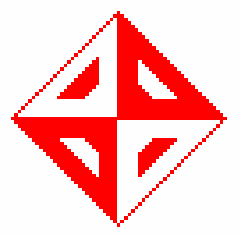
Detailed Design Report

January 8, 2005

SmartGuardian

SmartGuardian™ Software



Computer Engineering Department
Middle East Technical University

by
SmartSoft Network Solutions, Inc.
Tayfun Şen - Özden Meren - Sedat Zelyüt - Hakan Özadam - Akın Öztürk

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to state the design constraints of the computer software called SmartGuardian. It is supposed to be the major guideline for software developers working on this project.

We have built this design document on to the initial design document we had prepared before. Note that even though we have used our initial design as a basis, we have revised it thoroughly in light of the advices and criticism received from our guides.

This document contains detailed design description of the software system. Modules of the system and their functional, structural and behavioral aspects are explicitly stated. Corresponding DFD, ER and UML diagrams are also included in the document.

1.2 Problem Definition

There are various issues related with the Internet access of organizations, which can result in big revenue losses. These issues include employees consuming Internet resources contradicting with the company policies and various threats stemming from the Internet itself. That is why employers today want to control the Internet access of their organizations.

Network administrators are incapable of handling problems arising from local area network users since the problems are of high complexity and they should be handled

the instant of occurrence.

For these reasons there is an increasing demand for gateway applications which takes care of such issues.

1.3 Goals and Objectives

The Primary goal of SmartGuardian is to control and monitor the internet access of an organization. It is supposed to enforce the rules and policies defined by the network administrator of the organization. It checks the incoming and outgoing traffic for validity and bandwidth usage limits. It displays detailed reports about the internet usage of the organization via web pages.

SmartGuardian is designed to optimize the internet access of the organization by implementing bandwidth rules and policies. It is also supposed to improve the efficiency of the employees by implementing content filtering and by restricting the internet access according to the rules defined by the administrator.

1.4 System Overview

SmartGuardian is an application level gateway. It is going to be developed on Fedora Core 4 Linux operating system. It will be implemented using C++ and PHP. It has a web interface which runs on Apache web server. MySQL is the database of the system, though the system will use several external text files to store some system settings and most of the activity logs.

SmartGuardian is composed of 3 major components: SmartGuardian Wall, SmartGuardian Crawler and SmartGuardian Controller. SmartGuardian Wall and

SmartGuardian Crawler will be implemented in C++ and SmartGuardian Controller will be implemented using PHP.

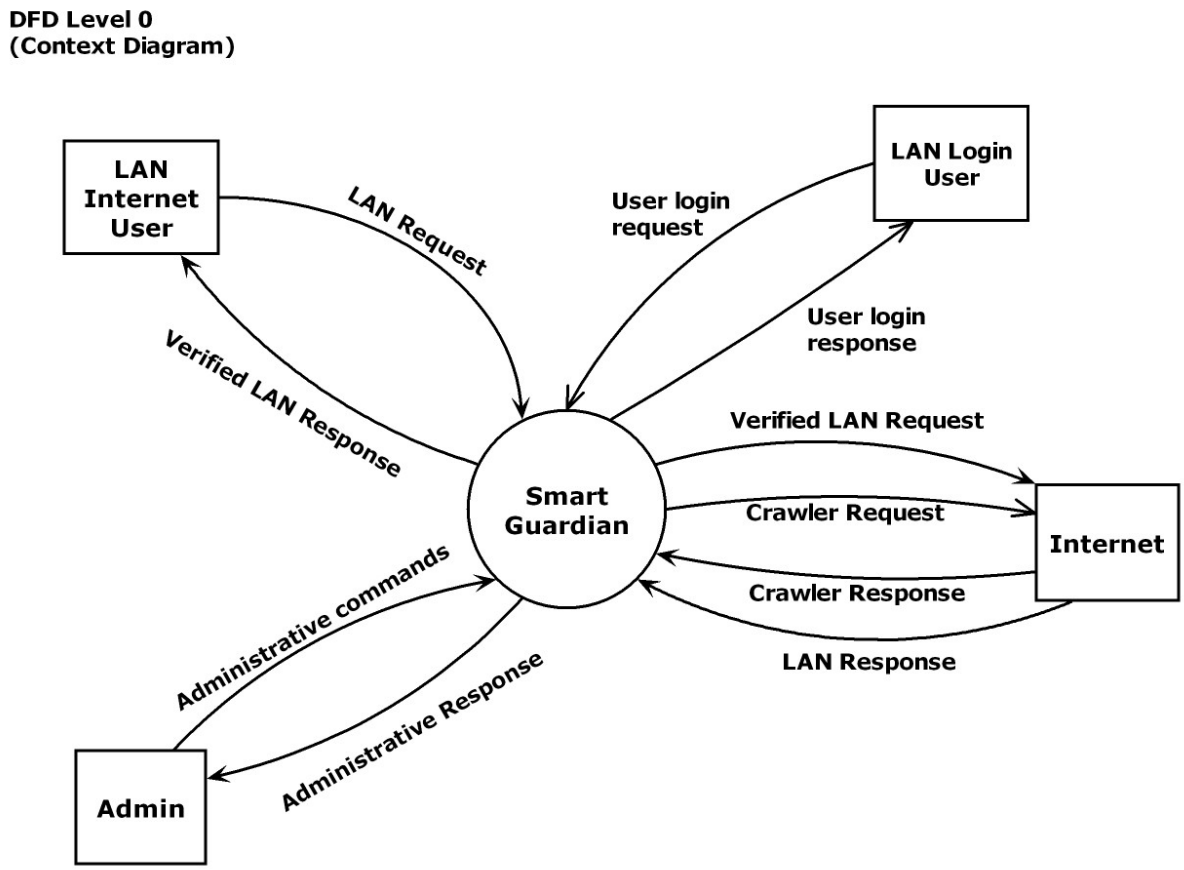


Figure 1.1: DFD Level 0 of the system

1.5 Definitions and Acronyms

DFD: Data Flow Diagram

ER: Entity Relationship

HTTP: Hyper Text Transfer Protocol

IP: Internet Protocol

SG: SmartGuardian software system

UML: Unified Modeling Language

URL: Uniform Resource Locator

1.6 References:

http://paulgraham.com/spam.html :

Paul Graham talks about his experiences in implementing a spam filter using the Bayesian Algorithm.

http://www.digitallumber.net/software/willow/ :

This is the homepage of the Willow software which provides content filtering.

www.mathworld.com :

For statistics and mathematics of the Bayesian Algorithm.

http://en.wikipedia.org/wiki/Data_mining :

Used for general data mining information and directions.

2 DESIGN CONSIDERATIONS

2.1 Module Decomposition

The SmartGuardian software system is composed of four main modules: Authenticator, SmartGuardian Wall, SmartGuardian Crawler and SmartGuardian Controller.

System structure is presented in a tree diagram in the next page.

2.1.1 System Structure

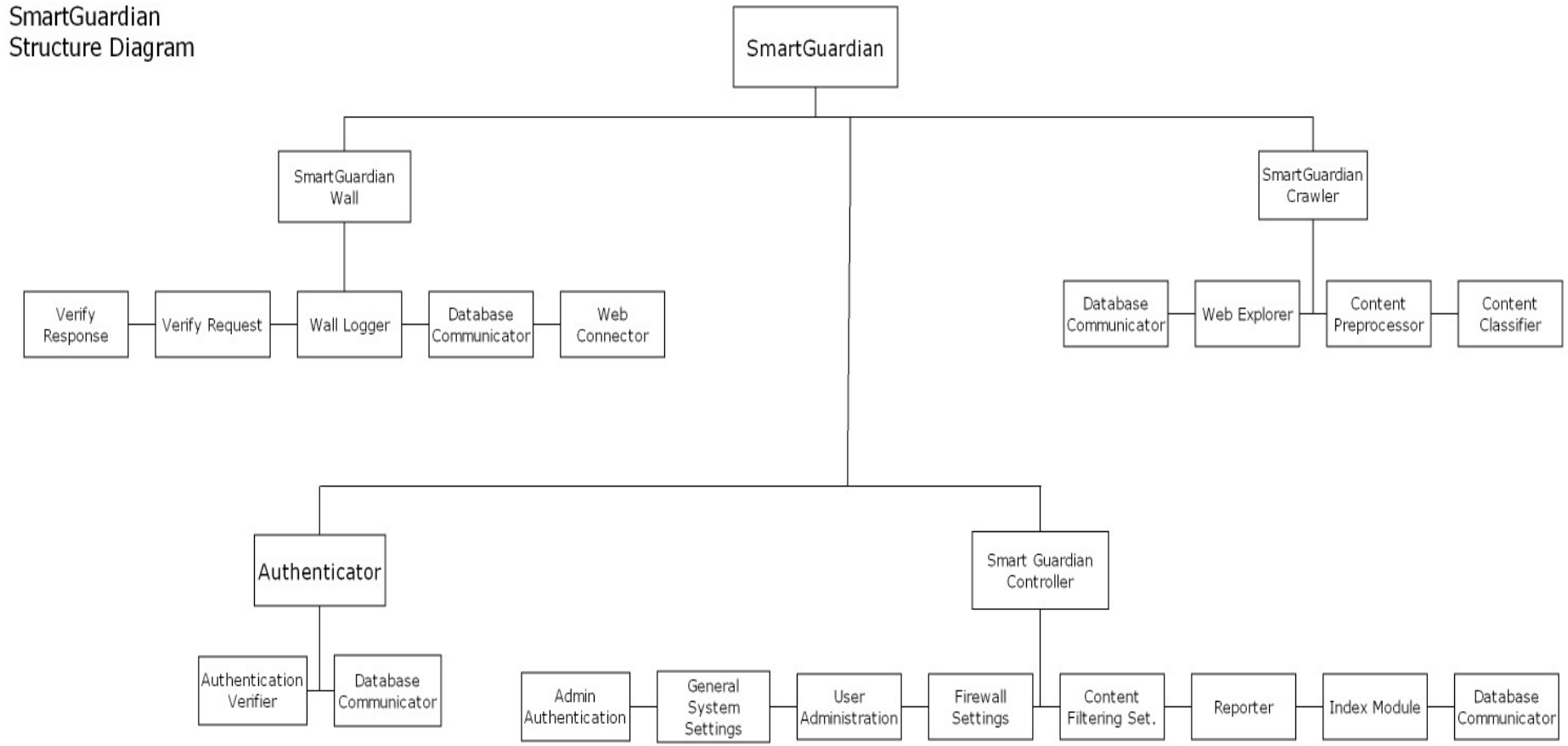


Figure 2.1: Structure Diagram

Level 1 Data flow diagram showing the main modules and their dependencies in the system is shown in the following figure.

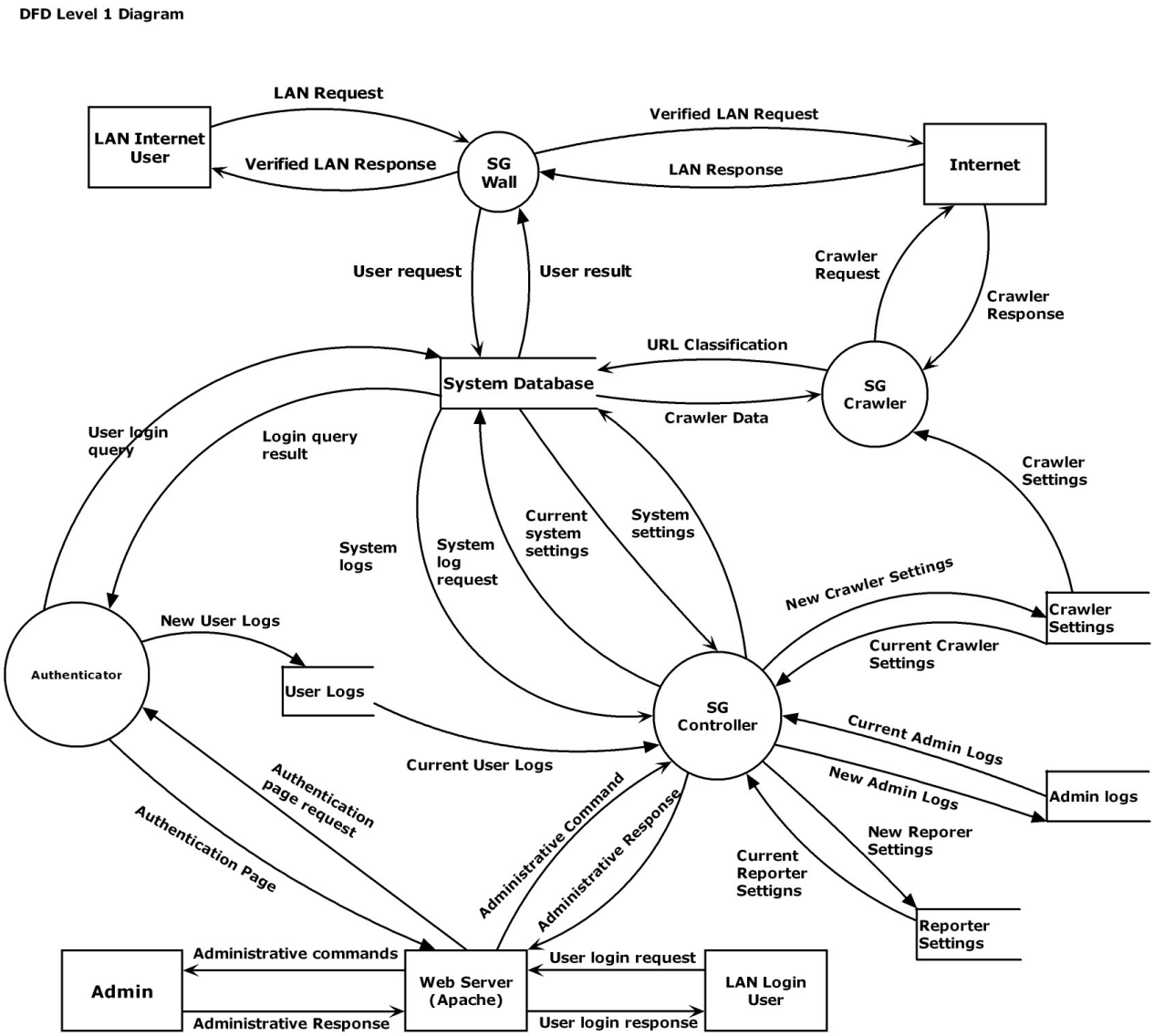


Figure 2.2: DFD Level 1

2.1.2 SmartGuardian Wall

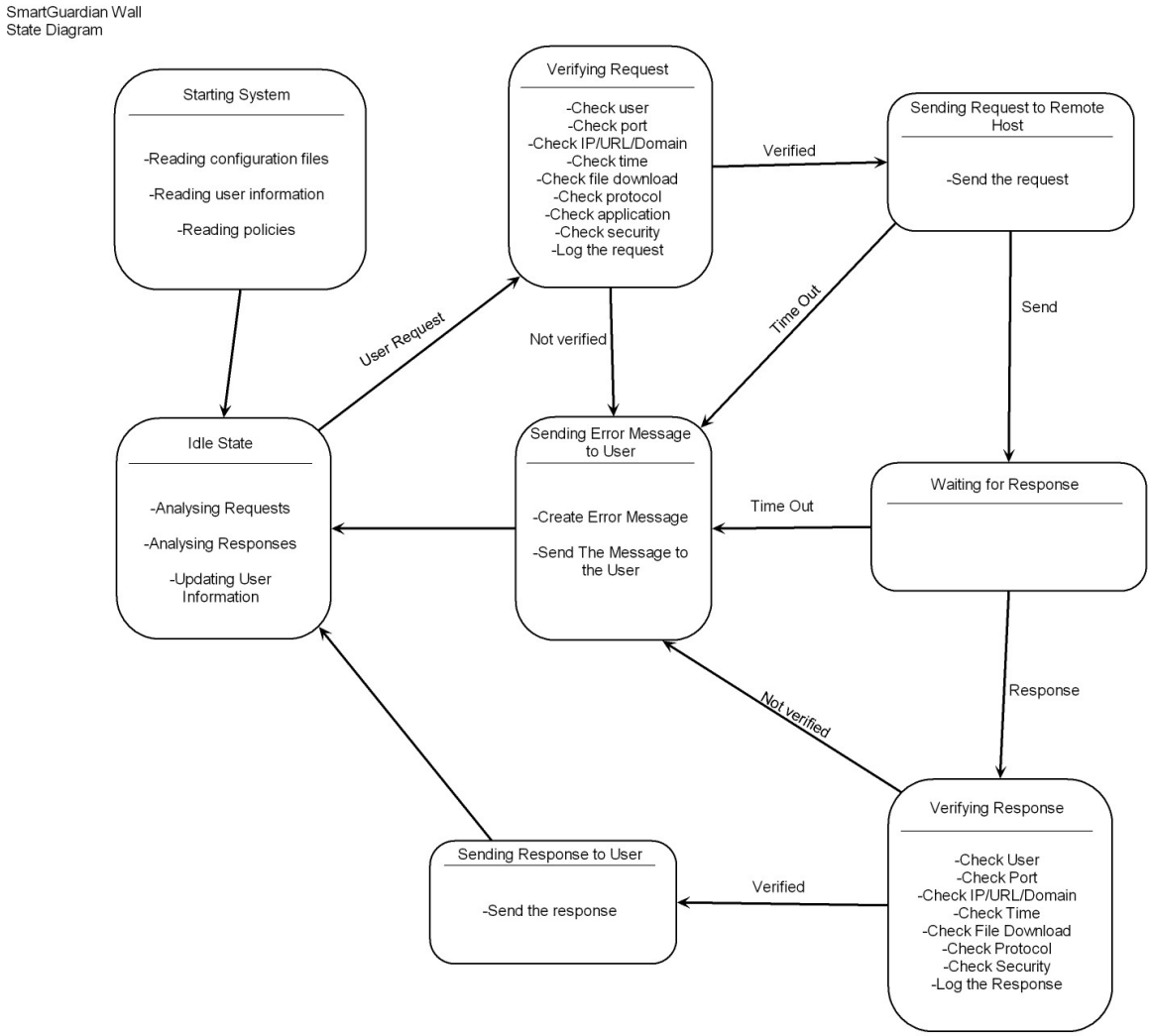


Figure 2.3: State Diagram of SmartGuardian Wall

The state diagram of SmartGuardian Wall is shown in the figure above. There are eight states that the system can be in. When there are no requests or responses the system is in idle state. In this state the system updates the user information such as the login status, bandwidth usage and the download limits of the users. Besides that, system checks and analyzes requests and responses.

