CEng 491

Software Requirements Specification

Secure Video Streaming Proxy Server

Sponsored by

Portakal Technologies

Team Prime'

Ant Ongun Kefeli

Doğan Poyraz

Yurdakul Göksu Orhun

Onur Cem Sırlı

December 12, 2010

Table of Contents

1. Introduction

1.1. Problem Definition

Video streaming and processing is used in many areas. For example, nowadays there are car surveillance systems in Ankara, which aims to determine the speed of cars and fine them if they exceed the allowed limit. However, the system is not able to do all the work. Now, it just can determine the car speed and detect speed limit violations and record the relevant frames. Then someone needs to check all these frames and fine them one by one by hand.

Software is being developed to automate this procedure by removing the human work completely out of the system. Instead of the police officers manually writing tickets, software will automatically detect and fine the violators. Note that, only in Ankara, cameras pinpoint around 18.000 traffic violations per day, however; only 2000 of them can be fined with manual labor. Therefore, automatizing the fining procedure will raise an important problem. Currently only 2.000 traffic violation footages are being stream from the central media server, however, when the update is complete the number of requested streams will raise to 18.000. The problem is that current infrastructure will not support the increased load and there does not exist a system which can distribute this load automatically between number of reverse proxy servers to maintain the functionality of the fining process.

Another example is face recognition software that the governments are using to track and detect the criminals. These systems use the footage that the distributed camera systems provide, like MOBESE or security camera networks in some malls. The software processes the video footage for faces of wanted criminals, when one is detected the software tries to find out more about the activities of the person, like who he/she met, where he/she is living etc. To dig out these information the software needs to process more footage to find the travel route and recent activities of the mentioned criminal. Clearly the whole process needs hours and hours of video footage which is divided into few seconded segments. As central media server cannot respond to this volume of video requests face recognition software output is vastly outdated due to delay of the required footage.

In both of these systems, there is an overwhelming work load on the server since these operations require dramatic amount of video streams per unit of time. And if multiple systems that accesses heavily to the media server run concurrently, load is multiplied and after a moderate amount of connections to the server, system cannot scale the and it may start to collapse causing serious issues on the other system that depends on the media server. Our system will mainly focus on the scalability issue by introducing reverse proxy servers that will share the load.

1.2. Purpose

This Software Requirements Specification document provides a complete description of the functions and specifications of the system that we will develop sponsored by Portakal Technologies. Rest of our development will take this document as the guideline. The intended audience is prospective software developers and all users.

1.3. Scope

Our Secure Video Streaming Proxy Server is a system that allows collection of streamed data from cameras, store the data retrieved from them and the share the data to authorized clients in a secure and scalable way.

The communication between cameras and server is done directly via Internet or intranet using camera IPs and other parameters required for connecting.

Clients will not directly connect to the media server to retrieve data, users will connect to the web-servers controlled by our system and web-servers will retrieve the data from media server and deliver it to the client. Details of this process will be given later.

1.4. Market and Technology Research

1.4.1. Literature and Market Research

As far as we researched the field, there are not many systems like the one we will produce. Usually, media streaming servers can operate only for a fixed amount of clients. Over the limit, they cannot provide data stream. There are some companies that offer solutions to this kind of problems. However, their solutions are not generic; they simply increase the bandwidth of their server in order to support more clients.

Our media streaming system will offer a more generic solution to this problem which is applicable to the most of the systems. Because, the work load on the media server will depend only to the number of web servers, not clients. Since a single web server will serve to lots of clients, work load on the main server will decline dramatically.

Eventually, we believe that the ideas that our product is introducing will be used widely among companies or people that provide media, especially video, streaming. Most probably municipalities in metropolitan cities will benefit most from this product because; recently some problems began to occur in their systems, such as freezing of the stream after first frame.

1.4.2. Technology Research

During the project, we plan to use C/C++, and some scripting languages (probably python) for coding. However, our language choices may change due to our efficiency, portability and security needs. Since there are many languages that allows us to do what we want, we will not have much problem if we decide to change our coding language.

Apart from the coding languages, we are planning to use Puppet software for Linux. Puppet is a system to centralize and standardize configuration and administration of your systems. These can be desktop systems, workstations, servers and so forth.

And for testing, we found some software for server testing. They are Apache benchmark, Grinder and Jmeter. When we began testing, we will decide on which one to use.

There are 2 main video formats that we will be dealing with: MPEG4 and MJPEG.

1.4.2.1. MJPEG

MJPEG is a class of video formats. In this format, each frame is separately compressed into a JPEG image. It is originally developed for multimedia PC applications but more advanced formats began to replace it.

Motion JPEG uses a lossy form of intraframe compression based on the discrete cosine transform (DCT). This mathematical operation converts each frame/field of the video source from the time domain into the frequency domain (or transform domain). Transform-domain is a convenient representation of the image.

1.4.2.2. MPEG-4

MPEG-4 is a collection of methods for compressing audio and visual digital data. It was introduced in 1998 and designated a standard for a group of audio and video coding formats. Uses of MPEG-4 include compression of AV data for web and CD distribution, voice and broadcast television applications.

Initially, MPEG-4 was aimed primarily at low bit-rate video communications; however, its scope as a multimedia coding standard was later expanded. MPEG-4 is efficient across a variety of bit-rates ranging from a few kilobits per second to tens of megabits per second.

1.5. Definitions and Abbreviations

Codec:	Coder-Decoder
DB:	Database
HTTP:	Hypertext Transfer Protocol
IP Camera/Source:	Internet Protocol Camera/Source
MMS:	Multimedia Stream
MS:	Media Server
SSL:	Secure Sockets Layer
SVSPS:	Secure Video Streaming Proxy Server
VPN:	Virtual Private Network

