Live StreamBox

<u>Prepared by</u>

NC Farhat Zaman

PC Nimrah Ashfaq

NC Zafar Mahmood

Submitted to the Faculty of Computer Software Engineering

National University of Sciences and Technology, Islamabad in partial fulfillment for the requirements of a B.E Degree in Computer Software Engineering

August 2012

ABSTRACT

The wide-spread adoption of camera-embedded mobile devices along with the ubiquitous connection via WiFi or cellular networks enables people to visually report live events.

This thesis proposes architecture of a real-time video streaming service. Live StreamBox is a mobile application for android cell phones providing live video streaming from cell phones to a website on the internet using Real time Streaming server. By using this application, any user can stream live videos from their cell phones over the globe on the internet. It comprises Mobile application, Darwin Streaming server configuration & management, Networking & Session Handling through real time streaming protocols (RTSP, RTP), Web application and Database management.

Dedication

We dedicate our work to our parents and teachers.

Acknowledgement

We would like to thank Allah Almighty for His incessant blessings which have been bestowed upon us. We are also thankful to our families for their continuous moral support which makes us what we are.

Finally we are grateful to the faculty of Computer Science Department of the Military College of Signals, NUST.

Table of Contents

List of Tables 6

List of Figures 7

Chapter 1: Introduction 8

1.1 Introduction 8

1.2 Objectives 8

1.3 Deliverables 8

1.4 Technological Requirement 8

Chapter 2: Literature Review 10

Chapter 3: System Requirements Specification 13

3.1 System Feature 1: Video Streaming 13

3.2 System Feature 2: Session Handling by Streaming Server 14

3.3 System Feature 3: Web Service 16

Chapter 4: System Design Specifications 17

4.1 Architectural Style 17

4.2 System Description 19

4.3 Detailed Design 20

4.3 Presentation Model 28

4.4 UML Diagrams 29

4.5 Data Flow Diagram 56

4.6 Design Patterns 57

Chapter 5: System Implementation 62

5.1 Live Streaming 62

5.2Live StreamBox 62

Chapter 6: Testing and Results Analysis 84

6.1 Functional Testing (Black box) 84

6.2 White Box Testing (white box): 85

Chapter 7: Conclusion and Future Work 91

APPENDIX A: GLOSSARY 92

APPENDIX B: REFERENCES 93

List of Tables

Table 4.1 User Table **22**
Table 4.2 Account Table **22**
Table 4.3 Video Table **22**
Table 4.4 Comment Table **23**
Table 4.5 Rate Table **23**
Table 4.6 Tags Table **24**
Table 4.7 Access Log **24**
Table 4.8 Stereotype **28**
Table 4.9 Class Diagram Description **51**
Table 4.10 Authentication Class **52**
Table 4.11 LiveStream Class **53**
Table 4.12 CameraStreamer **53**
Table 4.13 DataAttributeSetter Class **54**
Table 4.14 RTPSockets Class **54**
Table 4.15 MediaStreamer Class **55**

List of Figures

Figure 4.1 Architectural Style.. 17
Figure 4.2 System Block Diagram 18
Figure 4..3 Entity Relationship Diagram.. 21
Figure 4.4 Table View and key Exchange.. 24
Figure 4.5 Content Model.. 25
Figure 4.6 Hypertext Model 26
Figure 4.7 Access Model 27
Figure 4.8 Presentational Model.. 28
Figure 4.9 Use Case Diagram.. 29
Figure 4.10 User Login Sequence Diagram 38
Figure 4.11 User SignUp Sequence Diagram 39
Figure 4.12 Start Video Stream Sequence Diagram....................................... 40
Figure 4.13 Capture Video Sequence Diagram.. 41
Figure 4.14 Streaming Sequence Diagram 41
Figure 4.15 Stop Video Stream Sequence Diagram.. 42
Figure 4.16 Visit Website Sequence Diagram 43
Figure 4.17 Browse Videos Sequence Diagram.. 44
Figure 4.18 System State Machine Model.. 45
Figure 4.19 Authentication State Machine Model 46
Figure 4.20 Capturing State Machine Model... 47
Figure 4.21 Streaming State Machine Model... 48
Figure 4.22 Deployment Diagram 49
Figure 4.23 Class Diagram.. 50
Figure 4.24 Data Flow Diagram 56
Figure 5.1 Activity launching Android 64
Figure 5.2 Webend... 74