When a request comes the system goes to the verifying request state. In this state the system checks the status (logged in/ not logged in) of the user, and according to the policy it checks the port to be used and protocol to be used, the domain to be accessed, the time of the request, the file to be downloaded (if this is a file download request), the application that made the request, current bandwidth of the user and finally the security. Depending on the policy and the request, it either verifies the request and serves it or rejects it. If the request is verified the system goes to sending request state. In this state it sends the request to the remote host and then logs the request. Then it goes to the waiting for response state for waiting the response of the request that the system sent to remote host. If the request is not verified it goes to sending error state. In that state it firstly creates an error report stating the reason of the error and then sends the report to the user. Then it logs the request and the error and then goes to the idle state.

After the expected response comes in the waiting for response state, the system goes to Verifying Response state. In this state the system checks whether there is a logged in user at the destination IP, and according to the policy it checks the port and protocol of the response, the domain from which the response comes, the time of response, the file to be downloaded (if this is a file download request), current bandwidth of the user and finally the security. Depending on the policy and the response, it either verifies the response and sends it to the user or rejects it. If the response is verified the system goes to sending response to user state. In this state it sends the response to the user and then logs the response. Then it goes to the idle state. If the response is not verified it goes to sending error state. In that state it firstly creates an error report stating the reason of the error and then sends the report to the user. Then it logs the response and the error and then goes to the idle state.

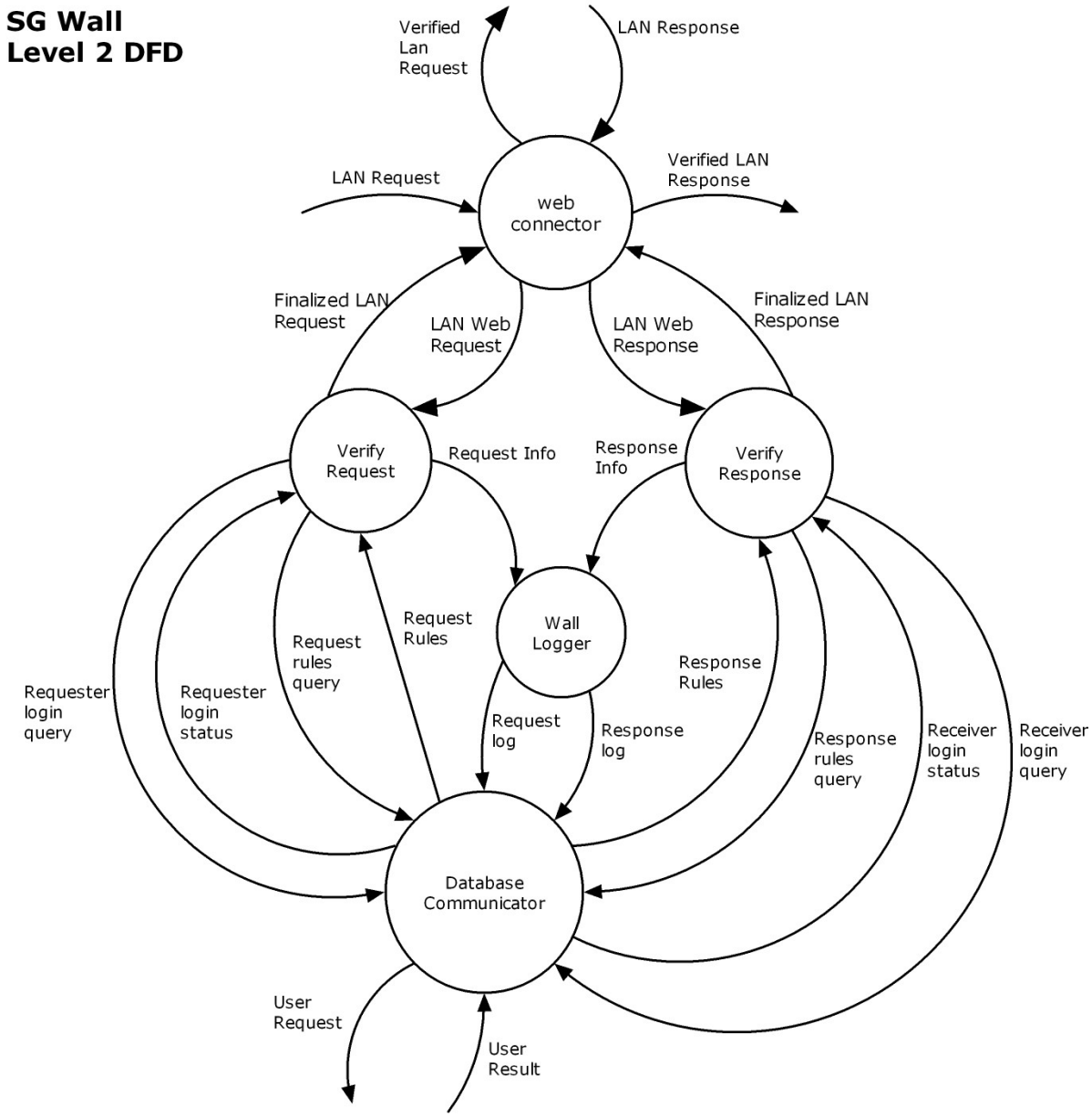


Figure 2.4: Level 2 DFD of SmartGuardian Wall

The main function of SmartGuardian Wall is to control incoming and outgoing traffic. It checks LAN requests (outgoing traffic) & LAN responses (incoming traffic) for validity.

SmartGuardian Wall controls the incoming and outgoing traffic according to the predefined rules and policies. SmartGuardian Wall works in the application layer; it inspects the HTTP requests flowing through the network for validity. It has the ability to block selected ports, IPs, URLs, file names and user agents and also it is capable of detecting threats caused by incoming or outgoing traffic and is capable of withstanding these threats.

This module is composed of five sub modules: Verify Request, Verify Response, Wall Logger, Web Connector and Database Communicator.

LAN requests (outgoing traffic) are handled by the sub module Verify Request, LAN responses are handled by the sub module Verify Response, these two sub modules report the result of their actions to Wall Logger which records them for monitoring purposes. All the data flow to the system database is via Database Communicator. Socket connections for both incoming and outgoing traffic are handled by the sub module Web Connector.

2.1.2.1 Verify Request

Verify Request takes the requests coming from a logged in user, examines the request, in its sub module Verify Request Rules and decides whether it is a valid request or not. Verification result (it can be passed or failed) is sent to Wall Logger module. Verify Request needs rules in order to make decisions about request. It gets those rules from the system database via Database Communicator

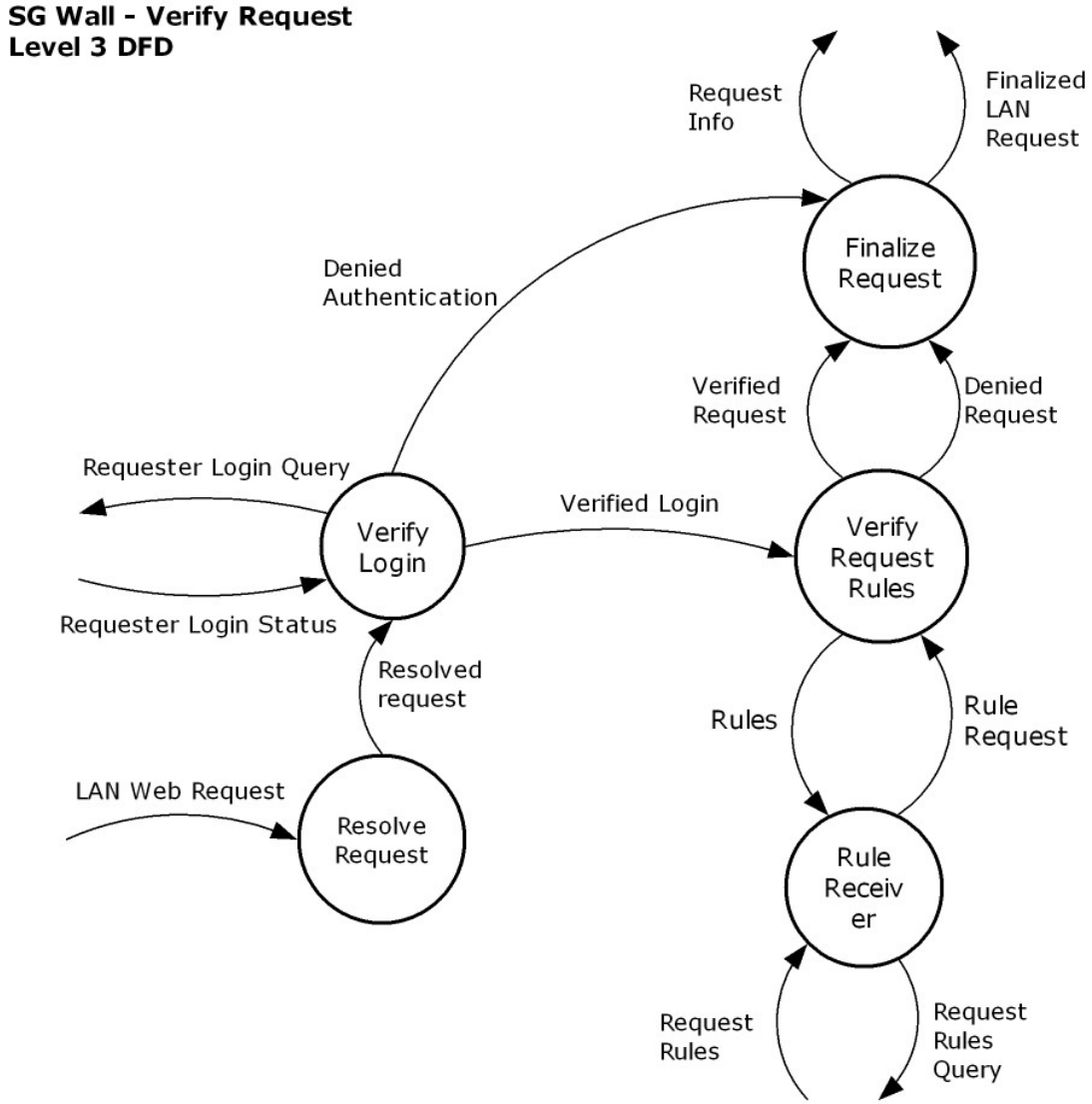


Figure 2.5: Level 3 DFD of Verify Request

2.1.2.1.1 Resolve Request

Incoming requests are resolved before any further processing. This process extracts the following information from the HTTP header which belongs to the request and prepares necessary data structures for the use of other processes. Resolve Request extracts all the information present in the HTTP header which is going to be used in rule verification.

2.1.2.1.2 Verify Login

This process checks the source IP of the LAN request and determines if this IP is among the logged in users (users are identified by their IPs). It gets the logged in users information from system database through Database Communicator module. If the IP does not belong to any logged in user, request is ignored. If the IP is idle longer than a max idle time which is determined by the system administrator, corresponding user is marked as logged out and request is ignored. If IP belongs to a logged in user (who has had network activities recently), then this process certifies that request comes from a logged in user and request is further analyzed by Verify Request Rules.

2.1.2.1.3 Verify Request Rules

Verify Request Rules, obtains the rules from the system through the process Rule Receiver. It checks the logged in request for validity according to those rules. It sends the result of verification to Finalize Request.

2.1.2.1.4 Finalize Request

Finalize Request is the last process of the verification chain. If there are no denials until this process then request is verified and sent to the internet and is recorded. If

Finalize Request receives any denial of rules then request is ignored and result is recorded.

2.1.2.2 Verify Response

Verify Response is an analog of Verify Request which has minor differences both in verification process and in decision mechanism.

Verify Response examines the incoming traffic for validity. If the response is valid it is sent to corresponding destination if not the response is ignored. In both cases the result is logged.

The data flow diagram of Verify Response is shown on the next page.

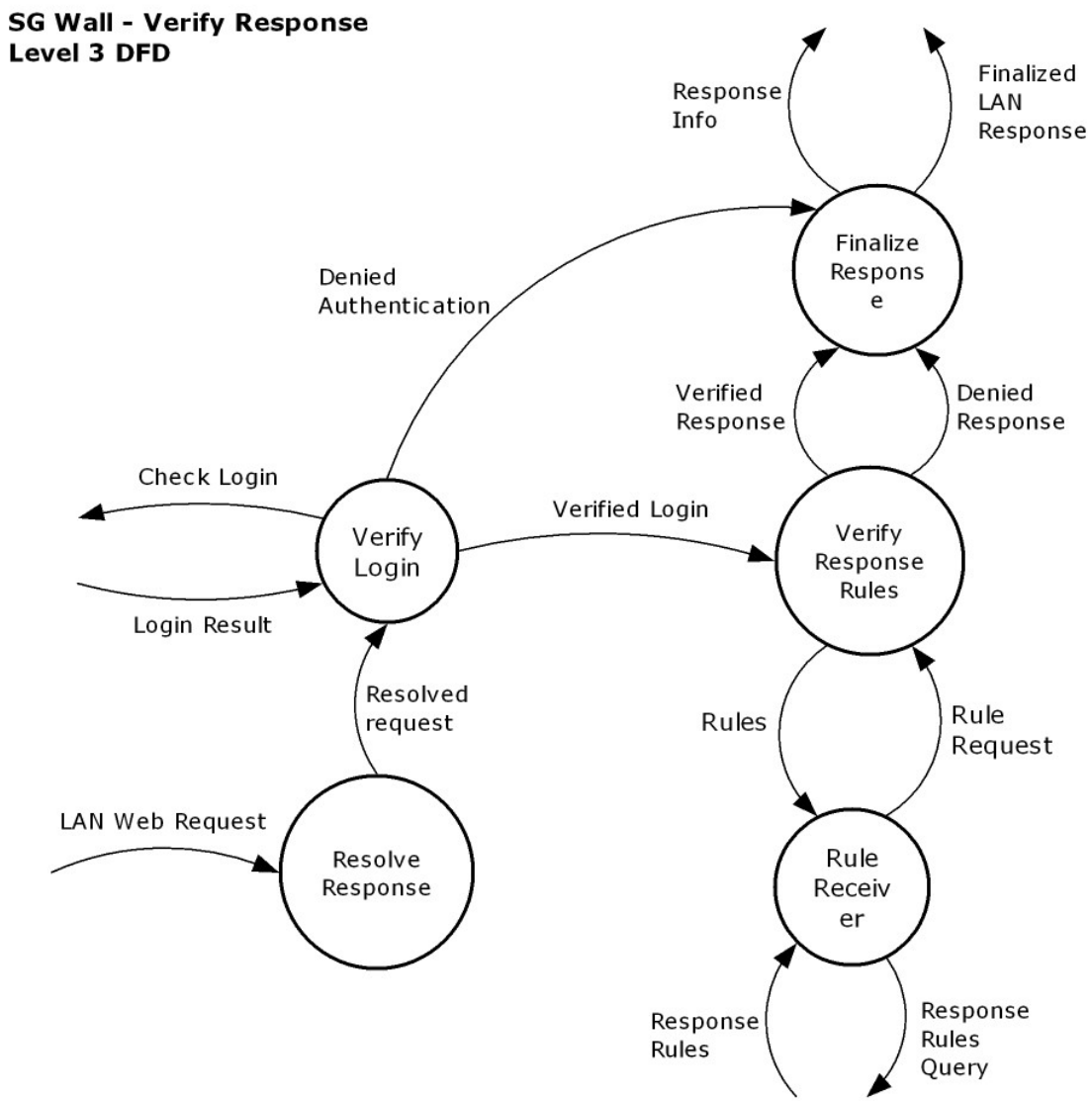


Figure 2.6: Level 3 DFD of Verify Response

Verify Response has the following sub modules:

2.1.2.2.1 Resolve Response

Responses coming from the Web Connector are first resolved; destination address, source address, port and all other available HTTP header information are extracted for the use of other processes of Verify Response.

