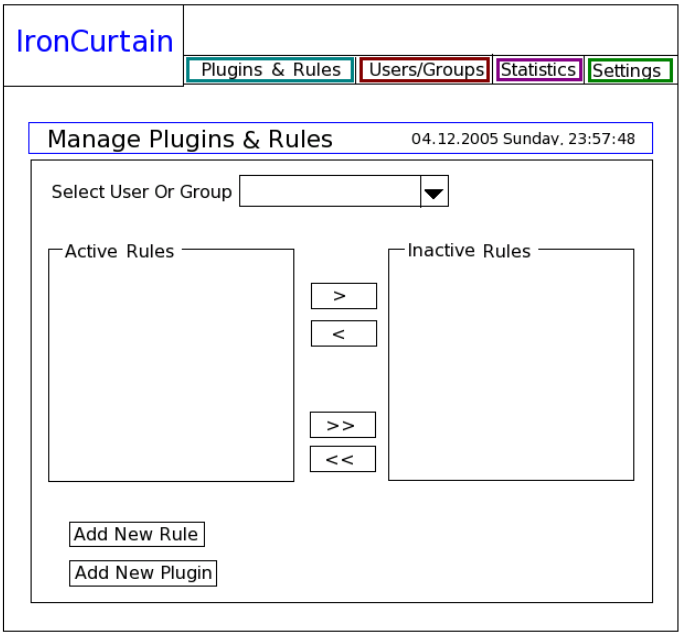
4.1.1 Summary View



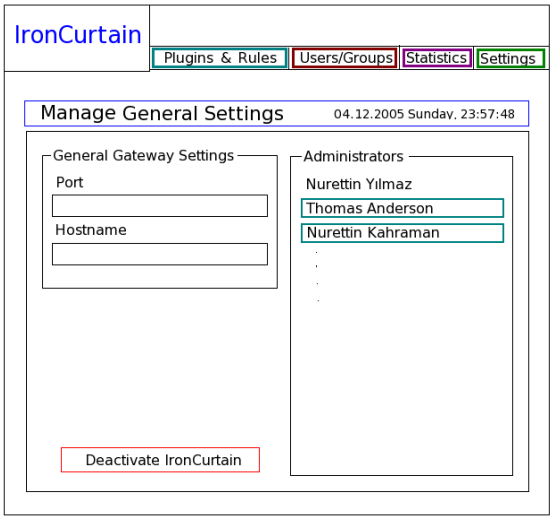
4.1.2 User and Group



4.1.3 Plugin and Rule Management



4.1.4 Settings



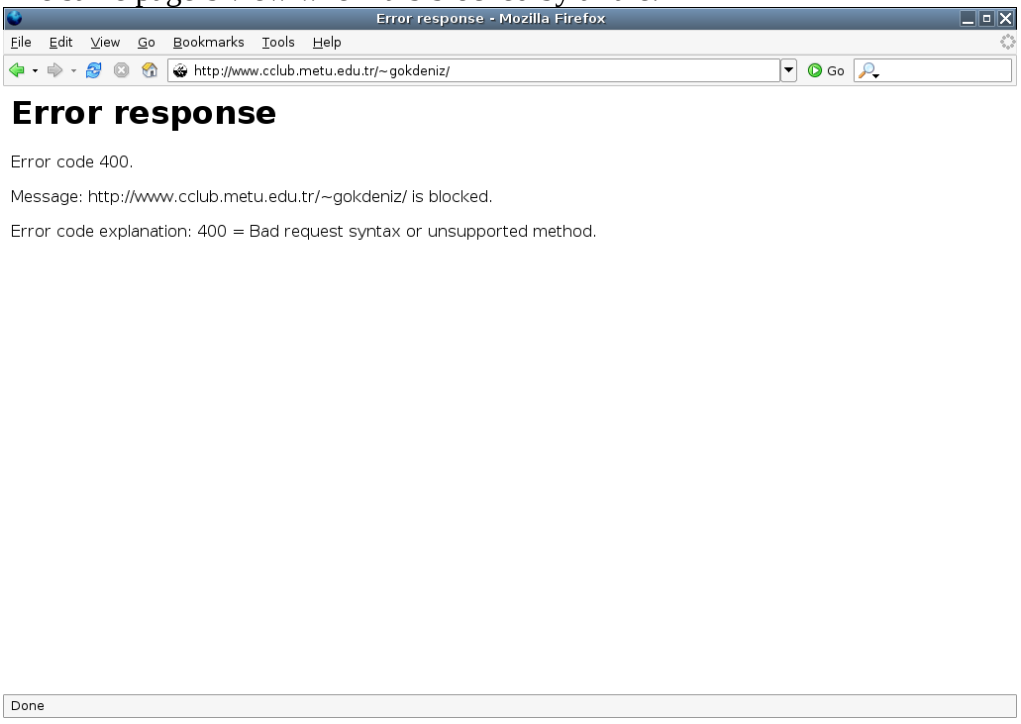
4.2 In Action

When IronCurtain is working, the results will be as follows:

The original page:

The same page with “Club” word replaced with stars, colored for emphasizing:

The same page’s view when it is blocked by a rule:



5 Requirements

This section gives details about and minimal software / hardware requirements.

5.1 Minimal Hardware Requirements

Minimal hardware requirements for our project are:

Development

- Intel Pentium IV 2 GHz or AMD Athlon 2000+
- 512 MB DDR RAM
- 40GB Hard Disk Space
- Internet Connection

End User

- Intel Pentium IV 2 GHz
- 1GB DDR RAM
- 100MB Hard Disk Space
- Local or Wide Area Network

5.2 Minimal Software Requirements

Software requirements for the project are divided into categories:

Development

- Recent Linux Distribution such as Ubuntu 5.10, or Windows XP or newer
- Full Installation of Python 2.4
- SPE Python IDE or Eclipse Installation with PyDev Plugin

End user

- Recent Linux Distribution such as Ubuntu 5.10, or Windows XP or newer
- Full Installation of Python 2.4
- SQLite Client
- Web Browser (Mozilla Firefox, Microsoft Internet Explorer)

6 Project Schedule

For our project IronCurtain, scheduling activities and tasks are given below.

6.1 Project Task Set

Framework Activities

- Customer Communication √
- Initial Design √
- Design √
- Programming
- Testing

Task Set

- Requirements specification √
- Learning the languages and tools √
- Prototype construction √
- Database construction
- Plugin & Rule construction
- Logging construction
- Interface construction
- Testing

List of deliverables

Documentation

- Project Proposal √
- Analysis Report √
- Initial Design Report √
- Detailed Design Report √

Prototype

Actual Implementation

Functional Decomposition of these tasks

Requirements Specification

Interviews with some software companies √

Internet Search √

Learning the languages and tools

- Determining the proper language √
- Determining the tutorials and manuals √
- Studying the tutorials √

Prototype

- Simple decomposition of communication
- Initial plugin engine
- Sample plugins and rules
- Initial web interface

Parts of Actual Implementation

Plugin & Rule Construction

- Plugin & Rule design √
- Plugin Engine
- Default Plugins
  - Tagging
  - Modify Header
  - Modify Content
  - Bandwidth Limiter
  - Image
  - Blocker
  - JavaScript Filtering
  - Keyword Filtering

Logging

- Logging and reporting engine
- Logging interface to plugins
- Statistics generation