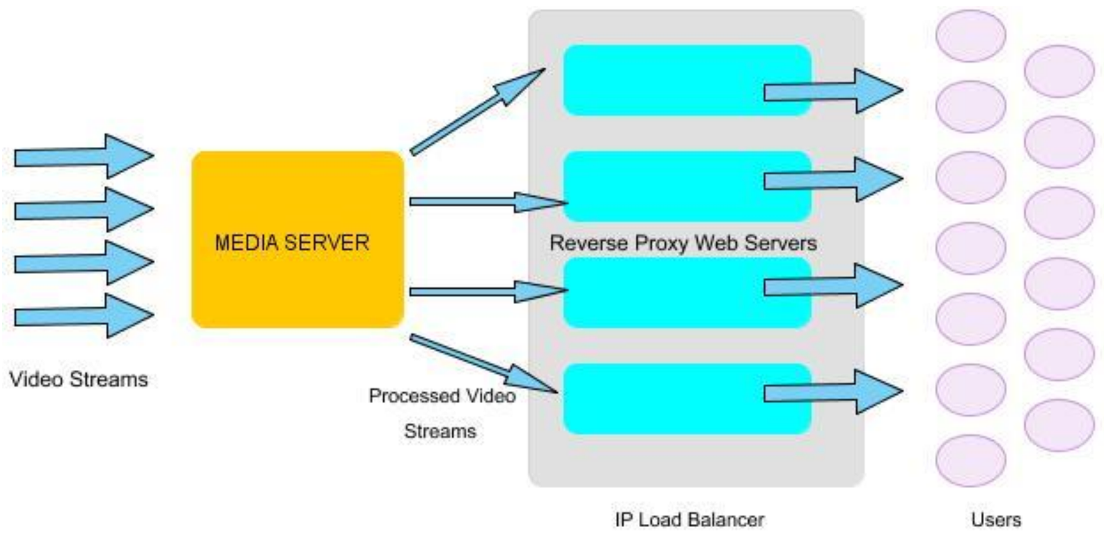
1.6. References

- An essay on W3C's design principles, view-source:http://www.w3.org/People/Bos/DesignGuide/introduction
- Authentication, Authorization, and Access Control, http://httpd.apache.org/docs/1.3/howto/auth.html
- IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)
- MJPEG, http://standards.jisc.ac.uk/catalogue/MJPEG.phtml
- MPEG-4 description, http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm
- Project Management Knowledge >> Rolling Wave Planning, http://project-management-knowledge.com/definitions/r/rolling-wave-planning/

1.7. Overview

Rest of the document contains six more chapters. Second chapter is Overall Description in which we will give a general description of the system. Then the Specific Requirements follows. In this part, more detailed functional and non-functional requirements will be explained. The next chapter is the Data Model and Description. This chapter will include information and data domain of the system and its organization. In the fifth chapter, we will describe behavior of the system. Then we will present our planning for the project and finish with conclusion.

2. Overall Description



Overall diagram of the system

2.1. Product Perspective

System consists of three main parts. The first one is the external streams which will feed into Media Server. Second one is the Media Server which will store the Stream taken from streams. Then, stream will be distributed to Web Servers accordingly which is the last part. This way, users will not directly connect to Media Server but via Web Servers. Number of Web Servers will depend on the active load. It may increase or decrease dynamically to balance the load. Clients will be handled by TCP/UDP Load Balancer.

2.2. Product Functions

Our system works with both human and non-human actors. Human actors are admins and users, non-human actors are system functions working as daemons. System daemons such as Load Balancer follow “Observer Pattern” . For instance, when state of a web server changes from light loaded to heavy loaded, a daemon is notified, namely Load Balancer, then the daemon handles the case without any need of human intervention.

2.2.1. Admin Functions

2.2.1.1. Login

Use Case:	Login
Primary Actor:	Admin
Goal in Context:	Accessing to the system
Scenario:	1. The system prompts the user for their username and password. 2. The admin enters username and password. 3. The system gets password registered to the user name. 4. The system verifies the password and sets the admin's authorization. 5. The admin is given access to the system to perform their job.

2.2.1.2. Management Selection

Use Case:	Management Selection
Primary Actor:	Admin
Goal in Context:	Accessing to sub-menu
Scenario:	1. Admin selects between stream management and user management.

2.2.1.3. Stream Management

Use Case:	Stream Management
Primary Actor:	Admin
Goal in Context:	Changing access rights of a stream
Scenario:	1. The admin enters stream management interface. 2. The admin selects the stream that he/she wants to manage. 3. The admin changes the access rights of the stream on the menu. 4. New access rights of the stream are set.

2.2.1.4. Add Stream

Use Case:	Add Stream
Primary Actor:	Admin
Goal in Context:	Adding new stream to Media Server
Scenario:	1. Admin clicks Add button in stream Management Window. 2. Window asks for the configuration settings of stream. 3. Admin enters the configuration settings. 4. Admin clicks OK button. 5. Stream is added to Stream List.

2.2.1.5. Remove Stream

Use Case:	Remove Stream
Primary Actor:	Admin
Goal in Context:	Removing stream from Media Server
Scenario:	1. Admin chooses a stream from Stream list. 2. Admin clicks Remove button. 3. Stream is removed from Stream List.

2.2.1.6. User Management

Use Case:	User Management
Primary Actor:	Admin
Goal in Context:	Changing access rights of a user
Scenario:	1. The admin enters user management interface. 2. The admin selects the user that he/she wants to manage from the user list. 3. The admin changes the access rights of the user. 4. New access rights of the user are set.

2.2.1.7. Add User

Use Case:	Add User
Primary Actor:	Admin
Goal in Context:	Adding new user to Relational Database
Scenario:	1. Admin clicks Add button in User Management Window. 2. Admin enters the user details. 3. Admin clicks OK button. 4. User is added to database.