2.1.2.2.2 Verify Login

Verify Login checks whether the destination IP is among the logged in IPs. If it is, the response is further analyzed by Response Rules, if not response verification terminates and situation is reported to Finalize Response.

2.1.2.2.3 Verify Response Rules

Requests whose login is verified are next examined by Verify Response Rules. Verify Response Rules gets polices and rules from Rule Receiver. It checks the response for validity according to those rules. The result of verification is sent to Finalize Request.

2.1.2.2.4 Finalize Response

Finalize Response gets the verification result, sends the response to LAN or blocks it according to the result and records it.

2.1.2.3 Wall Logger

Wall Logger keeps records of requests for both incoming and outgoing traffic. It records the verification results coming from Verify Response & Verify Request to the according tables in the system database through Database Communicator module.

2.1.2.4 Classes

The class diagram shown in figure 2.7 describes the basic classes of the SG Wall. Below are the descriptions of the most important ones.

2.1.2.4.1 Authenticator Class

This class keeps the login information of the users and the information about which IP belongs to which user. The ***checkUser*** method returns the ***userName*** of the user using the given IP address. If the user it not known then an empty string is returned. openSession method asks a user for username and password. If the user enters correct values a new session is started for that user.

2.1.2.4.2 WebConnector Class

This class is used for receiving and sending the requests and responses. Also the routing problem is handled by this class. It has two attributes namely ***ports*** and ***IPs***. These attributes are two vectors keeping the information of "which port is bound to which IP". We use this information to handle the routing problem. Every request is sent to the destination from a different port, so the response to that request will arrive from that port. By this way, we will be able to find the user to which we will send the response. The methods ***addBinding*** and ***findBinding*** are used for inserting a new binding and looking up a binding respectively. One other method of this class is

the ***listen*** method. This method listens the given port, and creates a thread for every request. Another method is the ***resolveRequest*** method. When a request arrives it is a sequence of characters. This method parses the given String, creates a new ***Request*** object and returns that object. ***resolveResponse*** method is anonymous to ***resolveRequest*** method. It does the same thing for a response. Another method in WebConnector class is the ***send*** method. This method sends the given ***request*** or ***response*** to its destination.

2.1.2.4.3 User Class

The ***User*** class is designed to keep the information of the users and handle their request. Since SmartGuardian will give different access rights to different user classes and will keep user based statistical information, the identification of the users is very important. Therefore before connecting to the internet the users will have to login. By this way, the users will be identified with their username and passwords. The ***username*** and ***password*** properties of the User class are necessary for this purpose. The ***gid*** property is the id the of the user group to which the user belongs.

Users can access the Internet from different computers. Therefore, when a request arrives, to determine the user making the request we need to keep the IPs If a user is does not make a request for a determined time his/her session will be timed out and the user will need to login again. So we will keep this information in the property ***lastRequestTime***. ***checkPasswor*** method will be used when a user is trying to login.

2.1.2.4.4 Policy Class

A policy is a set of rules that determines the action (allow/deny) of SmartGuardian Wall to take when a request or a response to a request arrives, and the ***policy*** class

keeps the information of a policy. Its ***rules*** attribute keeps the rules that form the policy. Its ***read*** method reads policy, whose pid attribute is given, from the database to the policy object. ***addRule*** method adds the given rule to the policy object. This method is used while creating the policy object. Another method of this class is the ***verifyRequest*** method. Given a request, this method decides whether the request is to be allowed or denied. It tries to match the request with the rules in the policy. When a rule matches the request, depending on the action property of that rule the request is allowed or denied. ***verifyResponse*** is anonymous to ***verifyRequest***. It checks whether a response to a request should be allowed or denied.

2.1.2.4.5 Rule Class

When a request or a response arrives, it's the defined rules that determine whether request or response should be allowed or denied. The ***action*** method of the ***Rule*** class is the action to take when a request or response matches that rule. If the value of this attribute is true the request/response is allowed, otherwise it is denied. The ***log*** attribute determines whether the request/response that matches this rule should be logged or not. If the value is true, then the request/response is written to the database. The ***priority*** attribute is very important in that, when a request/response matches more than one rule, the rule having the highest ***priority*** is applied. The other attributes, which are ***protocols***, ***applications***, ***domainClasses***, src***Ports***, ***destPorts, FileRules*** and ***timeRange***, are used while matching a request/response to a rule. For example, if the ***srcPorts*** attribute contains the source port of a request then the source of the request matches the rule. Otherwise it doesn't match the rule. If all the properties of a request/response match the rule then the request/response matches the rule. The read method of this class reads the rule with the given Rule_id from the database and constructs the rule object. The other methods are used while constricting the rule and matching a request/response to the rule.

2.1.2.4.6 FileRule Class

This class is used in Rule class for determining the files are denied to download and that are not. A file matches a file rule if the name of the file contains one of the words in ***keywords*** attribute, its extension is one of the extensions in the ***extensions*** attribute and its size is smaller than the ***maxSize*** attribute.

2.1.2.4.7 DomainClass Class

The ***domains*** attribute keeps the list of domains, URLs and IPs that are in this domain class and the ***containsDomain*** method is used to learn whether a given domain is in this class or not.

2.1.2.4.8 Request Class

When a request arrives all the information necessary for deciding whether to allow or deny is kept in this class. This information includes the ports and protocol used, the time at which the request is done, the application that made the request, source IP, destination IP, size of the request, the domain to which the request is done and the user that made the request. This class also includes the data, i.e. the content, of the request and the information whether the request is allowed or denied. One of the most important methods of this class is the modify method. This method modifies the HTTP headers according to the rules defined by the administrator. For example, the administrator will be able to add a rule that instructs the system to change the user agent to some other value, possibly a false one.

2.1.2.4.9 Response Class

The response class is analogous to the request class. It keeps all the information about the responses to the requests. The kept information is same as the requests information. The only difference is that response class does not have the field called application. Also the modify method is different from the request. The ***modify*** method of response parses and changes the HTML files on-the-fly according to the rules defined by the administrator. We will parse and change the html content so that certain files which can be embedded to the web page are blocked. For example, jpeg files can be blocked and the page will display a "Blocked by SmartGuardian" image instead. Similar to this, flash or java applications can be blocked and their places modified to reflect this change. Moreover javascript code will be blocked as the administrator specifies.

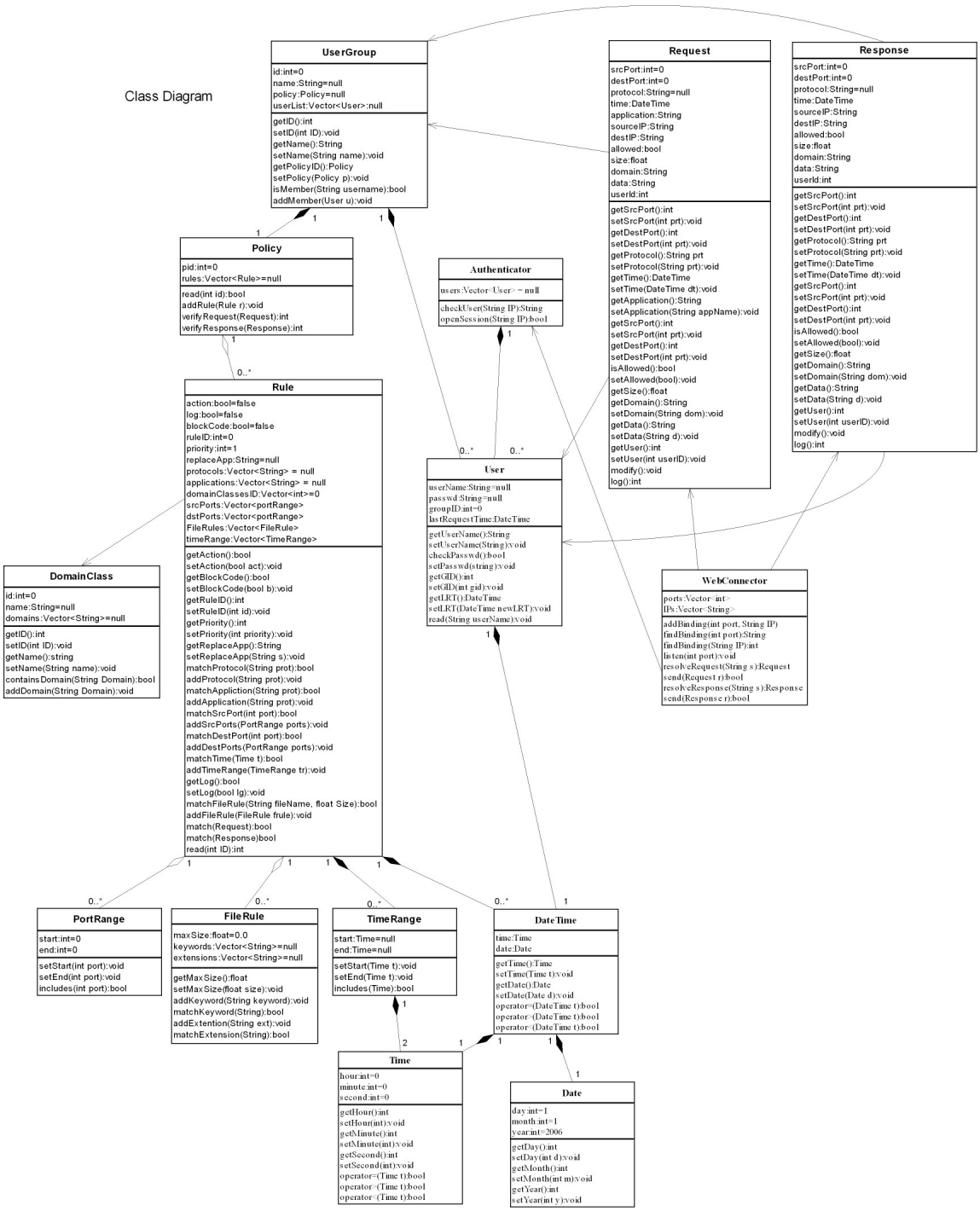


Figure 2.7: Wall Class Diagram

2.1.3 SmartGuardian Controller

SmartGuardian Controller is totally web based, it checks, the incoming requests, (Administrative Command) for authentication. If the user (network admin) is not logged in, it displays the login page. Login requests and responses are handled by its sub module called Admin Authentication.

This module handles all the administrative requests and displays the results of these requests. It can also display system logs via its sub module called Reporter.

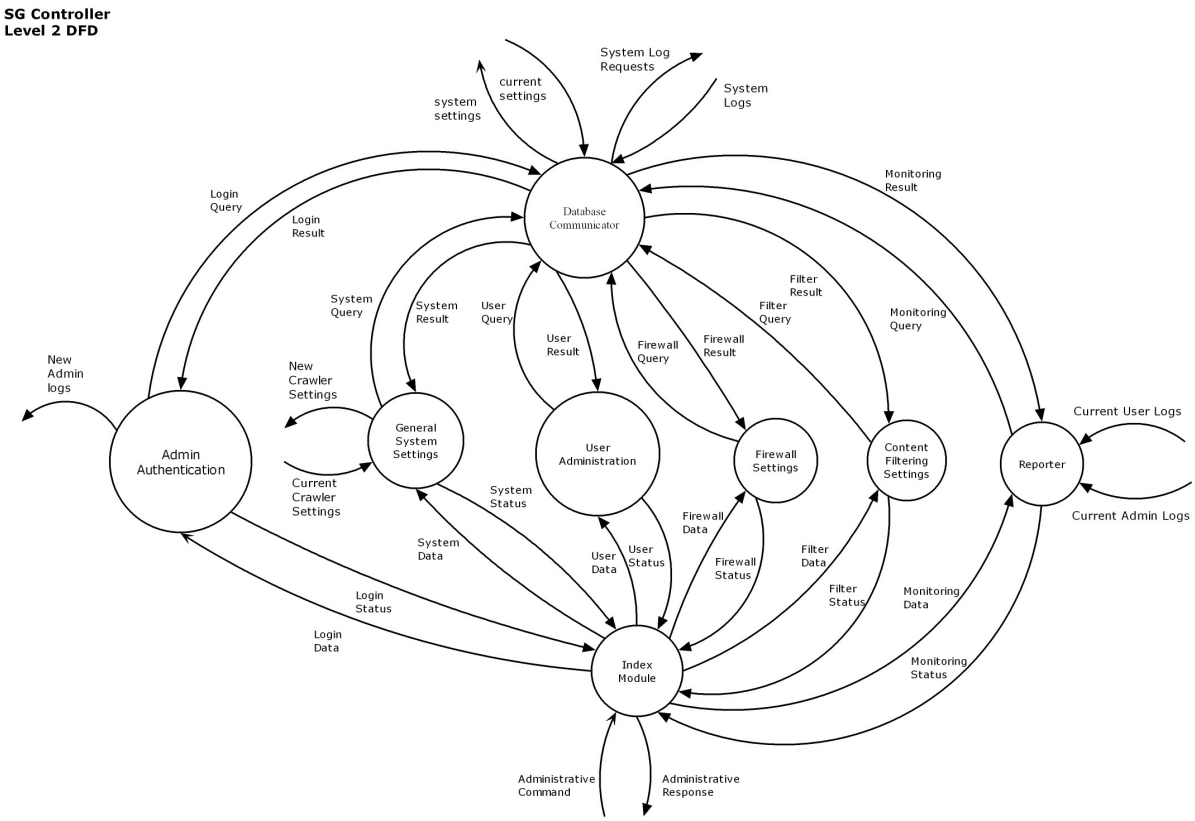


Figure 2.8: Level 2 DFD of SG Controller

When the SmartGuardian is started, the SmartGuardian Controller is in initial state. If a login request arrives, it goes to ***Checking Login*** state. In this state it firstly gets the admin usernames and passwords from the database, and checks the requested

username and password. If the login is successful, moves to the ***Idle State***. If the login is unsuccessful, it goes to ***Error Reporting*** state.

When the Controller moves to the ***Error Reporting*** state it generates the error message, and sends it to the admin. Then it logs the unsuccessful login and moves to the ***Initial State***.

When the Controller is in ***Idle State*** and an admin command arrives, the controller moves to the ***Applying Command*** state. In this state it firstly applies the command. These commands may be to create a new user, to move a domain to a domain class, to start the Web Crawler, to create a new policy, to modify a policy, etc. After applying the command the Controller logs the changes. Then it generates a message stating whether the command is successfully applied or not and sends it to the administrator. Then it goes back to the idle state.