Chapter 1: Introduction

1.1 Introduction

This section is written to specify the related work/background of Live StreamBox Project. Different streaming servers currently available to users are also included in the section. Moreover need of live streaming of videos using mobile phones over a low bandwidth network is also justified.

This document also explains the objectives of Live Streaming, deliverables of project (Live StreamBox), technical requirements, and a comprehensive project plan, explaining the work break down among the respective group members.

1.2 Objectives

The objectives of our project include:

a) To learn mobile application development/interfacing on Android OS
b) To Understand the configuration and architecture of streaming servers
c) To learn different techniques for video compression
d) To develop a web application for the web end users

1.3 Deliverables

Deliverables of the project are:

a) Mobile Application
b) Streaming Server
c) Web Application
d) Documentation/User Manual

1.4 Technological Requirement

In this project an Android mobile will be required for testing of mobile application and high processing machine (dual/quad core) for configuration of open source streaming server as available like Darwin streaming server, Wowza media server as well as for the web server (web server will be used for web application to access worldwide) like Apache, IIS, Apache Tomcat. Moreover internet facility will be required for streaming of the video to the server.

Chapter 2: Literature Review

The project mobile internet based live video streaming using mobile phones is very new idea in the area of mobile applications/services.

Previously there is little work done in the field of streaming video over low bandwidth networks and especially for hand devices i.e. mobiles. There are number of groups running this idea for high bandwidth networks i.e. DSL, WIFI. The need of live streaming is to facilitate users to interact with world live and share their activities over internet. Main concept of live streaming was surveillance, and social networking. There are IP cameras and computer integrated cameras working for this purpose which allow user to make account on streaming websites and then share live videos. The concept of surveillance is used for security of homes, offices and other places. People install tools live Stream Box and connect with online streaming servers and which enables them to watch their camera feeds anywhere in world by creating private account.

Moreover many commercial applications and products like QIK, Ustream.tv, LiveStream, Kyte, Bambuser, Next2Freinds, Molv, Stickam, LiveCLIQ, Seero, Yamgo, LiveCast exist in the industry which provide live video streaming through IP based high quality cameras. These solutions provide video input from a number of cameras which are connected to a streaming server or an end user application using high bandwidth wired or wireless network.

QIK is a “Smart Streaming” platform, which brings a lot of intelligence to the hard challenge of transmitting videos over mobile wireless networks. Qik offers the most flexible solution so that individuals can focus on capturing great moments knowing they’ll be able to:

a) Share with anyone — across handsets, platforms, carriers and networks
b) From wherever — reliable transmissions over 3G, 4G or WiFi networks
c) View whenever — show videos LIVE or anytime later
d) Capture Forever — automatically preserve precious and irreproducible moments

Ustream is the prominent live platform for broadcasting. At any time and anywhere, anyone can start enjoying with their family, friends or fans anytime, anywhere. Wide variety of events from movie premieres to high school sporting eventscan be viewed and broadcast by Ustream users.Ustream broadcasts include radical events, talk shows, concerts, sessions, movie openings, sporting events, interactive games.

Livestream is the leading source for real time event coverage. Itdesigns in a way that makes it simple for anyone to broadcast live to the web. To involve and grow their watchers on the web, mobile devices, and connected TVs, Event managers, content vendors, celebrities and artists around the world use Livestream's social broadcasting tools. Livestreamthrough its 'Zero Tolerance on Piracy' policy offers a distinctiveguarantee to protecting the rights of its users. The facility is offered as free (advertising-supported) or as a feature-rich, subscription for business.