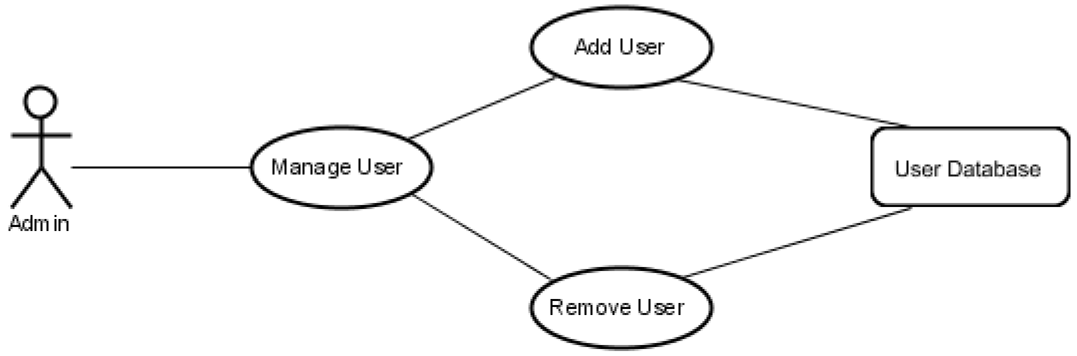
2.2.1.8. Remove User

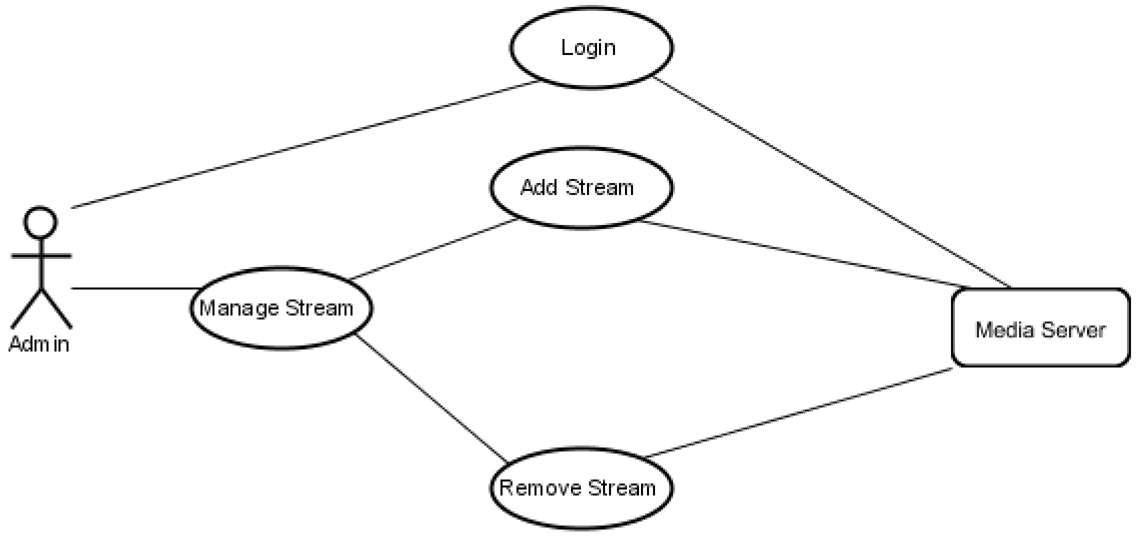
Use Case:	Remove User
Primary Actor:	Admin
Goal in Context:	Removing user from Relational Database
Scenario:	1. Admin chooses a user from User list. 2. Admin clicks Remove button. 3. User is removed from User List.

2.2.1.9. Use-Case Diagrams

In these diagrams, ‘Add Stream’ and ‘Remove Stream’ use-cases are specific operations of ‘Manage Stream’.

Similarly, ‘Add User’ and ‘Remove User’ use-cases are specific operations of ‘Manage User’.





2.2.2. User Functions

2.2.2.1. Log in

Use Case:	Login
Primary Actor:	User
Goal in Context:	Accessing to the system
Scenario:	1. The system prompts the user for their username and password. 2. The user enters username and password. 3. The system gets password registered to the username 4. The system verifies the password and sets the users authorization. 5. The user is given access to the system to perform their job.

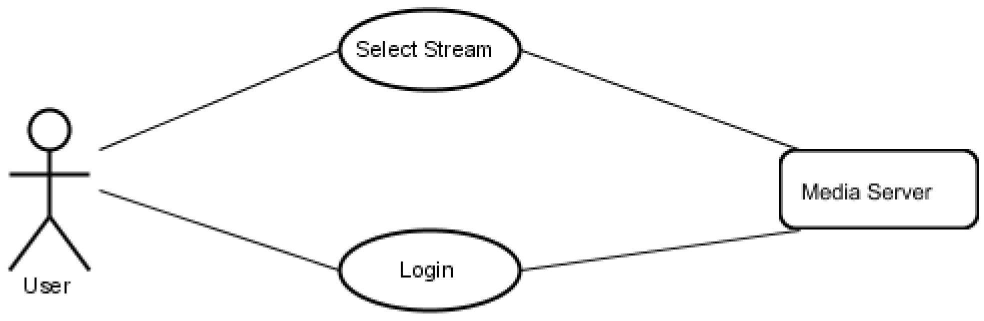
2.2.2.2. Stream Selection

Use Case:	Stream selection
Primary Actor:	User
Goal in Context:	Selecting a stream to access
Scenario:	1. User logs into the system. 2. After login screen, user is provided a list of streams that she/he is allowed to use. 3. User selects a stream and data flow starts.

2.2.2.3. Media Player Functions

Use Case:	Media player functions
Primary Actor:	User
Goal in Context:	Controlling playback
Scenario-1:	User presses play button to start data flow and play it.
Scenario-2:	User presses pause button to pause the stream at the current frame.
Scenario-3:	User adjusts volume level from the volume bar.

2.2.2.4. Use-Case Diagrams



2.2.3. System Functions

2.2.3.1. Load Check

Use Case:	Load Check
Primary Actor:	System Agent
Goal in Context:	Checking the load on web servers
Scenario:	1. Waits for a pre-determined time interval. 2. Checks the current load on each reverse proxy web server. 3. If a web-server serves more than a pre-determined number of clients, an idle web server is activated.

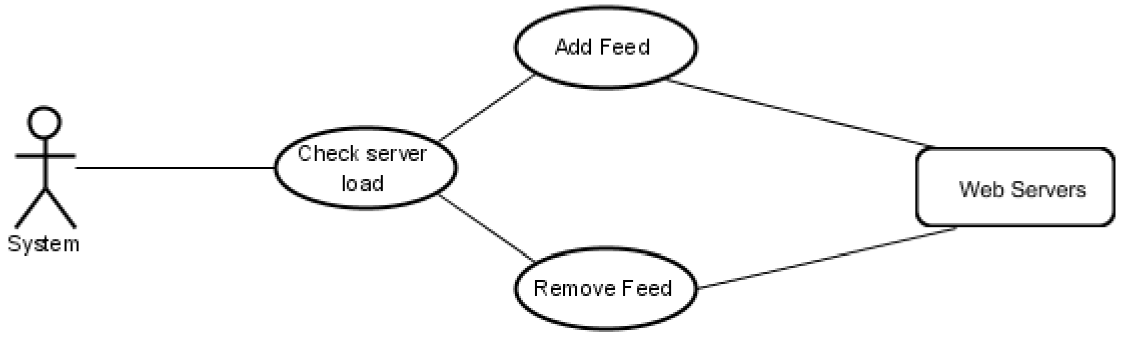
2.2.3.2. Add Feed

Use Case:	Add Feed
Primary Actor:	System Agent
Goal in Context:	Activating an idle web server
Scenario:	1. Agent decides that a new web server is needed. 2. An idle web server is activated and it starts to stream data to clients.

2.2.3.3. Remove Feed

Use Case:	Remove Feed
Primary Actor:	System Agent
Goal in Context:	Stopping a web server
Scenario:	1. Agent decides that less active web servers are enough. 2. One web server stops streaming and its clients are distributed among other active servers.

2.2.3.4. Use-Case Diagrams



2.3. Constraints, Assumptions and Dependencies

- Media streams are working fine.
- Streams and server have enough bandwidth.
- System has low latency.
- Video processing and networking libraries are available on the system.

3. Specific Requirements

3.1. Interface Requirements

In this part of the document the graphical user interface of the application is explained.

3.1.1. Admin Interfaces

3.1.1.1. Login Window

Admins should login to the system before using system management operations. This login window is the first thing the users face when they attempt to access the system. They shall be able to continue using the system upon entering a valid username and password.

3.1.1.2. Management Selection Window

Admins see management options which are Stream Management and User Management in this window.

3.1.1.3. Stream Management Window

This window lists available Stream feeds of the system. Admins can open dialogues for adding Streams to the system and removing them from the system using buttons on this window. Furthermore, stream access levels can be changed

3.1.1.4. Add Stream Window

This window prompts admins to enter configuration options of target stream. After entering correct settings stream will be added to available stream feeds list.

3.1.1.5. Remove Stream Window

This window prompts admins to approve removing target stream.

3.1.1.6. User Management Window

Admins should be able to list all users with their access levels in this window. There should be options to adjust users' access levels.