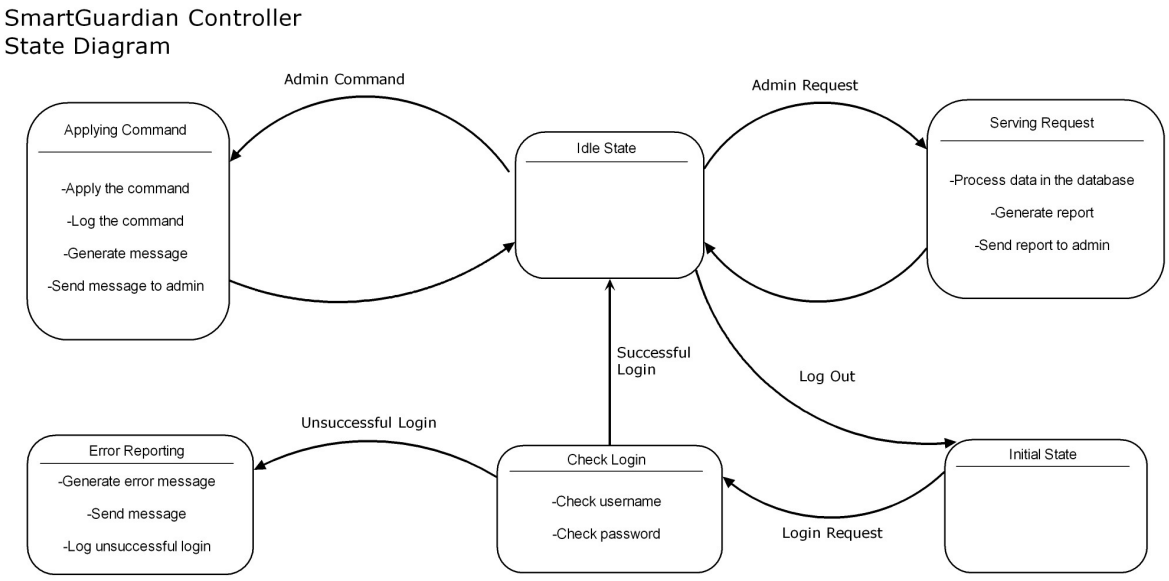


Figure 2.9: State Diagram for SG Controller

When the Controller is in ***Idle State*** and an administrator request arrives, it moves to the ***Serving Request*** state. In this state, it firstly analyzes the request. An admin can request to see statistical information about the users or domains, may request to see the current bandwidth usage, etc. Then the necessary data is obtained from the database and processed according to the request of administrator. After processing data, reports are generated and sent to the admin. Having finished with this state it returns to the ***Idle State***.

2.1.3.1 Admin Authentication

This process determines if requests are coming from a logged in user (possible network administrator) or not. Administrator logs into the system by this process. It checks username and password and records login result.

2.1.3.2 General System Settings

This module displays general system settings, and modifies those settings according to the demand of the user.

2.1.3.3 User Administration

This module displays current users of the system, new users can be added or existing ones can be deleted via this module. The status of each user is also displayed.

2.1.3.4 Firewall Settings

This module displays current setting of the firewall. Firewall settings can be modified by this module. It shows the current status of the firewall in detail, examines the modification requests for any possible inconsistencies.

2.1.3.5 Content Filtering Settings

This module displays current settings of the content filtering and its rules. Administrator can define, modify or delete policies using this module.

2.1.3.6 Reporter

System monitoring is achieved via Reporter. Reporter settings can also be done by this module. It is capable of displaying system status in various details which is determined by the user.

2.1.4 SmartGuardian Crawler

Crawler is responsible for maintaining URL classification tables of the database. Periodically, it gets the addresses from the database which is to be visited. It visits those addresses, analyses and processes their contents and tries to classify these contents then writes the result to the database.

2.1.4.1 The Disadvantages of Maintaining a Static Black List:

The Internet is an ever-changing place and a classification list today becomes quickly outdated. Moreover, it may not be in the interest of a company to download a list build by others. The company may have different view on websites unlike the company that creates those lists. The cost of acquiring a list may be high too, leaving the company to use its own resources in the classification process. Also, another disadvantage of a list is that these lists may not incorporate any web site

classification in another language. Since we are primarily aiming for the Turkish users, for example, this may a problem. Our solution below operates on any language, that is, the classification can be done independent of the web site language, once a training set is fed in to the Crawler module.

2.1.4.2 Introduction to the Crawler:

Our SG Crawler module is responsible for classification of web sites according to the web site's content. This module is important in that we will not depend on other web site classification list and we will use the list our SG Crawler module generates. Our Crawler module uses a very intelligent algorithm, which is precise. The one disadvantage is that it may need some computational power in calculating the probabilities of belonging to a domain class and also, we need some hard disk space for keeping the keywords associated with the websites.

Taking these into consideration, we will make our module to start execution at a predefined time set by the administrator. This will be at midnight or any other time when there is not much network activity, which may be slowed down by the Crawler module. A default behavior for an unclassified web site will be set by the administrator so that an unclassified website may be visited until it is classified at the next execution of the Crawler.

The detailed explanation of our algorithm follows next.

2.1.4.3 Algorithm

2.1.4.3.1 The Training of the Crawler:

Before any classification can be made, our module needs to learn which sites are in which categories to make a distinction between them. Patterns occurring in certain domain classes will be used when making a classification later on. In our design, the training will be needed to be started only once. The classification will smoothly work after this.

2.1.4.3.2 The Data Set:

We will start with a data set which composes of web sites that have already been classified. We can get this list from various resources on the Internet. The problem is that these lists do not include Turkish sites. We will add Turkish sites and other sites classified by us.

Next, we will divide this data set into two groups. The random 80% of the data will be set aside as the training set and the remaining 20% will be used as the test set. We will use the latter set as a validity checker of our algorithm and according to the success rate we will tune our algorithm to perform better.

2.1.4.3.3 Training:

The classified sites in the training set will be in our database ready to be explored. We will feed our crawler module with this set. The Crawler will then make a connection to the remote web site and download the content to be classified. Once the content is downloaded, it is needed to be pre-processed, that is, it will be stripped from unmeaningful html tags and other non-content related stuff. The output of this

step is fed to a process which will calculate word frequencies and store them as a word-frequency list in the database, related with that domain class the site is in.

Once the word-frequency lists are completed for each domain class, i.e. the readily classified web sites are exhausted; it is time for examining the date as a whole. For each word in each domain class, we calculate frequency of this word in other domain classes and we calculate the total number of sites in the other domain classes. Frequency of a word in other domain classes is found by summing up the frequency of the word in each domain class.

The training step ends when these frequency lists are built.

2.1.4.3.4 The Classification Step

Once the training step is over, word-frequency lists will be used to classify a web site that has not been classified before.

2.1.4.3.5 Introduction to Classification:

Using The Naive Bayesian Algorithm, the SG Crawler module will classify web sites according to the previous experiences. The word-frequency list is vital in this step. Our classification algorithm will start with a small set of examples but it will gradually learn and adapt to the changes the Internet asserts. We are guessing that classification will be a CPU consuming task so we will not make it "on-the-fly". According to the CPU requirements and practical applications, we will change this behavior in the future.

The "naive" word in the algorithm comes from the fact that our classification algorithm uses individual words and not word pairs or word patterns in the classification process. The gain in using word pairs is not worth the pain that is needed to maintain and analyze them.

The algorithm of classification is described in the following part.

2.1.4.3.6 Classification Problem:

For an unclassified site, here is the list of things we do for the classification:

1. First of all, for each word in the web site, find its probability of it being in a certain class.

1.1. This probability will be found by the following formula.

$$p1 = \frac{\frac{f1}{w1}}{\frac{f1}{w1} + \frac{f2}{w2}}$$

Here, p1 is the probability contribution of the word for the website to be in the domain class 1. f1 is the frequency of the word in domain class 1, w1 is the number of websites in domain class 1 and so on.

2. Select the 15 most important keywords in the website and find the combined probability.

2.1. The 15 most important keywords are found by their distance from the 50% neutral mark. The 15 probabilities that are farthest from the neutrality are the most important ones and they will contribute for the classification. Words that are nearer to the 50% mark tend to be appearing in many sites and thus they have no effect on the classification. Words like "the", "site", "a", "on" may be some examples of these words, they will be caught and marked by the crawler.

2.2. Combined probability of the fifteen words will be found by the formula below.

$$p = \frac{p1\,p2 \ldots p15}{p1\,p2 \ldots p15 + (1 - p1)(1 - p2) \ldots (1 - p15)}$$

2.3. Combined probability will be the probability that a website is classified to be in the selected class.

3. Find the probabilities for other domain classes by repeating the steps 1-2 for each domain class.

4. The domain class with the highest probability is the classification of that website. Update the word-frequency lists as needed. Add the website to the database in the appropriate domain class.

2.1.4.3.7 Future Developments:

- Taking word pairs into account.
- Taking the frequency of a word in a web site into account.
- Taking the number of pictures or sound etc. files into account when classifying.

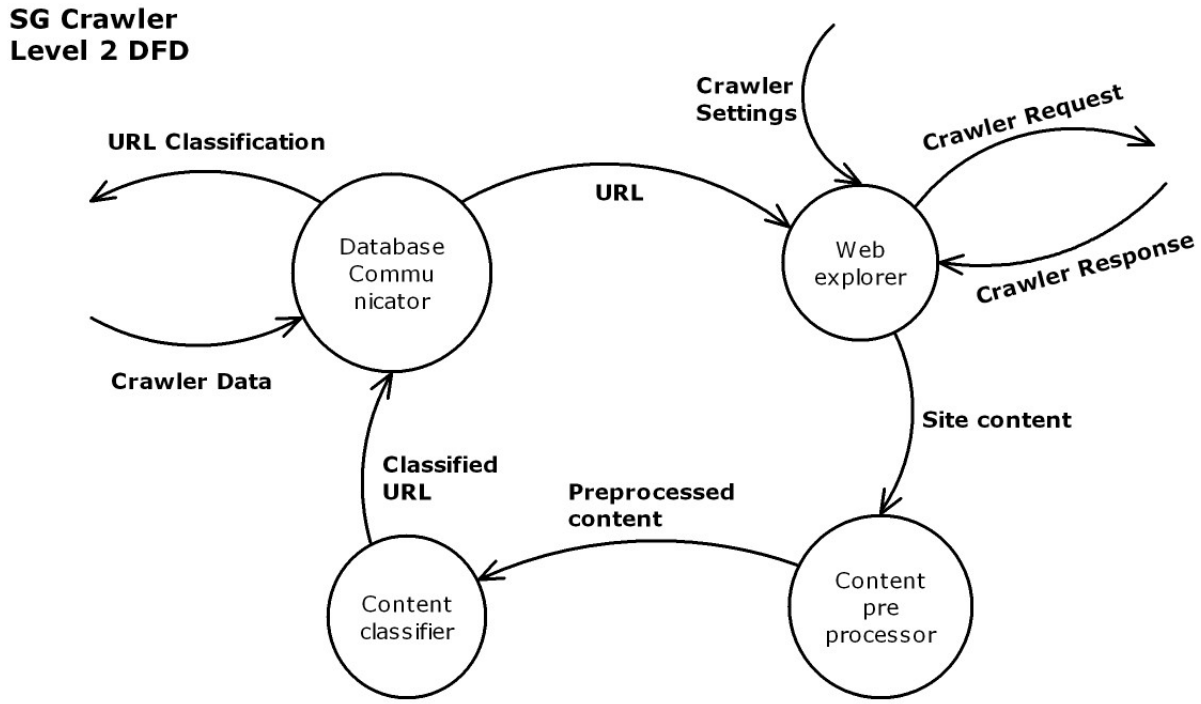


Figure 2.10: Level 2 DFD of SG Crawler

Crawler is composed of following sub modules:

2.1.4.4 Web Explorer

Web Explorer visits the addressees in the given list in appearance order; it also gets the response coming from the sites of these addresses and sends site contents to Content Preprocessor.

2.1.4.5 Content Preprocessor

Content Preprocessor inspects the site content; it processes this content and sends it to Content Classifier.

2.1.4.6 Content Classifier

This sub module takes preprocessed content, tries to classify it, if successful it puts the site to corresponding classification, if not, it marks the site as unclassified and writes the result to the database.

2.1.4.7 Class Diagrams

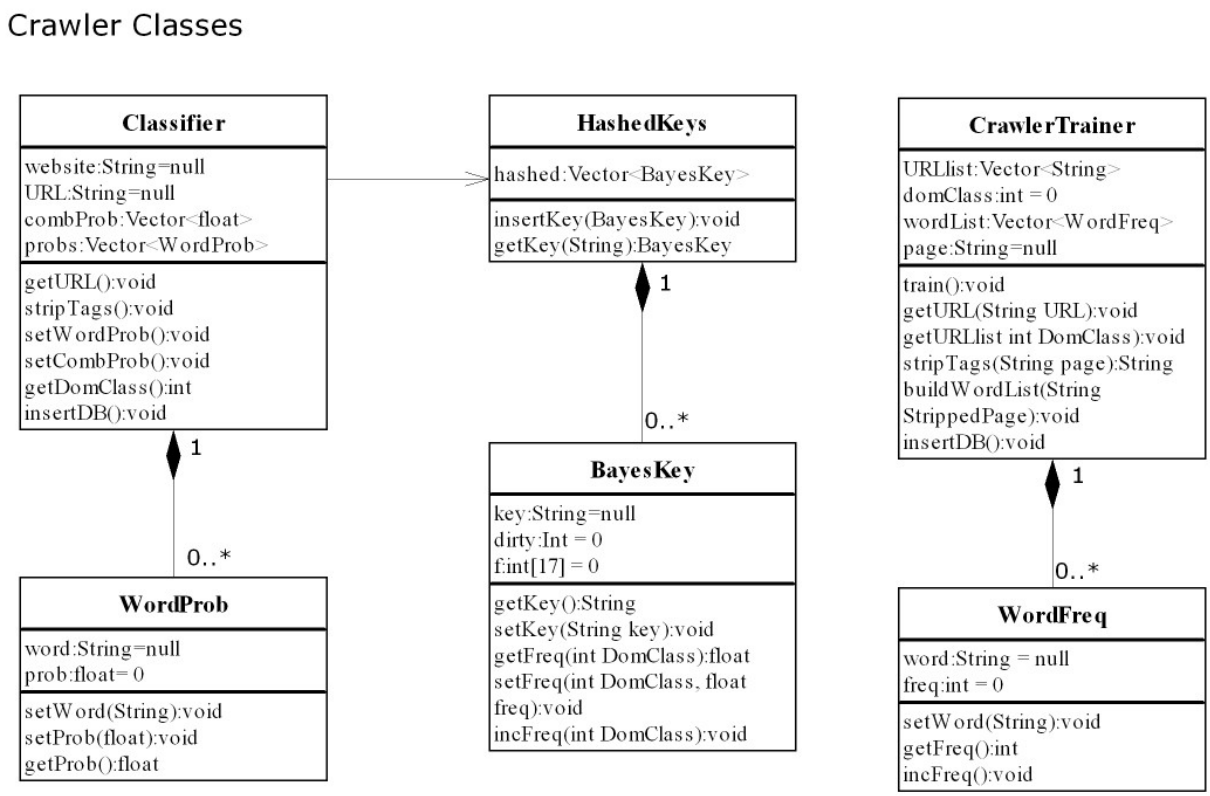


Figure 2.11: Crawler Class Diagram

Crawler Classes Explanation:

WordFreq Class:

This class is used in training the Crawler module. It has two data types for storing a word and the number of times that word has been seen in the domain class. The

three methods are for setting the keyword, getting the frequency of it and incrementing the frequency of the keyword.

CrawlerTrainer Class:

This class is the main class in training of the Crawler. It first gets the URL list of a given domain class and builds the Bayesian Key list. The getURLlist function gets the URL list of a given domain class. This URL lists is stored as a vector in this class. After this, the getURL function stores the web page as a string in the class. When the web page is stored, we need preprocessing to strip the unwanted tag and other html related stuff we do not want. The stripTags function does this preprocessing task. After this, it is time to build the word list and insert it to the database for later usage when classifying websites.

WordProb Class:

This class is used in the classification of a website. It stores a word and the probability associated with it. The functions allow you to set the word, its probability and get the probability.

Classifier Class:

This class is the main class in classifying websites. It has a WordProb vector as a data, it builds this vector and computes the combined probability for each domain class. The combined probabilities are stored as a float vector in the class. The domain class with the highest probability is the class of the website. This class will in summary implement the algorithm described in the Crawler Section of this document. After the classification, necessary elements in the database will be updated to reflect the classification contribution.

BayesKey Class:

This class is used by the HashedKeys class to build the hash version of the BayesKey objects. It holds keyword and the associated frequency in the 17 domain classes. The functions implemented are for getting and setting the BayesKey word and frequencies. Note that the class also has a dirty bit to keep track of modifications. After the program stops, the modifications will be dumped to the database in the disk.

HashedKeys Class:

This class is basically a database disk image in the memory so that the program runs faster, with minimal disk access. It uses hashing to fast access BayesKey objects.