Kyte as a web service provides free facility of streaming live videos from cells. It produces live streaming video, recorded video, and pictures from any compatible camera phone. One can easily sets one’s account page according to one’s preference and upload one’s videos to also upload your videos to MySpace or Bebo.

Bambuser allows one to stream live video from a well-matched cell phone at no cost. At a later time, all the recordings can be accessed easily. This can be because Bambuser streaming servers automatically record live steams. Videos can be made private by setting options so that only allowed users can watch the clips.

Next2Friends offers a pronouncedfree service for supported mobile phones to stream live videos for free. The options like auto-recording and private posting to third party video-sharing sites are not offered.

Molv is a new free mobile live video streaming. It supports many cell phones with more percentage of Symbian mobile phones. Automatic recording of live videos is supported. No geolocation and private streaming is available.

Stickam allows one to stream live video from a well-matched cell phone at no cost. With having an optimal broadcast resolution while live streaming, Stickam alsoprovide features of text-chat to facilitate users. Clips can be made private for selected users by setting the options accordingly. Features like geo-location or auto-posting to video-sharing sites is not provided.

LiveCLIQ is a free provider to stream live videos from the compatible mobile phone. Java technology supported cell phones are very famous for it. Position sharing through

GPS and streaming live video to a selected audience is provided. No features like live chat, auto-recording and auto-posting.

Seerois a free mobile live video streaming facility. It facilitates on-demand and live video with GPS option. Any mobile with GPS proficiencies works with Seero.

Yamgo is a free live video streaming service to mobile phones. By the addition of ads in the distributed videos to other platforms (desktop, mobile phones or other supported devices, Yamgo allows producers to make money from their video. The videos you stream are automatically recorded.

LiveCast is another free solution of live streaming from cell phones. Servers automatically record Live videos.

Our project Live StreamBox deals with streaming of videos over low bandwidth networks and only using android mobiles. And it will be very helpful for people who are moving because IP cameras are static and cannot move along with moving person and very limited area can be covered with DSL or WIFI networks. While mobiles device can easily be kept in pocket and any time anyone can start capturing video, which can be streamed to worldwide only using mobile internet facility over the mobile network which is also available easily. Goal of this project is same as of IP cameras but it will provide ease of portability.

Chapter 3: System Requirements Specification

The basic idea of video streaming is to split the video into parts, to encode and transmit these parts in succession to the streaming server, and enable the receiver to buffer, decode and playback the video as these parts are received, without having to wait for the entire video to be delivered.

3.1 System Feature 1: Video Streaming

3.1.1 Description and Priority

This will involve all video capturing, buffering, compression and delivering of this content to the streaming server from the mobile application. The application can be used for surveillance purposes; this feature of live video streaming is a very important feature of this project, as it turns your android phone in to a live video streaming device.

By using this feature the user can send live video stream from the camera on the android phone. This live streaming will then be provided by the web server over the internet. As the video is captured through the android phone, the cost decreases as no expensive cameras are required for the video streaming feature as the android phone provides its own good quality camera mounted in the phone.

Compression of video stream will be required with the suitable video encoder before streaming to the streaming server in order to cater the low bandwidth requirements in case of mobile network on which we have to stream our video. We will use some open source video encoding library or android built in media library video encoding features of H.264, 3GPP or MPEG-4 AVC, a best video compression standard.
In order to fulfill our very low bandwidth requirement of mobile internet, we will also set other options e.g. reducing file size, low resolution or less frame size, reducing frame rate.