3.1.1.7. Add User Window

Admins should be able to add new users in this window.

3.1.1.8. Remove User Window

Admins should be able to confirm or cancel user removal operation.

3.1.2. User Interfaces

3.1.2.1. Login Window

Users should login to the system before listing the available Stream feeds and viewing them. This window is the same as user login window in 3.1.1.1.

3.1.2.2. Stream Selection Window

Users display the Stream selection window after successful login. They see the list of available Stream feeds that they can view in this window.

3.1.2.3. Media Player Window

Users see media player window when they want to view a Stream feed. This window has a media player with play/pause, sound volume buttons.

3.2. Functional Requirements

3.2.1. Admin Functions

3.2.1.1. Login

Requirement-1: The admin shall be able to login to the system through web interface.

3.2.1.2. Management Selection

Requirement-2: The admin shall be logged in to fulfill further operations.

3.2.1.3. Stream Management

Requirement-3: Admin shall be able to manipulate settings of the streams served by system.

3.2.1.4. Add Stream

Requirement-4: Admin shall be able to add a new stream to the list of the streams served by the system.

Requirement-5: Admin shall provide correct configuration settings for stream source to complete the add operation.

3.2.1.5. Remove Stream

Requirement-6: Admin shall be able to remove a previously registered stream from the list of streams served by the system.

3.2.1.6. User Management

Requirement-7: Admin shall be able to manipulate users that exist in the system.

3.2.1.7. Add User

Requirement-8: To add a new user to the system, the admin shell enter a unique username and specify a password.

3.2.1.8. Remove User

Requirement-9: Admin shall be able to remove a previously registered user.

3.2.2. User Functions

3.2.2.1. Stream Selection

Requirement-10: There shall be streams in the system.

Requirement-11: User shall be able to reach the streams which he/she has the privileges to access.

3.2.2.2. Media Player Functions

Requirement-12: User's shall be able to watch the video stream if his/her system meets the system requirements of the player.

3.2.3. System Functions

3.2.3.1. Load Check

Requirement-13: There shall be load on at least one of the reverse proxy web servers.

3.2.3.2. Add Feed

Requirement-14: System shall decide on adding feed.
Requirement-15: System shall be able to add feed when there is an idle web server.

3.2.3.3. Remove Feed

Requirement-16: System shall be able to remove a previously active feed from a web server.

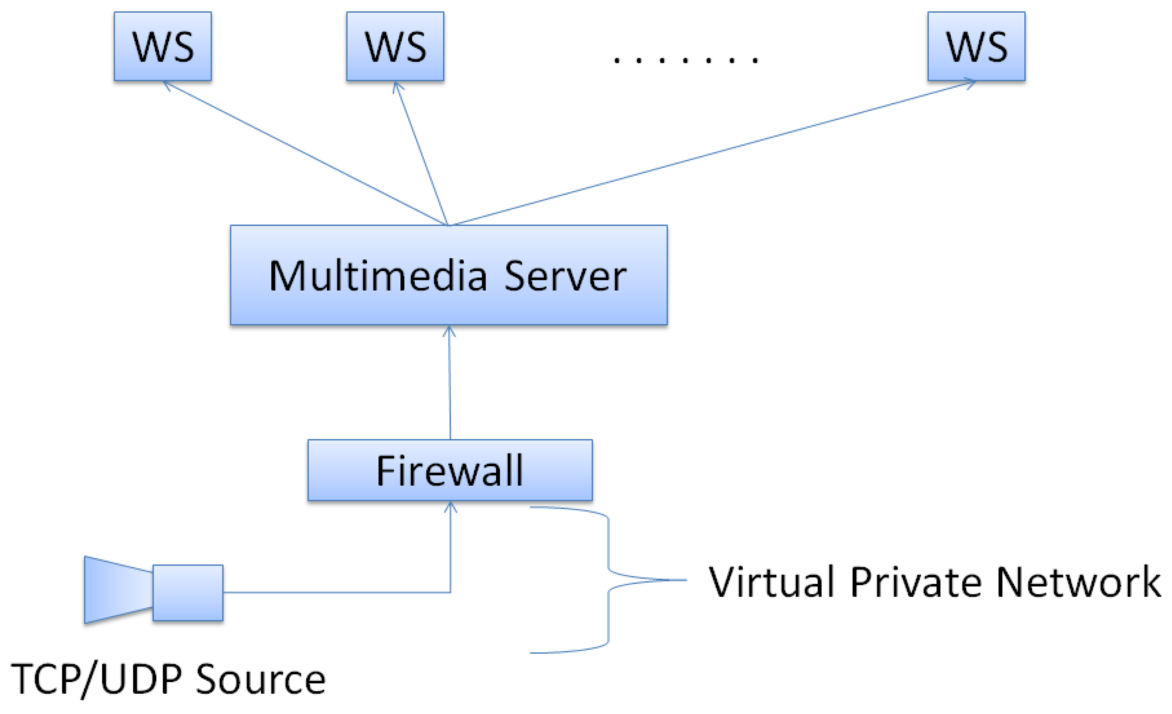
3.3. Non-Functional Requirements

3.3.1. Usability

There will be a web interface for users to connect system without another tool. We plan to follow W3C usability standards.

3.3.2. Security

In the system, streams and clients will have user groups that have pre-defined user rights. At the moment, we are considering only 2 user groups(public and private) but as the project progresses, probably new user groups will be needed and we will add new groups.



This part of the system is not in our scope, but in order to make system secure, there should be a system like this between media server and sources. A VPN should be active in order to prevent undesired connections to the server by unknown sources.

HTTP authentication protocol will be implemented in order to authenticate connections to the system. Users will authenticate themselves by supplying a username and password. The password will be verified by using HTTP digest access authentication in order to prevent undesired interception of the access credentials. All HTTP traffic will be encrypted using SSL encryption which will prevent theft, undesired access and/or corruption of data. We will utilize a firewall to secure the system from malicious attacks.

4. Data Model and Description

4.1. Data Description

Video File: Media Server records the video streams while serving them to users. As video files are relatively large, recording should occur after encoding the systems with appropriate codex. Furthermore, array of storage discs should be managed to ensure concurrent recording of multiple video streams. RAID 0 architecture can be employed on the discs for fast retrieval of video files.

User Information: Reaching the video streams are restricted to authorized users. Therefore, it is obligatory to store user information and authorization. Storage could be implemented in a file or relational database system, database system is chosen for performance concerns, as simple file system does not perform well in cases where user pool is relatively large.

Stream: Each video stream has its own unique ID in the SVSPS. This ID is used as an identifier at both stream processing and stream retrieval processes. Each stream source has several attributes related to it such as IP etc.

4.1.1. Data Objects

4.1.1.1. Video File

Saved in a file system. File path is included in the database as an attribute of stream object.

4.1.1.2. User

4.1.1.2.1. ID

Primary key for the User object. Consists of letters and digits. Length is restricted to a maximum of 20 characters.