Interface Construction

- Administrator web interface
- User authentication interface

Testing

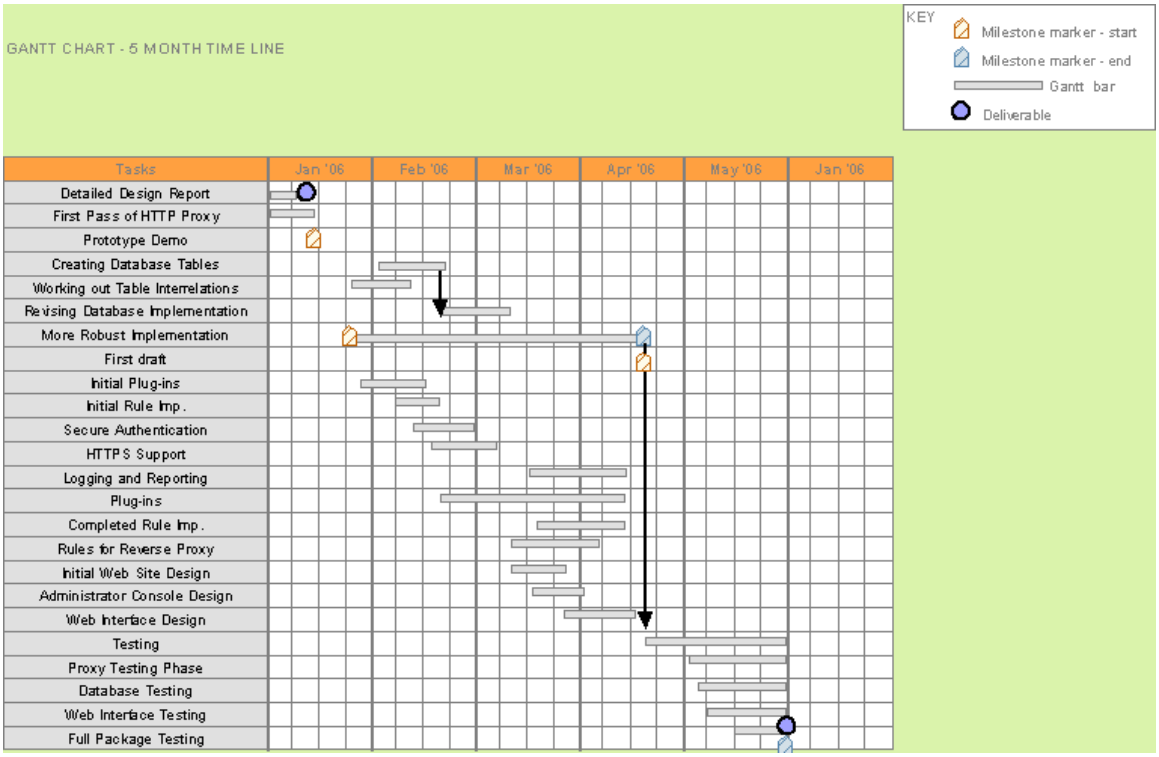
Web Interface testing

Proxy testing

Database testing

Full package testing

6.2 Gantt Chart



7. Testing

7.1 Unit Testing

- **Plugin engine and default plugins:** Processing rules synchronously in threads may be problematic, we are going to point at this while testing. We will test the plugins one by one, to ensure proper and stable operation of each.
- **Database:** We will use database mostly for accessing user information and reaching statistical information. Database structure is very important for our project since fast responding system is necessary for users to access internet. Testing for performance issues will be mostly related to the database.
- **Web interface:** Security issues will be important since web interface is will be a common use environment, securing user information will be important.
- **Log module:** Statistical data will be gathered by this module. Database testing will be an important issue because we will record information on the database and also, get statistical information from the database.

7.2 Integration Testing

We are going to test the system as whole after combining different plugins with the plugin engine in IronCurtain. We will ensure synchronized processing of plugins is correct, and stability is not hampered by the plugin engine's actions.

7.3 Alpha Testing

After producing code that works with a satisfactory performance and reliability, and after finishing unit tests and integration tests, we will conduct our alpha test. In the alpha test, we will install IronCurtain to our computers, then we will define some rules that we need in our daily web activities. Then we will measure how effective is IronCurtain's actions.