3.1.2 Stimulus/Response Sequences

For starting video streaming application, the user will select “start video streaming” option from the mobile app interface and its connection with the streaming server will be establish via mobile internet. After the user will be connected, the app running on the android will start streaming live video by using the camera on the android phone, to the streaming server. The video stream will be compressed and buffered as per requirement before uploading to the streaming server. The end user will close the connection of live video stream by selecting “close video stream” option on the user interface which will stop uploading of video.

3.1.3 Functional Requirements

REQ-1: There will be a video streaming app which will be running on the android. This app will capture the video, compress, and buffer and stream this live video to the streaming server so that the users around the world can access the video through web.

REQ-2: There will be a user interface for this application which will provide different control options in the form of interface e.g. start video streaming, end video streaming.

REQ-3: Some streaming server will be used which will handle live streaming sessions capable of streaming our particular video format. Some open source streaming servers are Darwin streaming server, Wowza media server, Sirannon, Subsonic, Icecast etc.

REQ-4: Mobile internet will be used for providing cost effective live streaming functionality from mobile rather than limited range network like WIFI.

REQ-5: Suitable video codec will be chosen for streaming over the very low bandwidth provided by the mobile internet.

3.2 System Feature 2: Session Handling by Streaming Server

3.2.1 Description and Priority:

For live video streaming over the globe, we will setup an open source streaming server compatible with our application supported media formats., thus providing REAL time streaming, delivering files with the little help from the web server and handling much traffic loads. The server will be based on three main REAL time protocols:

a) Real-time transfer protocol (RTP)
b) Real-time streaming protocol (RTSP)
c) Real-time transport control protocol (RTCP)

These protocols support other major protocols like TCP or UDP that direct web traffic. These are used for launching and monitoring media sessions between end points. RTSP also describes and defines sequence of controls. These controls are helpful in monitoring multimedia playback like PLAY, SETUP, PAUSE, RECORD, and etc. We have to very carefully choose the streaming server capable of streaming the encoded video format. All the risks associated with session maintenance are handled by the streaming server protocols.

3.2.2 Stimulus/Response Sequences

When the user at the app end will select “start video steaming” from the mobile interface, our app will make an RTP connection with streaming server, thus start delivering compressed video stream packets to the streaming server. There the RTSP, RTP and RTCP protocols maintain the session between uploading and delivered ends thus providing simultaneous delivery and playback of video at the viewer end without having to wait for the entire video to be delivered. Video streaming will continue until the user select “stop video stream” option from the user interface.

3.2.3 Functional Requirements

REQ-1: REAL time protocols RTSP and RTP based streaming server will be deployed for streaming purposes in order to provide REAL time streaming of the content.

3.3 System Feature 3: Web Service

3.3.1 Description and Priority

We will setup a web server hosting a web site that will provide our users over the world with feature of viewing live video streams. Buffering will be done in order to achieve some time gap between the start of delivery and the beginning of playback at the client. The benefit of the buffering includes low delay before viewing starts rather than full file download.

3.3.2 Stimulus/Response Sequences

When any user will visit our site, and select the video link or embedded player within our web page, our web server hosting the page will request the required file from the streaming server by directing the user to the streaming server. The streaming server will then stream the video directly to the user computer, bypassing the web server using real time protocols. That stream will be received, decoded and played in the appropriate size window in the media player embedded in the web page.

3.3.3 Functional Requirements:

REQ-1: A web server will be setup for hosting our live streaming social website.
REQ-2: Suitable media player will be embedded in the web page capable of decoding and playing our streamed media format.
REQ-3: Video resolution will be set at the sender end.

Chapter4: System Design Specifications

4.1 Architectural Style

We have used Layered architectural style because

a. It is suitable for applications that involve distinct classes of services that can be organized hierarchically.
b. The structure of the system is organized hierarchically into layers.
c. Each layer provides service to the layer above it and serves as a client to the layer below.
d. The connectors are typically procedural calls.