2.1.5 Authenticator

LAN Users are allowed to access the Internet only after logging into system. LAN users log in to the system from a web page generated by Authenticator. This module checks the user name and password for validity. If login is successful, it records the result to the corresponding table in the database.

2.1.6 Database Communicator

Though it has been stated that SmartGuardian is going to use MySQL as the database, it will provide great portability if it is implemented database independent. Database Communicator is the module of the system which will make it easy to port the system to other databases.

Modules of the SmartGuardian are going to be implemented database independent. Generic functions are used in order to access database. Those generic functions

belong to the module Database Communicator and it is Database Communicator which takes care of database dependent issues. Thus, in order to port SmartGuardian to another database platform, developers only need to rewrite the Database Communicator module.

Database Communicator is used by all parts of the system which need database access. Those modules call generic database functions of Database Communicator.

2.1.7 Data Description

In this section the entities, data flows and processes are described.

2.1.7.1 Notation

The notation utilized in the data dictionary uses square brackets [ ] to denote optional items, curly brackets { } to denote items which may occur one or more times (repetition), and a vertical bar | to separate alternatives (selection).

2.1.7.2 The Data Dictionary

External Entity: LAN Internet User
Description: A user with the intention of connecting to the Internet.

External Entity: LAN Login User

Description: A user with the intention of connecting to the internet by first logging into the system. The user then will be presented a login page and after a successful login, he/she will be able to connect to the Internet.

External Entity: Internet
Description: The Internet here is not only the web page content but connection to any remote server which has a service associated with it.

External Entity: Admin
Description: Admin is a privileged user who has administration privileges. He/she will be able to manage the system, i.e. adding users, changing module settings, viewing current system status etc.

Process: SmartGuardian
Description: This is the system at its highest abstraction. The SmartGuardian Gateway is essentially a software system which controls the Internet access of organizations, taking security issues and organization policies into account.

Data Flow: LAN Request
Description: This is the request a LAN user makes with the intention of connecting to the Internet.
LAN Request = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.

Data Flow: Finalized LAN Request
Description: This is the request verified by the submodule verify request which is sent to Web Connector to be sent to the Internet.
Finalized LAN Request = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Data Flow: LAN Web Request
Description: This is the request sent by the Web Connector to verify Request for examination.
Finalized LAN Request = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: Verified LAN Response
Description: This is the data from the Internet a LAN user receives, which is verified by the system.
Verified LAN Response = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Data Flow: Finalized LAN Response
Description: This is the response verified by Verify Response submodule which is going to be sent to LAN through Web Connector.

Finalized LAN Response = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Data Flow: LAN Web Response
Description: It is the response coming from the Internet via Web Connector. It will be examined on Verify Response for validity.
LAN Web Response = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: User Login Request
Description: This is the login request a LAN user makes before initiating communication with the Internet.
User Login Request = LAN IP, LAN port no, gateway port no, user name, password.
LAN IP = IP of the login requesting client. It is the IP of the LAN user in this context.
LAN port no = The port number used by the requesting client during the communication process. It is the port number of the LAN user in this context.
Gateway port no = The port number used by the gateway server during the

communication process. It is the port number of the gateway in this context.
Username = Username of the user who is using the LAN IP. This username must match with the password in the database.
Password: Password of the user who is using the LAN IP. This password must match with the username in the database.

Data Flow: User Login Response
Description: This is the login response a LAN user gets after his/her login attempt. It is created by the gateway server.
User Login Response = {error number, error explanation} | success message.
Error number: This is the error number returned with the error message if the login attempt is unsuccessful.
Error explanation: This is the error explanation given by the gateway.
Success message: This is a confirmation that the login attempt was successful.

Data Flow: Verified LAN Request
Description: This is the request made by the LAN user which is verified to comply with the policies the user is subject to. It has the same attributes as the User Internet Request since it is verified and forwarded by the gateway.
Verified LAN Request = Source IP, destination IP, protocol, source port no,

destination port no, [additional request info], data.
Source IP = IP of the requesting client. It is the IP of the gateway in this context.
Destination IP = IP of the destination server. It is the IP of the remote server in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the communication process. It is the port number of the gateway in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the remote server in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: LAN Response
Description: This is the response made to the LAN user which is verified to comply with the policies the user is subject to. It has the same attributes as the User Internet Response since it will be verified and sent to the user.
LAN Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this

context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the communication process. It is the port number of the gateway in this context.
Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the LAN user.

Data Flow: Crawler Request
Description: This is the request made by the crawler module to the Internet. The crawler downloads web sites and classifies them.
Crawler Request = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the crawler residing server. It is the IP of the gateway server in this context.
Destination IP = IP of the destination server. It is the IP of the remote server in

this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the communication process. It is the port number of the gateway in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the remote server in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: Crawler Response
Description: This is the response to the crawler from the Internet for its request. It has the same attributes as the “crawler request” data flow since it is only a response to that request.
Crawler Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.

Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the communication process. It is the port number of the gateway in this context.
Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is processed by the SmartGuardian Crawler module.

Data Flow: Monitoring commands
Description: This is the response to the crawler from the Internet for its request. It has the same attributes as the “crawler request” data flow since it is only a response to that request.
Crawler Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.
Protocol no = Protocol used by the client and server while communicating with each other.

Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the communication process. It is the port number of the gateway in this context.
Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is processed by the SmartGuardian Crawler module.

Data Flow: Administrative Command
Description: This is a system command that the administrator issues to control the way system works.

2.2 Database Design

ER diagram is a representation of our database which is the core part of our system. In the ER diagram we have a set of entities that are designed for each module's needs.

Most important entity in this relation is our ***Rule*** entity. It has relations with ***portrange, timerange, file_rules, protocols, Policy & Domain_Classes.*** In order to define a rule to database we have a wide possibility due to having lots of attribute of file rules. These attributes are; ***rule_id*** that uniquely identifies the rule, ***priority*** for a rule since we may define some rules and that may crash due to administrators (one rule

allow a response and the other does not), for this reason we have a priority ***attribute***, besides that ***action***, allow or deny, ***date***, date of defined rule and ***description***, is short description for administrator to remember the predefined rule. In addition, ***rule*** entity has relations with protocols, timerange, portrange & Domain Class. Due to this reason, it is possible to define a time range a port range and domain classes for each rule. For each user we have user groups and for each user group we only one policy. So for a defined rule we insert it into policy table for it to be valid for users that is why rule entity has a relation with policy. Lastly there is an attribute ***logged*** that is a flag for rule that whether that the logs related with that rule will be kept in disk or not. Our logs are kept as response and request that is why logs table is in ternary relationship with domains & users due to response and request relations. ***ReplaceApp*** is a string that have the names of application (Mozilla firefox etc.) that the system checks the content of this attribute and object to write headers this value. ***Blockcode*** is a Boolean value that is used to check whether the codes that work in web pages (JavaScript or flash etc.) are allowed or not.

There is an entity ***Applications*** that has a relation with ***Rule*** table that keeps the application program such as (Firefox 1.0, Internet Explorer navigators etc.) that is a part of rule for each user group that defined by administrator as a part of rule. It will also have to be matched in processing of requests.

For each rule as mentioned above there are some time intervals that are kept in ***timerange*** entity. It has two attributes ***start*** & ***end.*** Type of these attributes is ***time*** and they are keys of this entity, ***portrange*** entity that has two attributes (***start*** & ***end***) for upper and lower bound of ports that are allowed or denied.

For protocols that a rule valid we have ***Protocols*** entity that consist of for five types of Protocols such as HTTP 1.0 HTTPS etc. Since a rule might be valid more than one

protocol. Besides that we have ***File_Rules*** table that keeps an id and size attribute. This entity has two relations with ***Keywords*** & ***extensions*** entities. Keywords entity has ***word*** attribute that keeps some key words that will be used in filtering process. Extensions entity has extension attribute that contains extensions of file.

Domain_Classes entity has an id ***did & name*** attribute that defines that domain. In ***Domain_Classes*** has ***domain.*** Domain entity is a set of URLs. In URLs attribute there will be kept IP "10.34.234.13", domain "www.ceng.metu.edu.tr" or URLs "http://www.gospatial.com/html/viewers.html". These are some sample data that can be stored in here. Each domain class has classification keys for Bayesian algorithm which is adapted to our Crawler module implementation in Crawler module. We have entity ***Bayes_keys*** with attribute of ***word*** and ***f1, f2… f17*** that is useful attributes for Bayesian algorithm. In other word, for each attribute there will be kept a frequency number for each domain class. Our system keeps these words in database.

The other important entity is ***User***. It keeps whole user in local area network. Name and surname of each user are attributes of this entity. It also keeps username & passwords that are checked in user authentication process. Each user has an e_mail address that is kept in the system in order to communication of system with the user. Each user must be a member of a ***user_group.*** Each user group has an id ***gid*** and must have unique policy. The policy table is consists of many rules that are specified for that policy, in other way for user groups.

In this ER diagram we have two relation ***response*** & ***request.*** These relations are going to be a table in the database. Each user has some requests and most requests has response that have a set of attributes such as ***source_port, dest_port, source_ip, destination_ip, time, protocol*** etc. These attributes are used in validation and security

consideration of request. Likewise such consideration will be achieved for responses. Besides that these attributes are considered as part of logs since system keeps requests and response as logs.

Besides the entities in ER, we have an isolated entity ***admin*** that will be stored into database as table. Since system use this table for authentication process apart from other tables, there is a relation between admin table and user table for keeping information that who created the user.

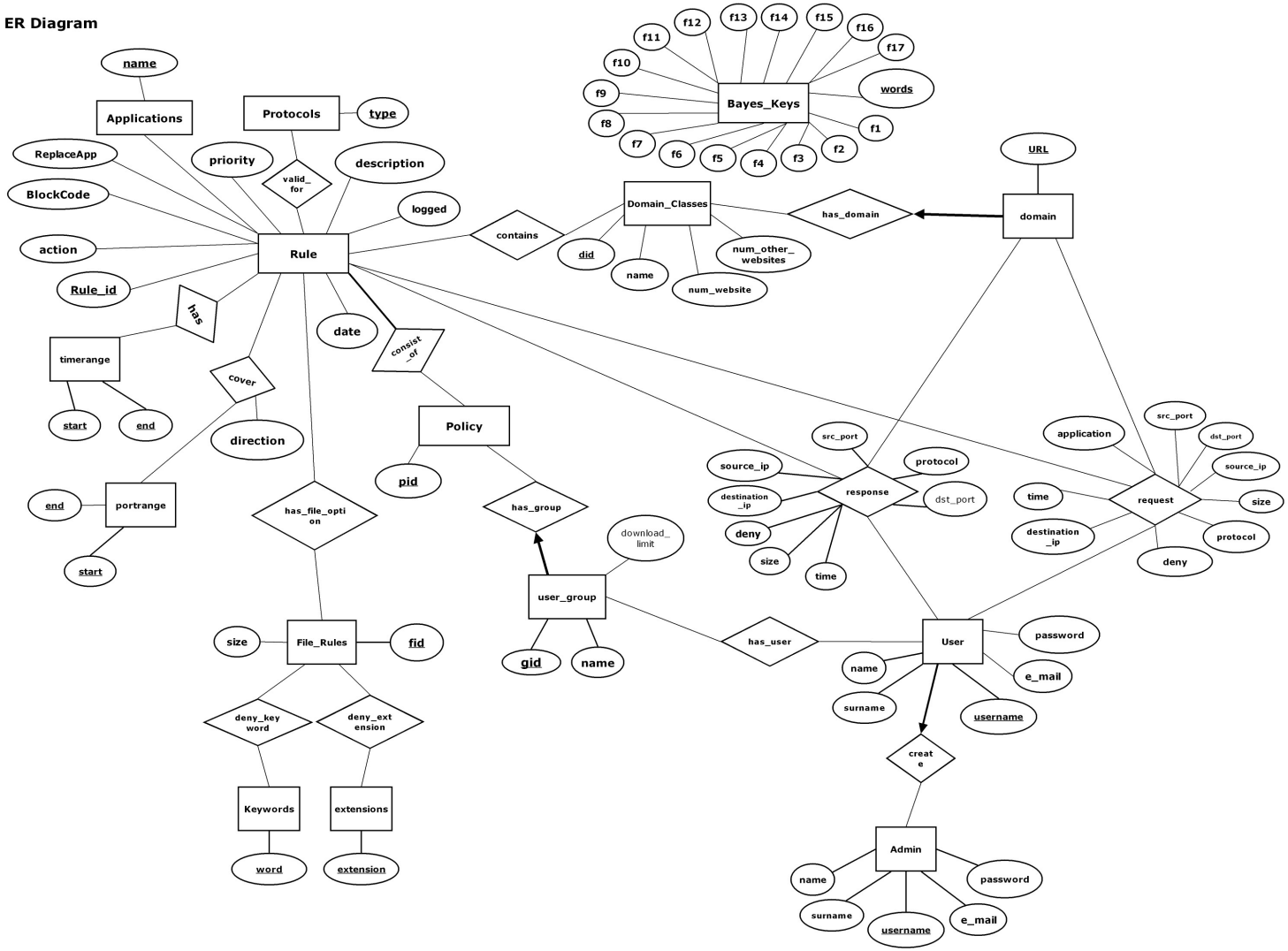


Figure 2.12: ER Diagram

Below we have the data dictionary of ER. In the data dictionary we have entity descriptions and some relation descriptions of our ER diagram. I the data dictionary explanations of entities and relations are given. There we have also description of attributes of entities and relations.

Entity type name: Rule	
Definition: Rule is the core part of the system database. Since it keeps all related information that an administrator can define for a rule. ***Rule_id*** for primary key, ***date*** for updating issues, ***priority*** for preventing crash problem, ***description*** for remembering the rule purpose, ***action*** for allow/deny the request or response when conditions satisfies. Besides that it has relations with ***timerange*** and ***portrange*** entities since admin will specify range for time and port for a rule. It is also possible for a rule to extend more than one protocol. That is why we have ***Protocols*** and a relation with this table. We have a file rule for a specified rule with ***has_file_option*** relation. We have file size for allowing maximum amount of file size. File entity has relation with extensions and word in order to eliminate file with specified extensions or word in name of file. Each rule belongs to a Policy which is unique for a user group. Besides that each rule has relation with Domain_Classes due to define domain classes for apply the rule. Lastly, Rule entity has a relation with ***Logs*** entity in order to keep logs with related rule if it is specified.	
Data element:	Rule_id
Description:	An integer that uniquely identifies Rule
Where used:	Primary Key to Rule table
Data element:	action
Description:	Flag for rule to allow/deny [0 | 1] request or response
Where used:	In checking the rules that allow or deny if condition is satisfied
Data element: Description:	Priority An integer for priority of specified rule [0 | 10]

Where used:	In checking the priority of a Rule for applying
Data element:	date
Description:	Date values of rule that keeps the day of rule description.
Where used:	For eliminating the rules that are too old in order to make system efficient
Data element:	Description
Description:	A long string ***char[50]*** a description of rule
Where used:	Making administrator to remember the rule that s/he defined
Data element:	logged
Description:	A Boolean flag for understanding whether logs with that rule will be kept in disk or not
Data element:	ReplaceApp
Description:	A string value that keeps the application name that the system writes to header of request as seen application (not the using one)
Data element:	BlockCode
Description:	A Boolean value that will be used in whether blocking the application or not during rule matching process and blocking porocess

Entity type name: portrange
Definition: portrange entity will be used to keep for specified rule in order to restrict or allow the port range for users. Start represent the lower bound and end represents upper bound of port

Data element:	start
Description:	An integer that represents the lower bound of a port for specified rule.