4.1.1.2.2. Password

Password for the related ID. Consists of letters, digits and special characters. Length is restricted to a maximum of 20 characters.

4.1.1.2.3. IP

Last known IP address of the user. When a login process occurs, IP is updated. IP field can be used by admin to put location restriction on some streams, or altogether block an IP address. Previous accesses of users shall be logged.

4.1.1.2.4. Group

User group that defines someone's access rights to the system.

4.1.1.3. Stream

4.1.1.3.1. ID

Primary key for the stream object. Consists of 16 digits. Note that the digits are stored as characters.

4.1.1.3.2. IP

IP address of the stream source. This IP address can belong to an IP stream as well as a virtual or physical computer that is streaming a video.

4.1.1.3.3. Format

Format of the streamed video. MPEG4 is the usual format which is widely used by security streams; however other formats like MJPEG could be received.

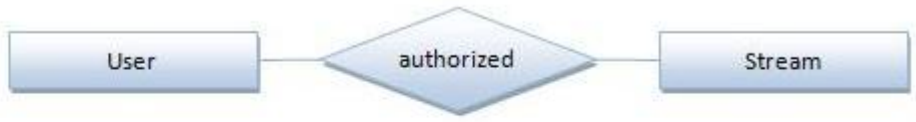
4.1.1.3.4. Security

Streaming source can be using a security protocol such as SSL.

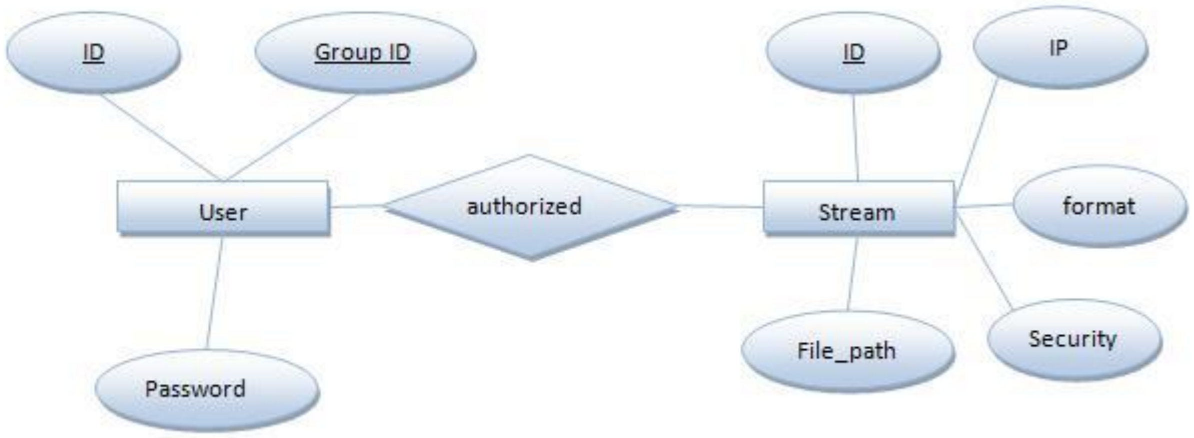
4.1.1.3.5. File Path

Media server records the streams concurrent to the streaming. Record file resides on the storage discs. File_Path attribute addresses the video file on the file system.

4.1.2. Relationships

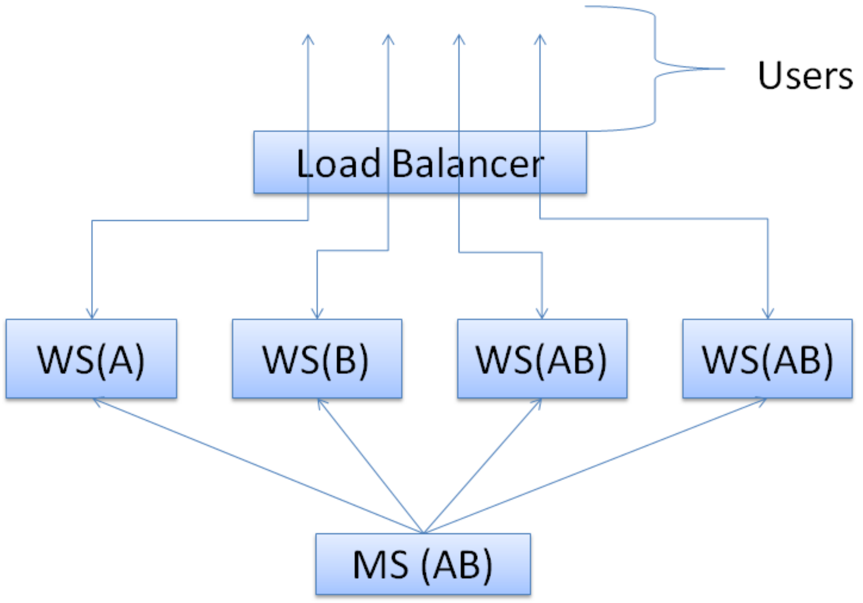


4.1.3. Complete Data Model



5. Behavioral Model and Description

In this chapter, we will describe the major events that occur in our system and describe how the system behaves on an event.



	MS	WS1	WS2	WS3	WS4	Users
A	3	20	-	18	9	47
B	3	-	32	10	7	49

These symbolic numbers represent outgoing streams fed per stream source on Media Server and Reverse Proxy Web Servers. MS has 3 feeds for each source and Web Servers access and scale up number of feeds. Therefore, having 6 feeds on MS and using 4 WSs we can provide feed to 96 users in this example.

5.1. Description of Software Behavior

5.1.1. Media source addition to the system

External sources can connect to media server in two ways. The desired case is automatic self-registration; process of this case is explained in 5.1.1.1. Second connection method is adding the stream source manually; process of this case is explained in 5.1.1.2. Both methods lead to indistinct processes after addition is complete.

5.1.1.1. Adding by automatic self-registration

Most of the IP stream sources run with embedded Linux operating systems. These have a configuration file on their file systems which contain configuration information required to perform self-registration to the media server. Username, password and IP address related to media server, streaming protocol of the source (HTTP, MMS etc.) and optional security protocols (SSL etc.) are parts of this information.

Streaming source sends a request to the target media server with the configuration settings. If the media server accepts the validity of the settings it sends an acknowledgement message back to the streaming source. Then, the source is given a unique ID and added to the stream list and its settings are saved to the database.

5.1.1.2. Adding manually

If an admin wants to add a new streaming source manually, the target camera configuration settings should be available. By using the media server interface, admin introduces the configuration settings of the streaming source. The settings should include all the related parameters discussed in the automatic registration process. However, in this case username, password and IP fields should reflect the streaming source.

Media server sends a request to the target streaming source. If source is available and responds with an acknowledgement message, it is added to the server list with a unique ID and its settings are saved to the database.

5.1.2. Removing media source from the system

In order to remove a media source from the system, admin selects the source from the stream management interface and marks it for removal. ID selection of cameras are handled by admin interface and admin is prompted for approval for removing that source. After admin approves the operation, the streaming source is removed from the server list and database with its unique ID.