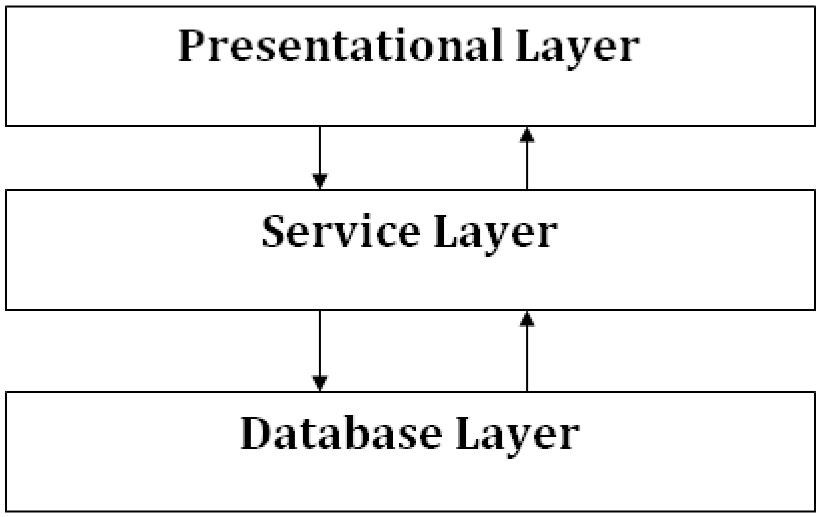


Figure 4.1 Architectural Style

General system architecture would look like as below

4.1.1 System Block Diagram

The complete system block diagram is shown in figure 2. We have used three-layer architecture which is as follows:

a. The first layer is the presentation layer which comprises of mobile and web interface.

b. The second layer is the service layer, comprises of streaming and web server.

c. The third is the storage (database) layer.