Data element:	end
Description:	An integer that represents the upper bound of a port for specified rule.
Entity type name: Applications	
Definition: Application table keeps all application names which is unique for each application. This table is created like Protocols table that whether defined application is allowed or not. Like web browsers.	
Data element:	name
Description:	A string value that is primary key to Applications table which describe the application that is defined in rule definition
Entity type name: timerange	
Definition: portrange entity will be used to keep for specified rule in order to restrict or allow the port range for users. Start represent the lower bound and end represents upper bound of port	
Data element:	start
Description:	A time value that represents the lower bound that restrict user in using internet.
Data element:	end
Description:	A time value that represents the upper bound that restrict user in using internet.
Entity type name: File_Rules	
Definition: Is an entity with attribute size; it has a relation ***has_file_rules*** relation with Rule entity. It keeps the file rules in a given rule; maximum downloadable size of file, allowable or deniable files extensions and keyword in the file.	
Data element:	size
Description:	The value that keeps the maximum downloadable size of a file

Where used:	It is used in getting the bandwidth limit of policies for specific user
Data element:	fid
Description:	Primary key for File_Rules entity
Entity type name: Keywords	
Definition: Keeps a set of keyword that is used in both file rules of policies and ***Web Crawler*** module. It will be one of the largest tables of the database. It has an attribute Word as a string	
Data element:	words
Description:	Contains a string like "***sex***" that give hints as a keyword for the what the file contains
Where used:	In data mining issues of ***Web Crawler*** and file comparing file names with those keywords - in string matching -
Entity type name: extensions	
Definition: Keeps a set of extensions of files that will be used in file rules for which is for a policy.	
Data element:	extensions
Description:	Contains a string like "***mp3, jpeg, mpeg, zip***" that defines the type of file
Where used:	It is used in file rules application of a policy whether the file with this extension is allowable or not
Entity type name: Domain_Classes	
Definition: In the database we have 18 different types of domain classes which have ***has_domain*** relation that keeps domains that are inserted by either user or Web Crawler. Each domain has id and name.	
Data element:	did
Description:	Primary key for domain classes entity

Data element:	name
Description:	Like did it is also some default name that is an attribute of Domain_Classes
Where used:	It is useful and more recognizable in defining a ***Policy*** & ***Domain_Classes*** relation
Entity type name: Protocols	
Definition: Protocols keeps some protocol types in type attribute since a rule may be defined more than one protocol (HTTP & HTTPS etc.).	
Data element:	type
Description:	A string with CHAR(5); that keeps protocol type for a specified rule.
Entity type name: domain	
Definition: the entity set is composed of URLs and domains which are unique for each domain	
Data element:	URL
Description:	URL is a string that specify the domain; it can be an IP ***10.345.35.7*** or ***www.metu.edu.tr***
Where used:	It will be used in checking of request or response of a user whether the domain is allowable or not. Besides that Web Crawler will be used for data mining process of related domains web pages
Entity type name: user_group	
Definition: It represent types of user; each user has unique policy	
Data element:	gid
Description:	It represents the user group name as an integer

Where used:	Primary key of the entity
Data element:	name
Description:	Name of the user groups, in order to define user groups with better understandable way.
Entity type name: Bayes_Keys	
Definition: A table for defined Domain Class that is unique for each entity. It keeps ***words*** attribute in order to classify a page by these keywords using Bayesian algorithm and a ***frequency*** value for each word.	
Data element:	words
Description:	A string for each class that is given for classifying the URLS by applying Bayesian algorithms.
Data elements:	f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16,f17
Description:	An integer value that specifies the frequency of keyword for each domain class.
Entity type name: Policy	
Definition: Entity that has relation with both rule and user_groups; the aim of this entity is to keep all rules for a specific user_groups in one policy since if we canee a user_group and all related rules with that group.	
Data element:	pid
Description:	Identifier of the policy entity as primary key. (Integer value)
Entity type name: User	
Definition: It keeps the entire user that benefit from Internet in LAN. Besides that it has attributes of description of the user. Each user that has ***request*** & response relation	
Data element:	name

Description:	String ; name of user
Where used:	Identifying user in order to send a message
Data element:	surname
Description:	String ; surname of user
Where used:	Identifying user in order to send a message
Data element:	username
Description:	String ; username of the user : Primary of entity
Where used:	Is checked in authentication of user
Data element:	password
Description:	String ; password of the user
Where used:	Is checked in authentication of user
Data element:	e_mail
Description:	Email address of the user
Where used:	It will be useful for sending system administrator messages

Entity type name: Admin

Definition: Admin will be also stored in database. But it does not have any relation with other entity sets. It all have same attribute with user. However admin authentication will be achieved from system directly.

Relation type name: response

Definition: Response is a relation between ***User*** and ***domain***. For each request that send internet throughout system, related domain returns response. Then some information related with response stored in database. Attributes of response taken from package header are below:

Data element:	source_ip
Description:	IP address of domain that have been requested
Where used:	Checking for security and policy consideration
Data element:	destination_ip
Description:	IP address of user that requested related domain's pages
Where used:	Describing the computer
Data element:	deny
Description:	Status of related response is valid or not
Where used:	Checking of validity of request
Data element:	size
Description:	Size of pages or files that the user requested
Where used:	Comparing with download limit of the policy that the user in
Data element:	time
Description:	Time of the response
Where used:	Checked in updating & deleting response from database
Data element:	src_port
Description:	Port number of the response that web server connect in.
Data element:	dest_port
Description:	Destination port number of the response that users want to use.
Data element:	protocol
Description:	Protocol of the response
Where used:	For security consideration

Relation type name: request	
Definition: Request is a relation between ***User*** and ***domain***. Each user has a request during internet application for a specific domain. Then some information related with request stored in database. Attributes of request taken from package header are below:	
Data element:	source_ip
Description:	IP address of user that requested related domain's pages
Where used:	Describing the computer
Data element:	destination_ip
Description:	IP address of domain that have been requested
Where used:	Checking for security and policy consideration
Data element:	deny
Description:	Status of related request is valid or not
Where used:	Checking of validity of request
Data element:	size
Description:	Size of pages or files that the user requested
Where used:	Comparing with download limit of the policy that the user in
Data element:	time
Description:	Time of the requested
Where used:	Checked in updating & deleting requests from database
Data element:	src_port
Description:	Port number of the request that user connect in.
Data element:	dest_port
Description:	Destination port number of the response that web service

	serve for related requests
Data element:	protocol
Description:	Protocol of the request
Where used:	For security consideration

2.3 Graphical User Interface

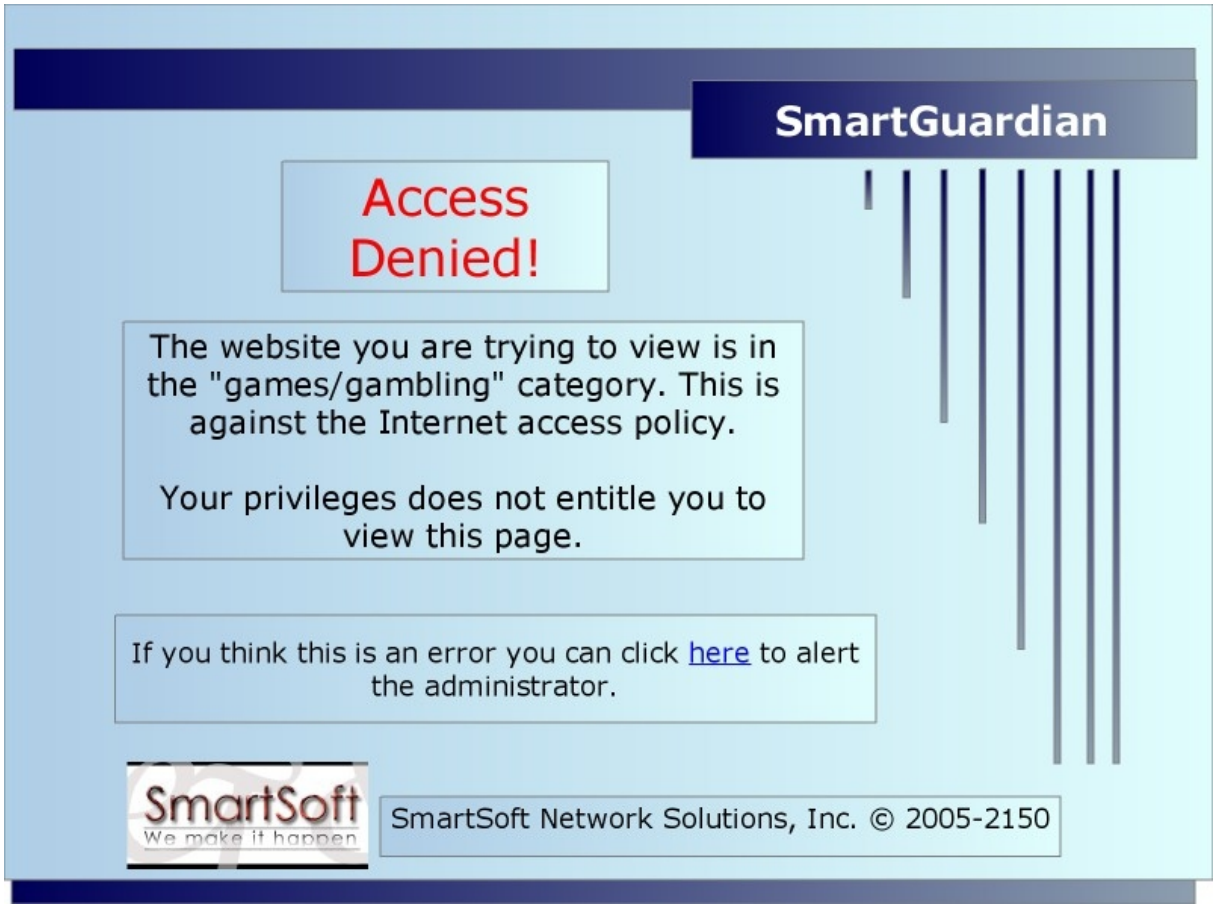


Figure 2.13: Access Denied Warning

In the figure 2.13 you can see the web page that is displayed when the user tries to access a denied page.

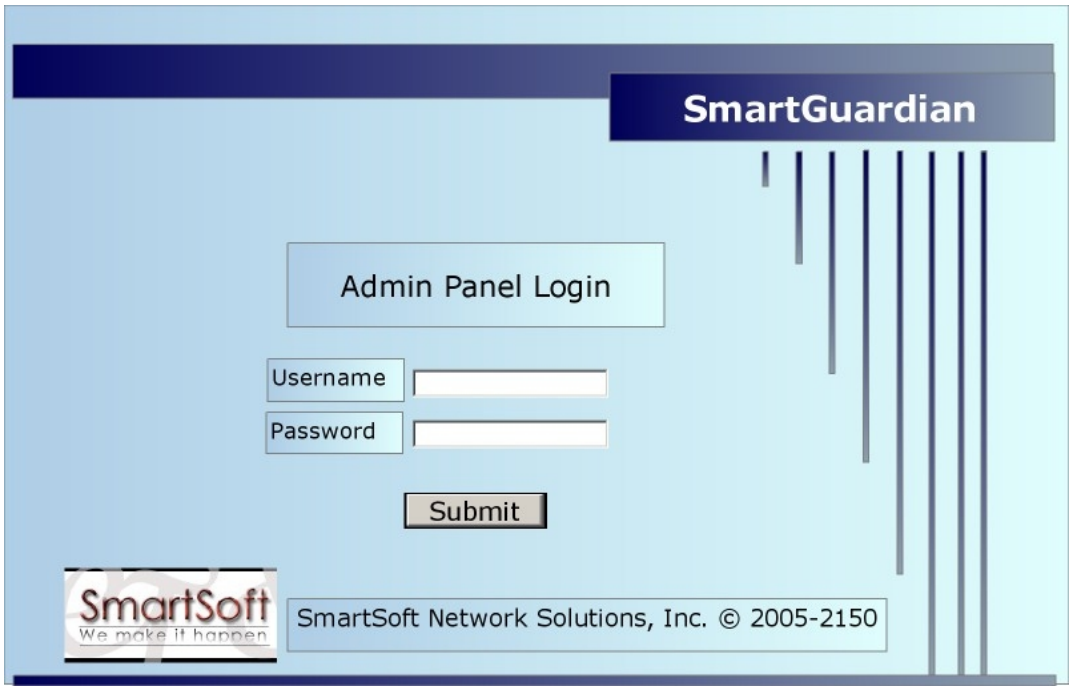


Figure 2.14: User Interface Login Screen

In the figure 2 .14 you can see the login page used when the admin wants to login to the system to make changes.

Figure 2.15: Administration Page

In the figure 2.15 you can see the welcome page displayed after the login to the system.

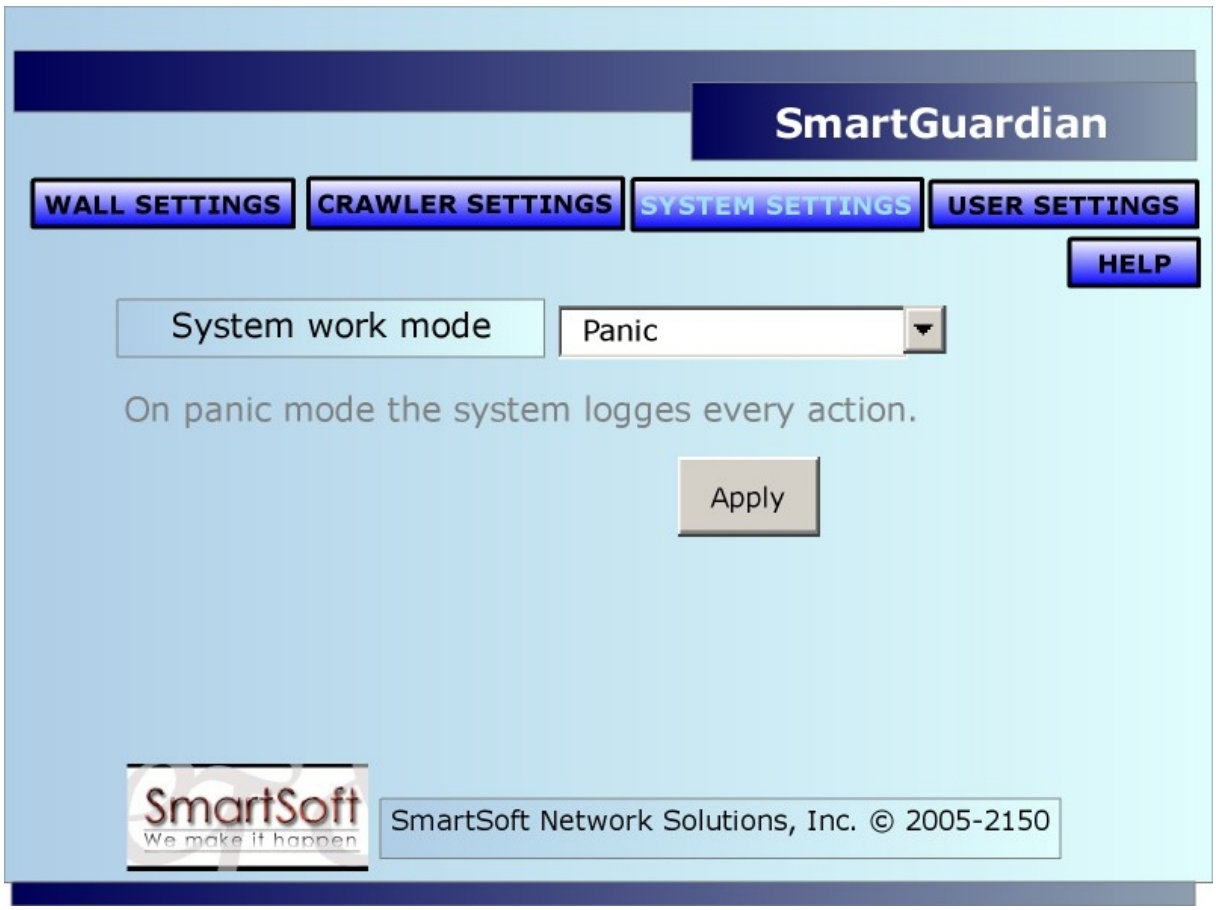


Figure 2.16: System Settings

In the figure 2.16 you can see the system settings page displayed after selecting sytem settings from the web menu.