5.1.3. Encoding and decoding media

IP cameras send the data usually in MPEG-4 or MJPEG format. When requested by a client, it is encoded again according to requesting client's decoder type. The data is also stored in the media server.

In addition to codec process, media server changes the resolution of the streamed video to a pre-determined value.

5.1.4. Activating a new web server and feeding it

IP Load Balancer periodically checks the number of users that a web server serves to. If the number is between pre-determined minimum and maximum values, IP Load Balancer waits for the next periodical check. If the number exceeds the maximum value a new web server is activated and load is redistributed among web servers.

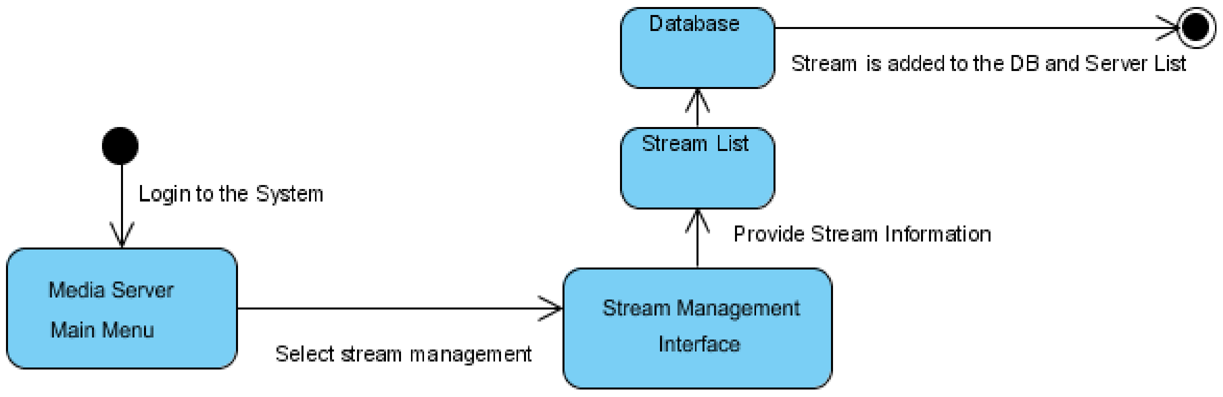
5.1.5. Deactivating a web server and stopping its feed

If Load Balancer detects web servers are used unnecessarily, that is, if the load can be distributed to less number of web servers according to pre-determined minimum and maximum values, one of the web servers is deactivated and load is redistributed among remaining web servers.

5.2. State Transition Diagrams

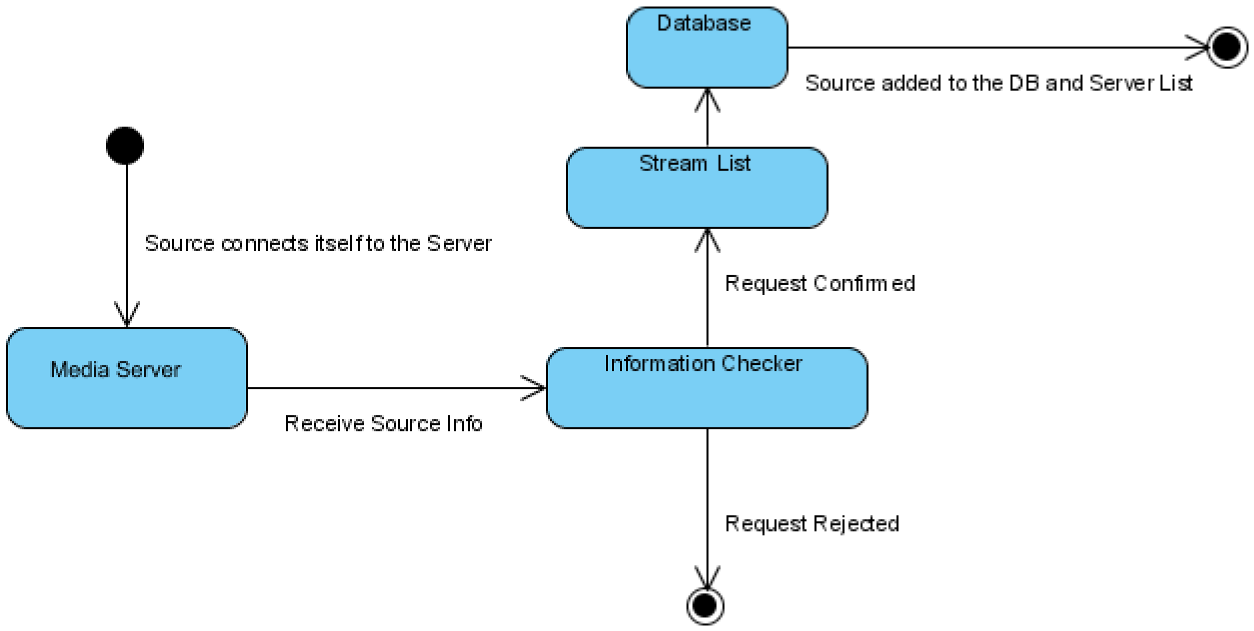
5.2.1. Media source addition to the system

5.2.1.1. Adding manually

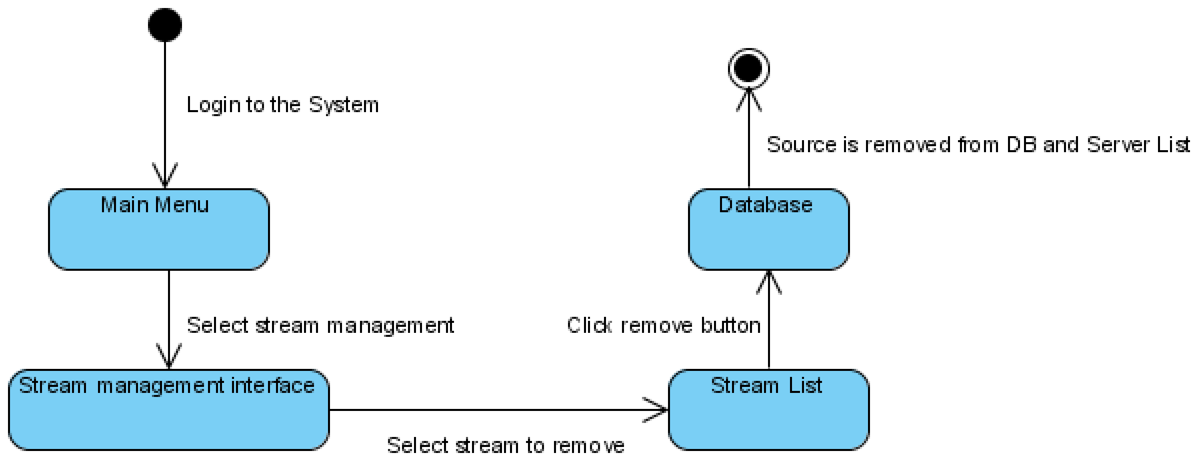


Database
Stream is added to the DB and Server List
Stream List
Login to the System
Provide Stream Information
Media Server Main Menu
Stream Management Interface
Select stream management