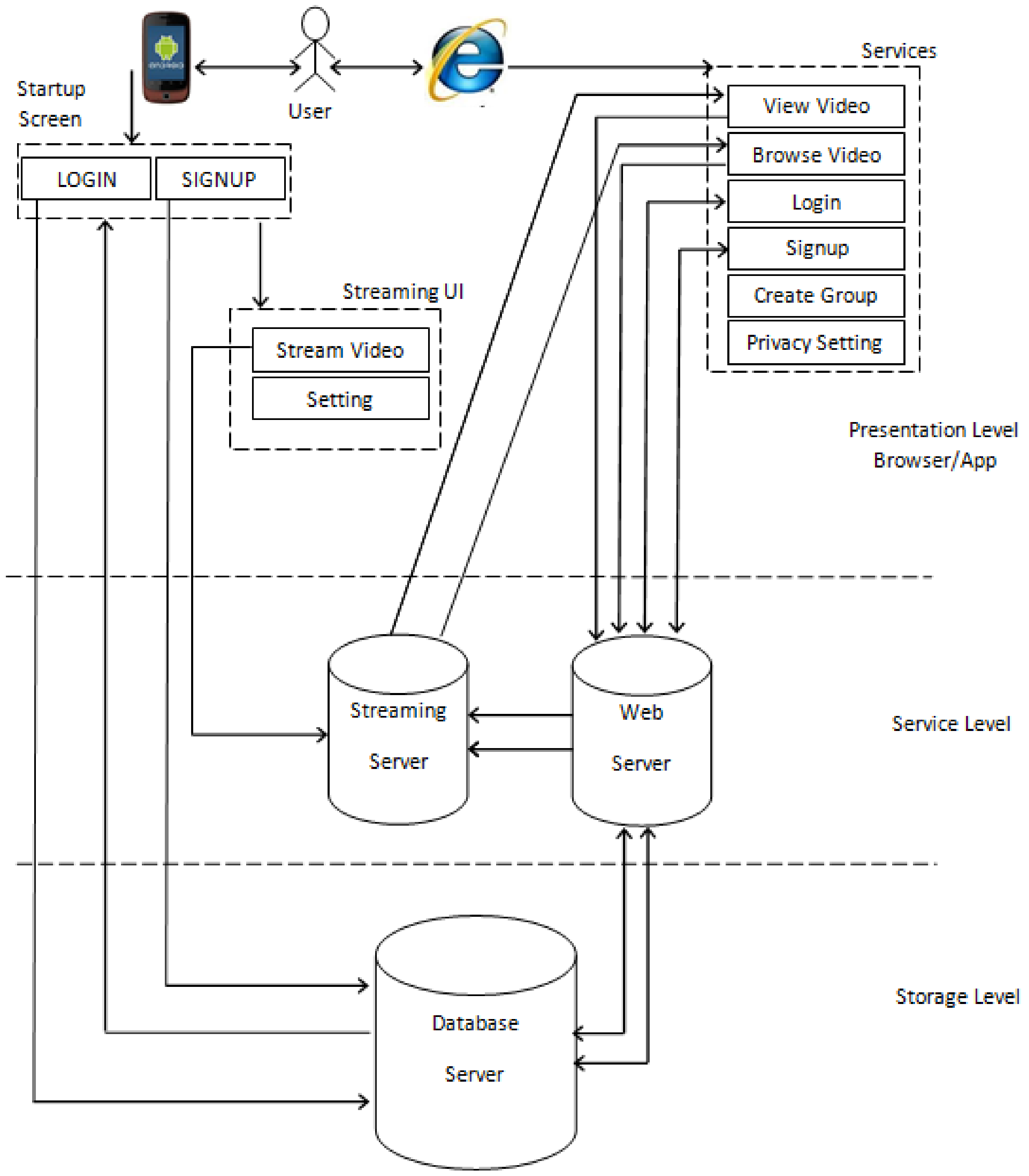


Figure 4.2 System Block Diagram

4.2 System Description

The description of system block diagram (Figure 3) is as follows:

Users:

- Mobile User
- Web User

Components:

- Presentation Level
- Server Level
- Storage Level

a. Mobile User

The user will use Android application through his mobile and he must be a registered user. Users can register themselves using the mobile application through the web. If already registered users can sign in and go live.

Login

Users will have entered his username and password for live video streaming. Username and password will be validated by the system from the database. On validation the user will be logged in and can stream or access the live videos.

Sign up

New users will have to register first to go live and stream the video. They will be asked for basic information for the creation of the account. Then this information will be sent to the database, if username or email is found duplicate in database then error message will be displayed and user can either provide a different username or email or can simply exit the registration. After successful registration user will be signed in to system to go live.

Stream Video

Authenticated users can stream their videos to streaming server via mobile internet i.e. EDGE.

b. Web User

Web application can be accessed from anywhere for viewing live videos without any authentication. User can also sign in/sign up on the web for making groups and editing profile and video information. Some services provided by web application.

View Video

Users can visit and view live streamed videos by android users without any authentication, request is forwarded to streaming server via web server for fast streaming.

Browse Video

Users can also search for videos by category as tagged by the android users i.e. education, sports, entertainment etc.

Sign Up

Users can register to our system to stream the live video and for the privacy settings. The data of user is saved to database for future use. Then this data will be sent to the database, if username or email is found duplicate in database then error message will be displayed and user can either provide a different username or email or can simply exit the registration. After successful registration user will be signed in to system to go live.

Sign In

Only registered users can sign in to system for editing of profiles

Create Groups

Service of creating group is used for making a social group to allow other people to share their videos; user can add some other users to a group and can share specific videos with them (optional requirement).

Privacy setting

Privacy setting is a service used for personal settings assigned to groups by users; 'Create Group' is a pre requirement for this service.

4.3 Detailed Design

The detailed design comprises of database/data design represented as ER diagrams/class diagram and appropriate UML diagrams representing the use case diagram, logical view, dynamic view and implementation view.

4.2.1 Database Design

Database design is the process of producing a detailed data model of a database. It contains all the needed logical and physical design choices and physical storage parameters needed to generate a design. It is used to determine the relationships between the different data elements.