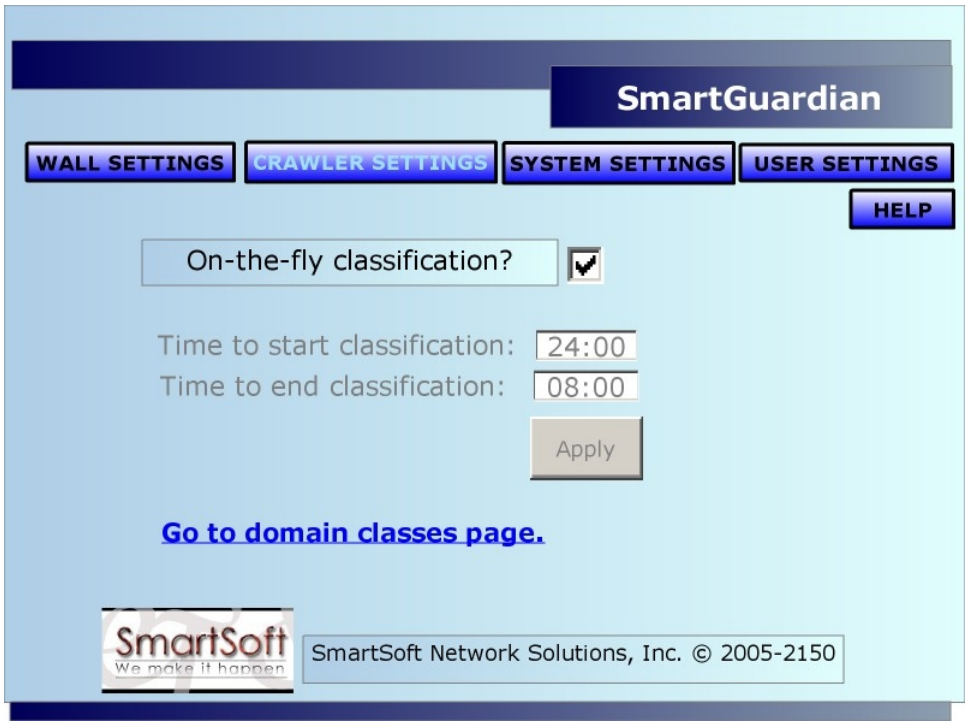


Figure: 2.17

In the figure 2.17 you can see the crawler settings page displayed after selecting crawler settings from the web menu.

2.4 Behavioral Description

You can see the use-case diagram below. It shows actions from a sample space. One can notice what the administrator and/or users are expected to do by interacting the software system.

As can be seen, the administrator can initiate a crawling session, configure the system in a detailed sense and monitor the system status. A user is expected to request a web page using the SmartGuardian software.

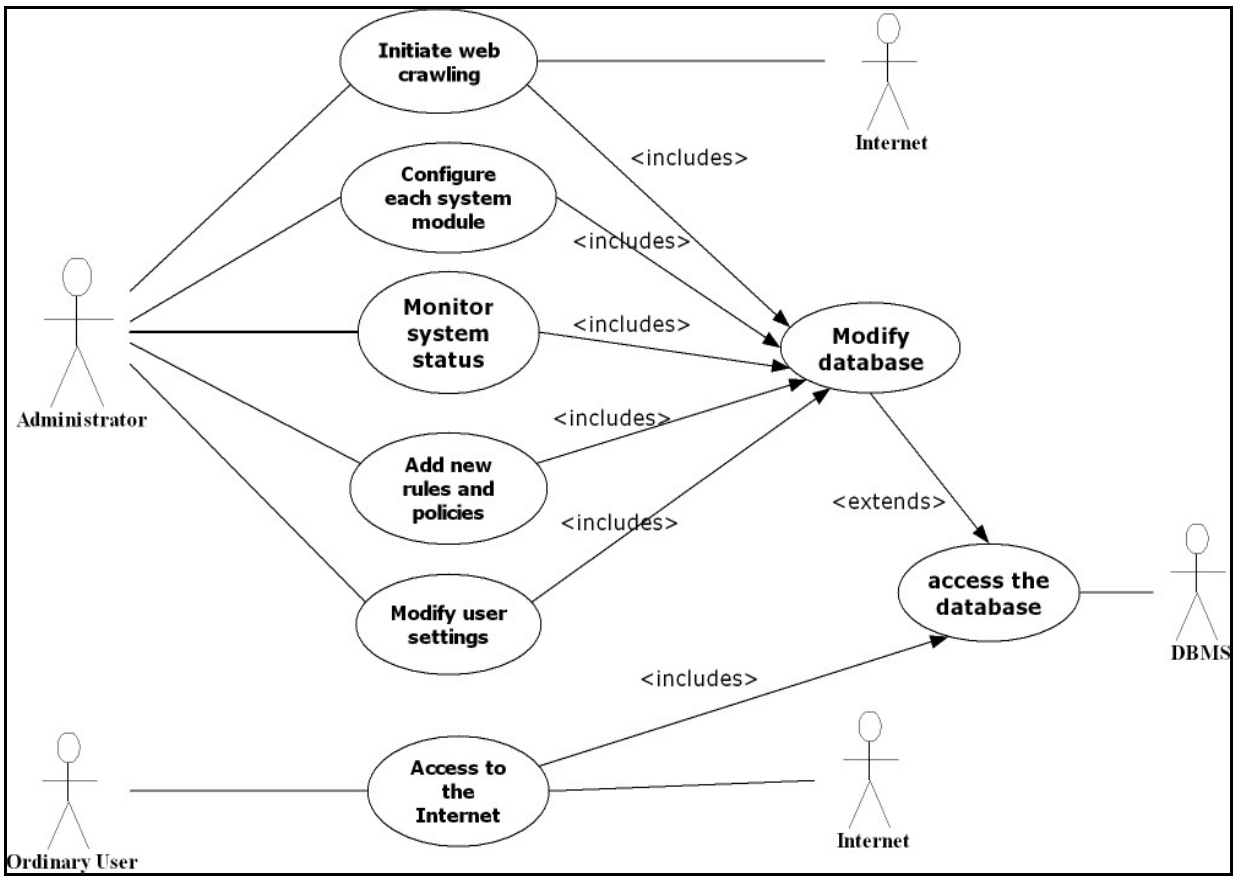


Figure 2.18: The use-case diagram

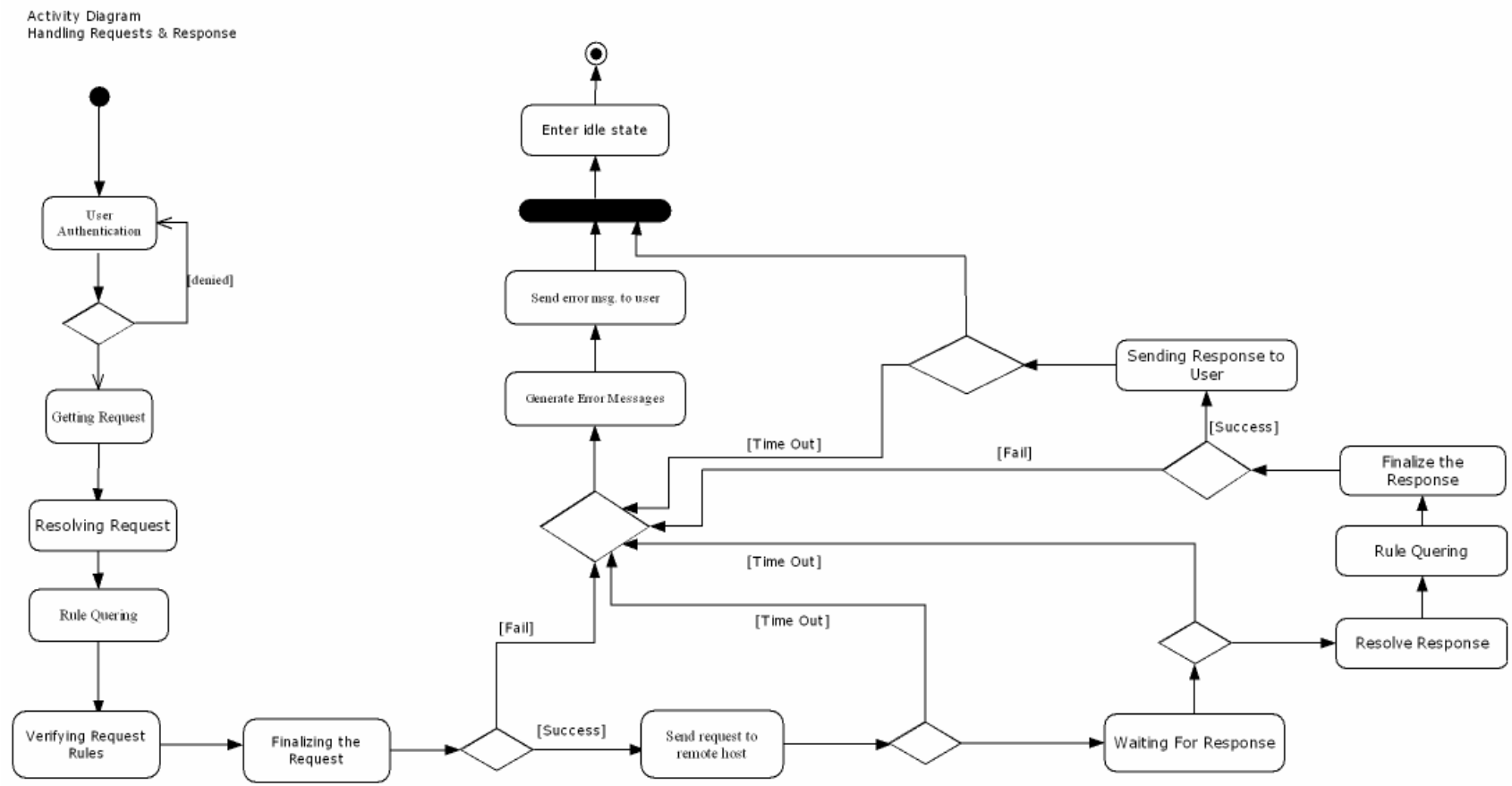


Figure 2.19: Activity Diagram of Handling Request & Response of user

In figure 2.13 we have a revolution of an activity that is achieved by user represented as an activity diagram. User has an authentication process at the beginning; if he/she is not successful, authentication process will be repeated again and again. After authentication the request of user will be get by system in order to analyze. Firstly, all information related with request will be get from request; destination IP, source IP, port, protocol etc. After that rule table will be queried in rule querying phase. After that it is going to be analyzed in Finalizing the Request part. Whether the request satisfies the rule or not, consequently it will fail and go to Generate Error Messages part and related report will be send to the user that have the request. If it is an innocent request and satisfies the rules that is defined in database, than it will be sent to remote host. There is possibility to expire due to time restriction due to networking of computer, if this situation happens than time out activity is activated and error message created. Otherwise the request will be successfully sent.

A request has a response in network system and for a specific request it is important to have related response that is sent by some web server. That is why after sending the request to remote host the system waits for response. There is again time restriction of this activity state, for that reason time expiration is considered. If the system gets related response correctly and quickly the response will be resolved for having information (source IP & port, destination IP & port, protocol etc.). As it happen in request part rule table queried by rule querying in order to match the response information with rule. Matching the appropriate rule system decides to allow request or not. If not, error generating state will be activated; else system sends the response to user and transact into idle state.

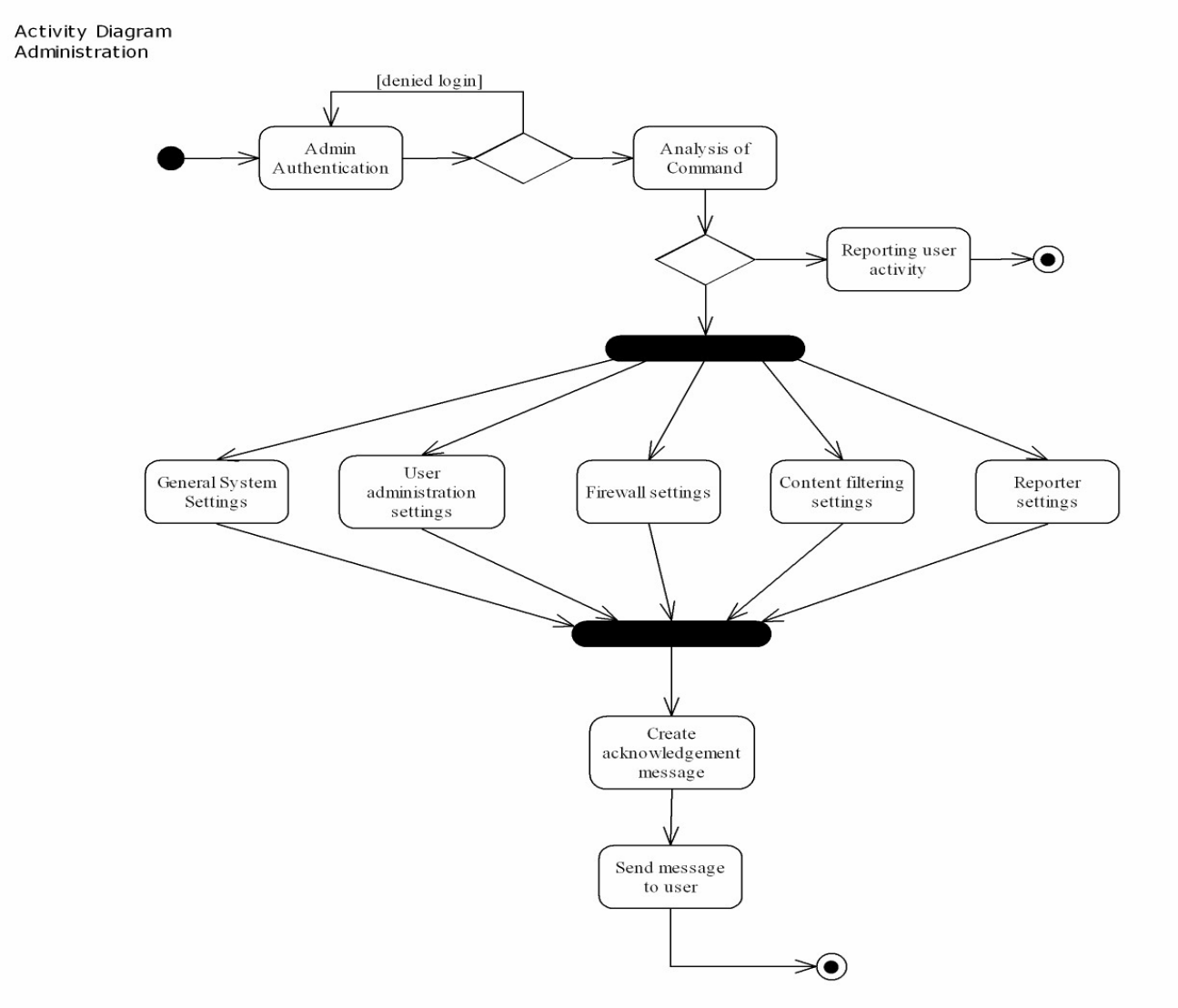


Figure 2.20: Activity Diagram of Handling Administrator Command

In figure 2.15 we have a lifecycle of an administrator command. Firstly, like users the administrator should also authenticate to the system in order to effect to system settings or monitoring action of users. After authentication the user has choices, one is just monitoring user actions history with Reporter sub module. It is done by investigation of reporter to log scripts that are stored in log files. The other choice of administrator is system settings. It would be general system setting, user setting, firewall setting, content filtering setting or reporter setting. Each of these settings is done by PHP interface. Then an acknowledgement will be sent to the administrator

whether the settings are successfully done or not. That is the end of the administrator command cycle.