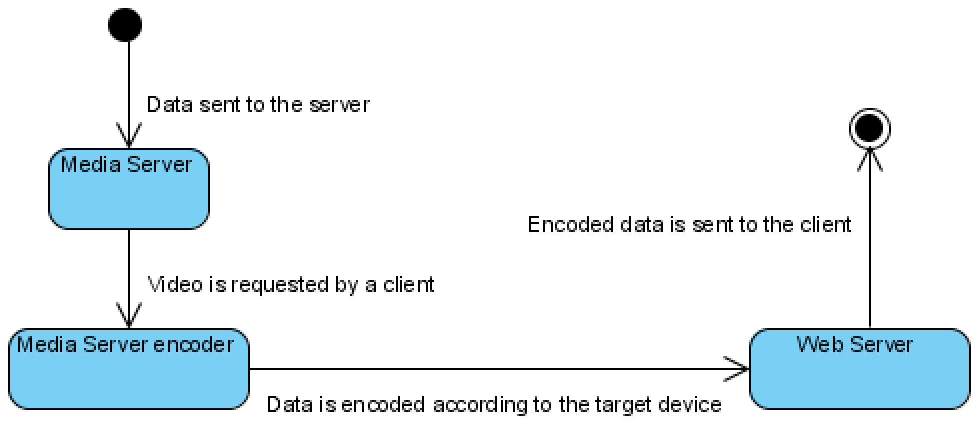
5.2.1.2. Adding by automatic self-registration



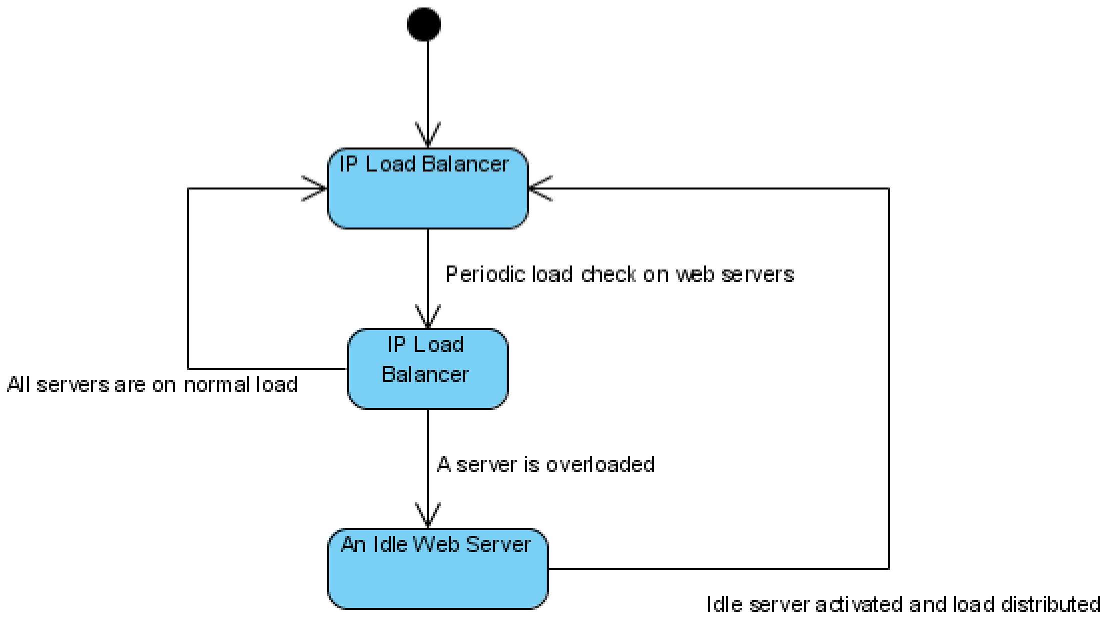
5.2.2. Removing media source from the system



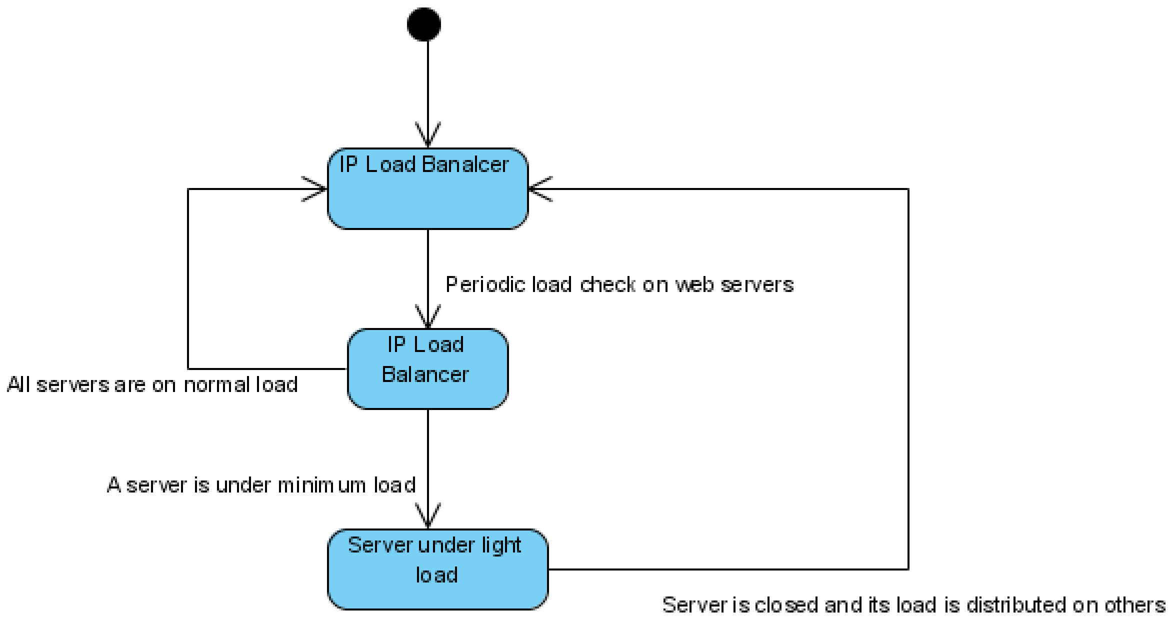
5.2.3. Encoding and decoding media


Data sent to the server
Media Server
Video is requested by a client
Media Server encoder
Data is encoded according to the target device
Web Server
Encoded data is sent to the client

5.2.4. Activating a new web server and feed it


IP Load Balancer
Periodic load check on web servers
IP Load Balancer
All servers are on normal load
A server is overloaded
An Idle Web Server
Idle server activated and load distributed