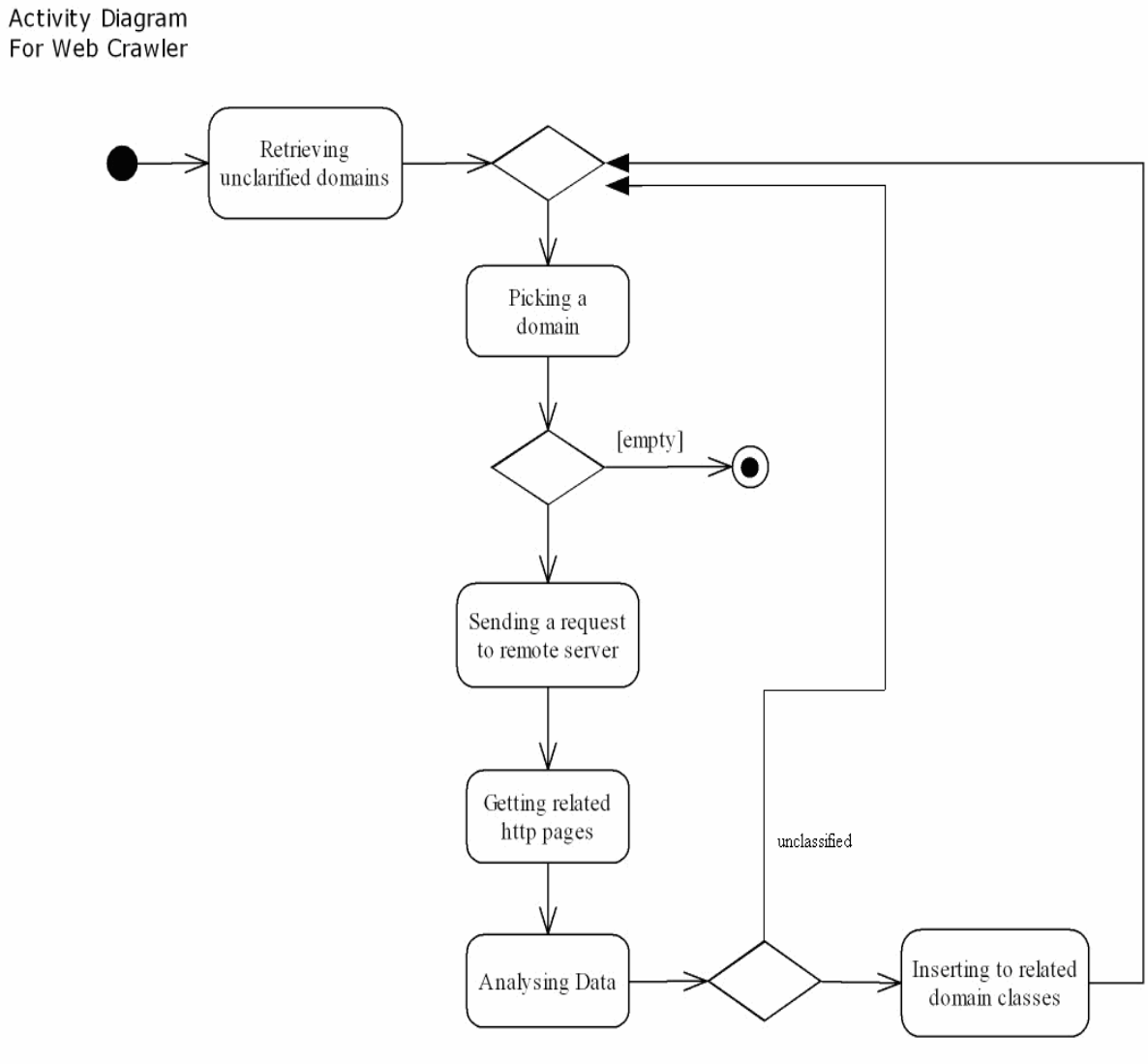


Figure 2.21: Activity Diagram of Web Crawler lifecycle

The aim of the web crawler is domain classification of unclassified domains in the ***Unclassified Domain Class*** table. It gets domain from URL list to a buffer. After investigating each domain and applying data mining process after getting related HTTP pages it try to classify these domains into one of the 18 different domain

classes. If it is not successful after applying data mining it skip insertion process and pick another domain.

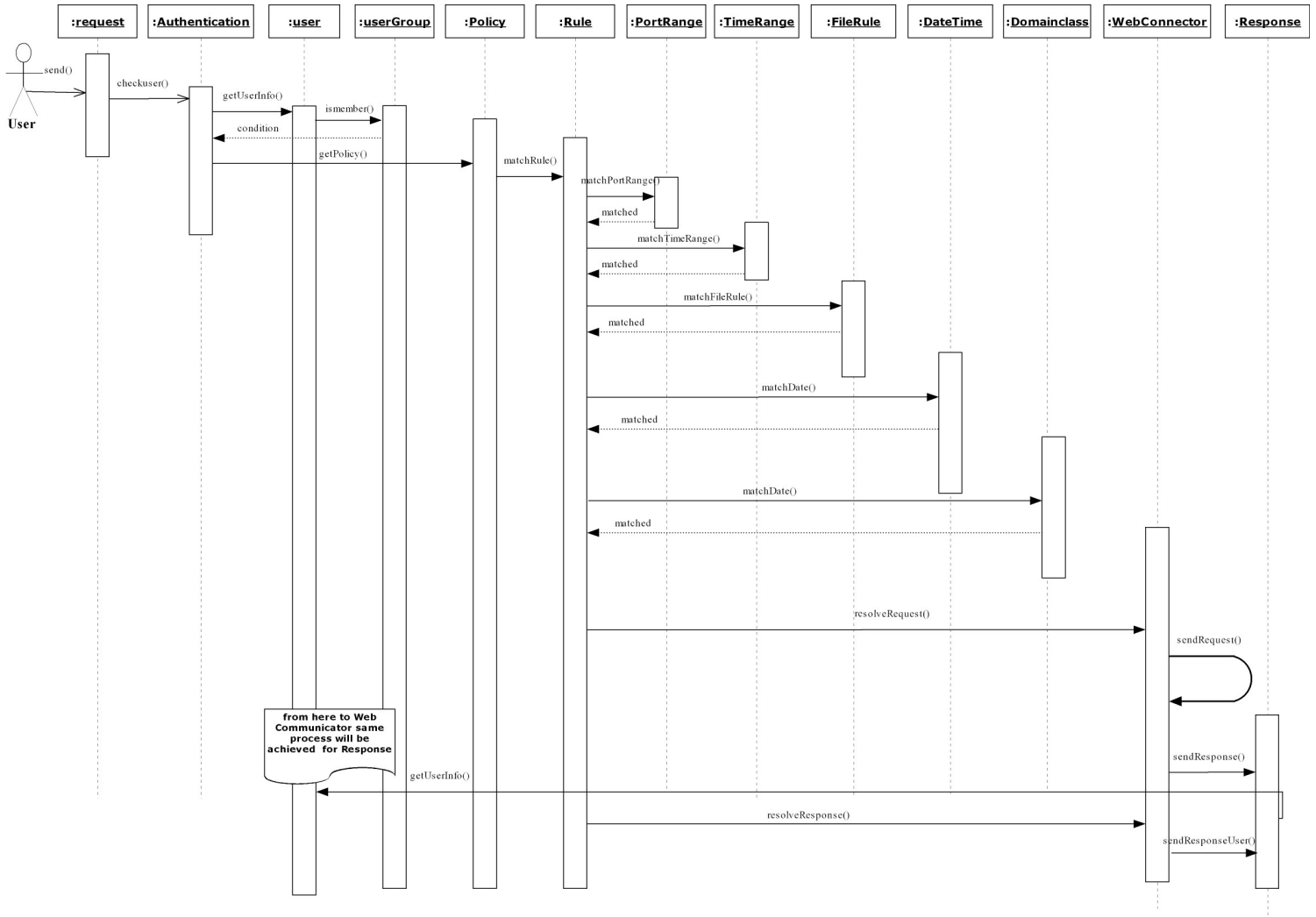


Figure 2.22: Sequence Diagram

UML sequence diagrams model the flow of logic within system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. In the figure above there is scenario based sequence diagram of ***SmartGuardian Wall***. It is activity set of classes of the system.

A user sends his/her request and request class get this request for processing. Authenticator class gets user info and get user information (username, password), and check for validity. During this validation this class sends user class for learning that this user exists or not then from user group class sends Boolean value (condition) to authentication. After that Policy class gets related policy for that request by user group information. After that, rule class match the request with its information with an existent rule. Rule class communicate with ***Portrange***, ***Timerange***, ***FileRule***, ***DateTime***, and ***Domainclass*** classes for matching of the request. After that related Boolean values (matched) considered for resolving the condition of request that whether the request will be forwarded or not by consideration of ***WebConnector*** class.

After all the activities with request be done, the Web Connector class listen related port for response of the request. Then, ***response*** class considers response for processing. Like request, response information considered by user class. User class gets related policy and match related rule for this response. Same process will be achieved for response; that is why in the figure it is not redrawn. After all response handled and will be forwarded to user properly or as an error message.

2.5 Syntax Specification

2.5.1 User and Admin Login History Syntax:

For ordinary user logs:

<username> <date> <time> <result> <source ip> <\n>

For administrator logs:

<username> <date> <time> <result> <source ip> <action> <\n>

Action is an integer code representing the action taken by the administrator.

Each action will be on a separate line. And each entity is separated by a whitespace.

File Names:

Actions are logged in a daily basis, more explicitly, there will be a separate log file for each day with the day being the file name. ex. User logs on the day 03/12/2005 will be saved in a file u031205.log, also admin files created on the same day will have a031205.log as its name.

2.5.2 Crawler Settings:

<start_time> <end_time>

The two items we have determined which will be included for sure in the settings file is the starting time and the ending time for the crawler module. We will add new items as may be necessary.

Crawler Logs:

<starting time> <number of websites visited> <number of websites classified> <finishing time>

<starting time> : The time crawling started.

<number of websites visited> : This is the number of websites visited during this crawling session.

<number of websites classified> : This is the number of websites classified during this crawling session. This number does not include websites that are not classified, for example when a website cannot be reached.

<finishing time> : This is the time crawling session finished.

2.5.3 Reporter Settings:

<sort by=[table column]> <show=[table]> <use charts=[pie | bar]>

<sort by=[table column]> : This item determines according to which criteria will the information coming from the DB will be sorted.

<show=[table]> : This item determines which tables from the database will be displayed to the administrator.

<use charts=[pie | bar]> : This item determines how the database information will be shown to the administrator. It will enable visual presentation to the administrator.

2.5.4 Controller Settings:

<timeout> <ip range> <max number of denials>

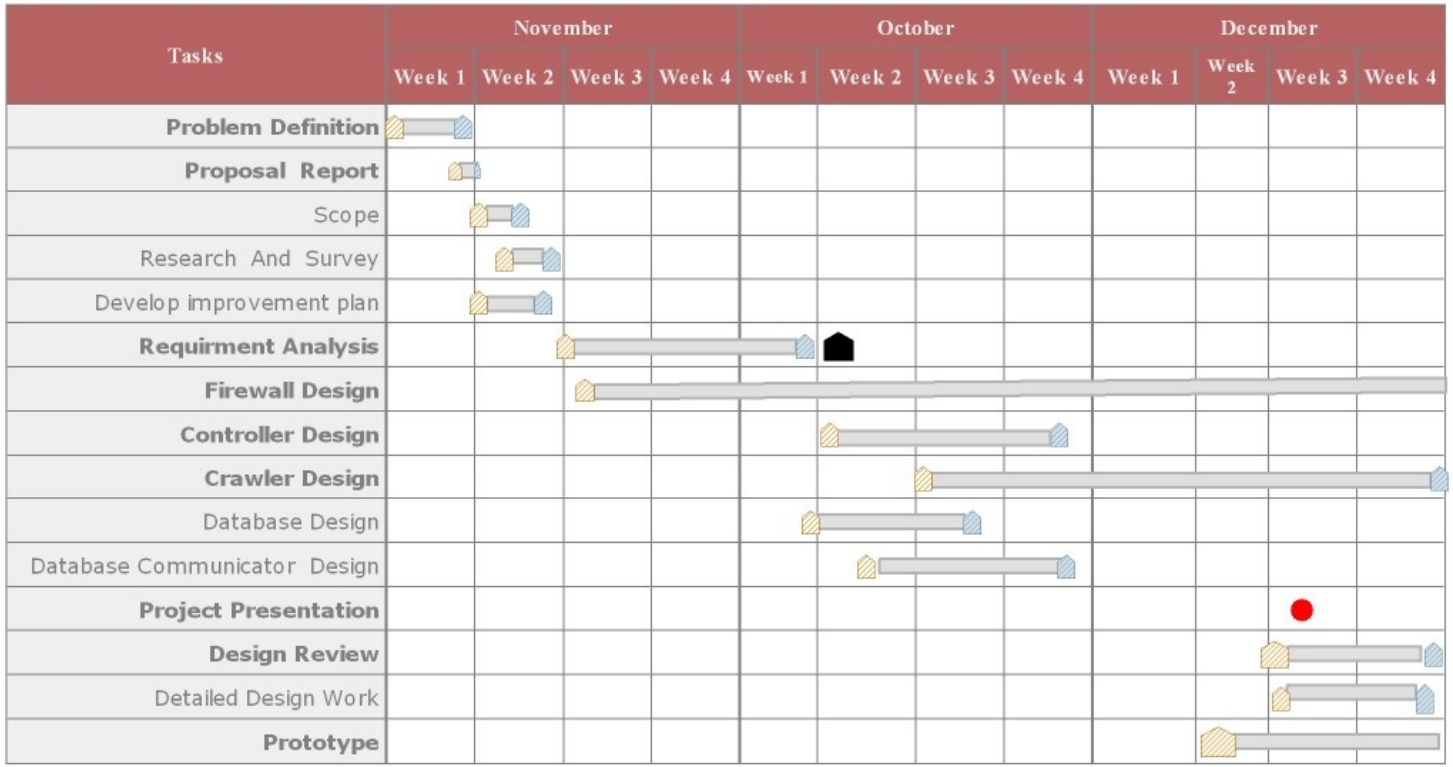
<timeout> : It is the maximum amount of time that administrator can stay idle until the system logs him out.

<ip range> : This is the allowed IP range that can connect to the controller module. It is added for extra security.

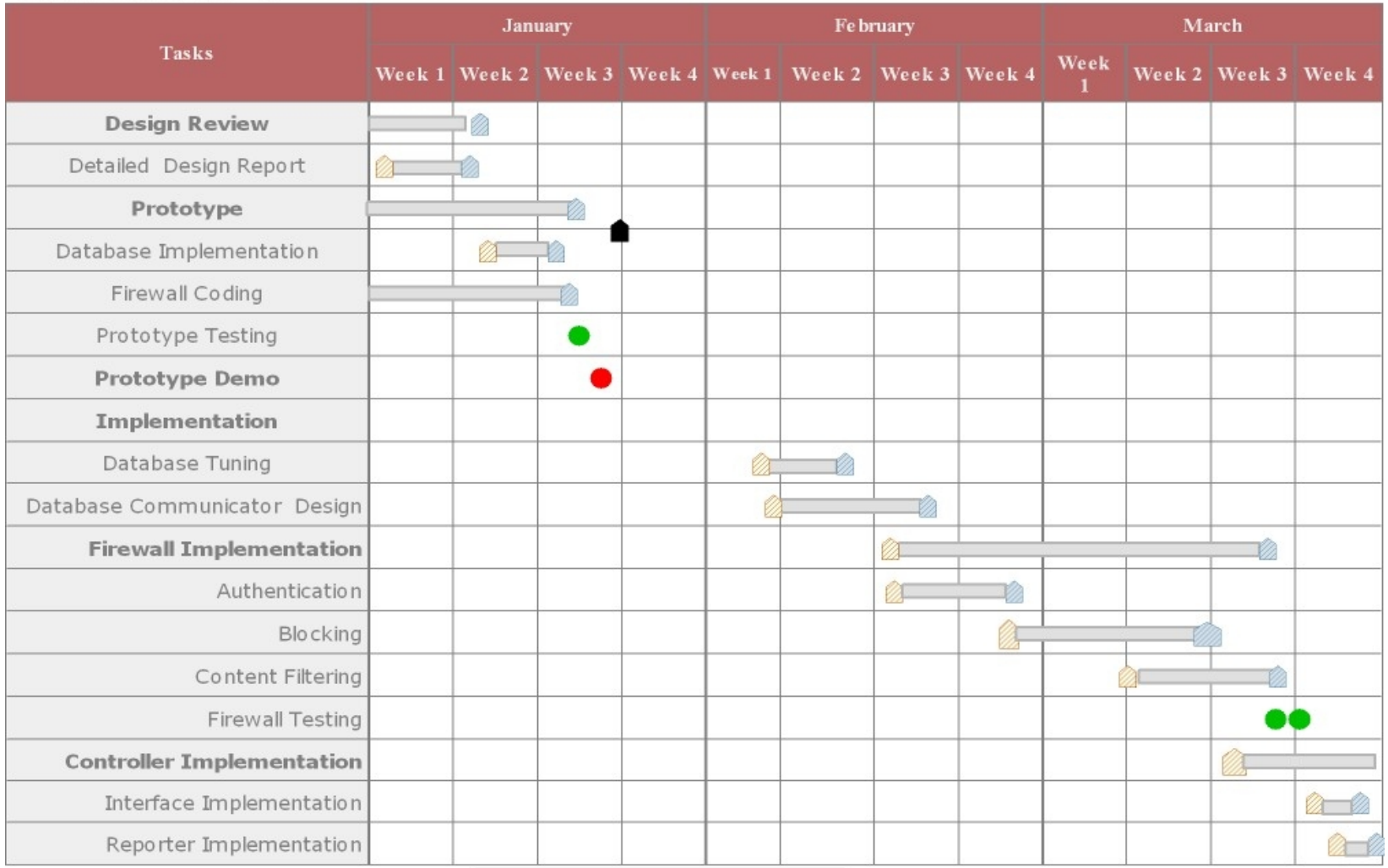
<max number of denials> : This is the maximum number of login failures from a particular IP. If the number of denials exceeds this number, the IP will be banned.

3 APPENDIX

GANTT CHART - 1



GANTT CHART - 2



GANTT CHART - 3