5.2.5. Shutting down a web server and stopping its feed

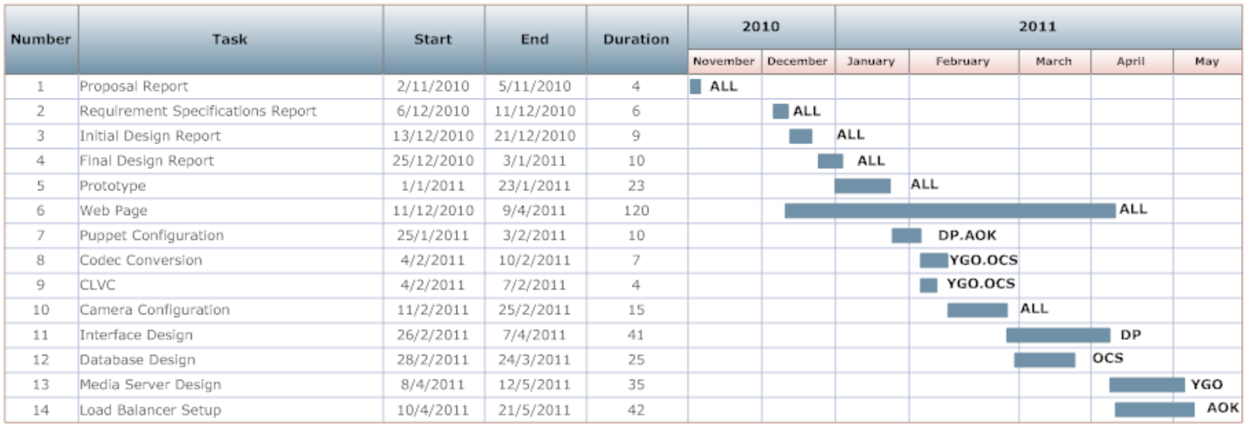

IP Load Banalcer
Periodic load check on web servers
IP Load Balancer
All servers are on normal load
A server is under minimum load
Server under light load
Server is closed and its load is distributed on others

6. Planning

6.1. Team Structure

Structure of a development team is a very important aspect of a project. So that everyone knows their role in the team and what is expected from them. We decided to have a democratic structure in the team as small team size and equal experience of the members made this structure very convenient. Since we make decisions together after discussing the issue, we do not need and a hierarchy in the team. For each part of the project, we assign everyone their roles in the relevant part. So, everyone knows his/her own role. If needed, we may exchange the roles in the team.

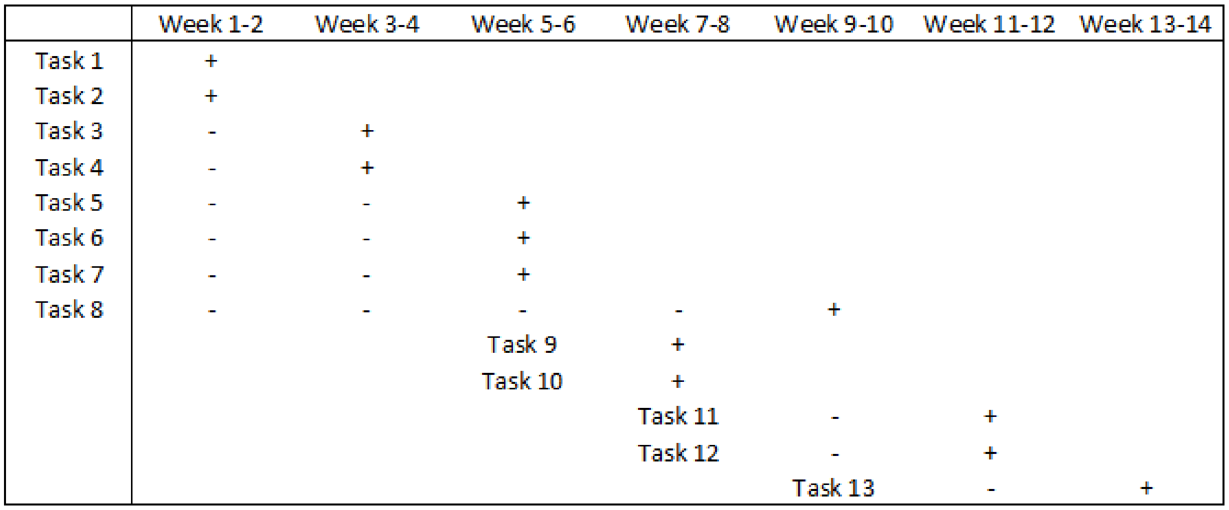
6.2. Basic Schedule



Number	Task	Start	End	Duration	2010		2011				
					November	December	January	February	March	April	May
1	Proposal Report	2/11/2010	5/11/2010	4	ALL						
2	Requirement Specifications Report	6/12/2010	11/12/2010	6		ALL					
3	Initial Design Report	13/12/2010	21/12/2010	9			ALL				
4	Final Design Report	25/12/2010	3/1/2011	10			ALL				
5	Prototype	1/1/2011	23/1/2011	23				ALL			
6	Web Page	11/12/2010	9/4/2011	120						ALL	
7	Puppet Configuration	25/1/2011	3/2/2011	10				DP.AOK			
8	Codec Conversion	4/2/2011	10/2/2011	7				YGO.OCS			
9	CLVC	4/2/2011	7/2/2011	4				YGO.OCS			
10	Camera Configuration	11/2/2011	25/2/2011	15					ALL		
11	Interface Design	26/2/2011	7/4/2011	41						DP	
12	Database Design	28/2/2011	24/3/2011	25						OCS	
13	Media Server Design	8/4/2011	12/5/2011	35							YGO
14	Load Balancer Setup	10/4/2011	21/5/2011	42							AOK

6.3. Process Model

We plan to use “Rolling Wave Planning” for further design because it suits better for our purposes. Due to dynamic nature of our project, we are not able to plan the whole process, we decided to plan in two weeks long periods. After two weeks has passed we will discuss what we have done and plan the next two weeks accordingly which may result in adding new objectives for our further progress in the project.



	Week 1-2	Week 3-4	Week 5-6	Week 7-8	Week 9-10	Week 11-12	Week 13-14
Task 1	+						
Task 2	+						
Task 3	-	+					
Task 4	-	+					
Task 5	-	-	+				
Task 6	-	-	+				
Task 7	-	-	+				
Task 8	-	-	-	-	+		
			Task 9	+			
			Task 10	+			
				Task 11	-	+	
				Task 12	-	+	
					Task 13	-	+

This is a sample schedule of rolling wave planning. In the beginning we start with 8 tasks and complete 2 of them in first period and 2 of them in the second period. However, 2 new tasks arise in our third period and we add them to our list while we work on previous tasks. This routine continues until the end of project.

7. Conclusion

This requirement document shows Prime's approach to secure video streaming. Problem is defined with sufficient examples to indicate its importance. User interactions with the system and intra-system relations are stated. Various planning models has been modified and combined to fit our team's working process. Furthermore, communication with the sponsor company has been developed via this report.