4.2.1.1Entity Relationship Diagram

Live StreamBox system has to hold the data of users for authentication for both mobile end and web end. Database is maintained to achieve the required functionality.

The schema of database is given in figure .

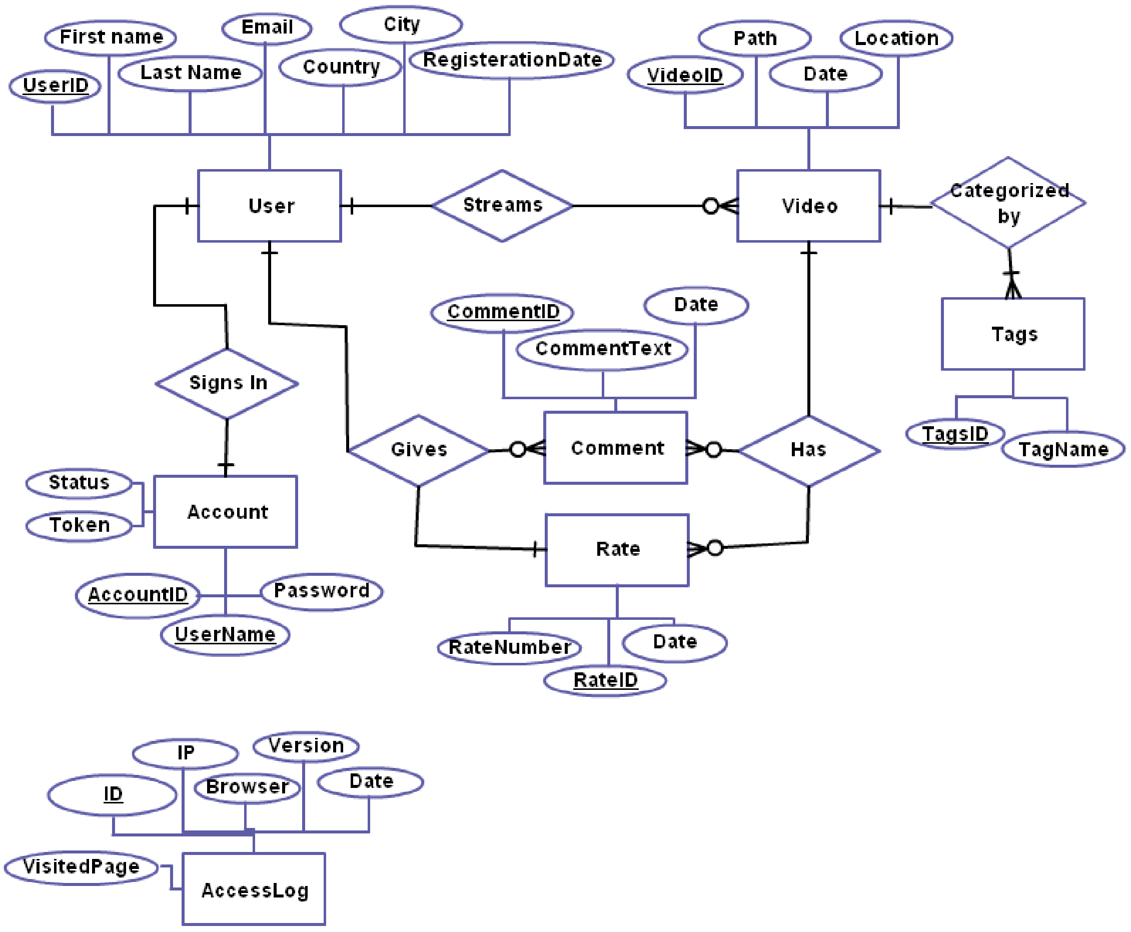


Figure 4..3 Entity Relationship Diagram

The entities and their attributes are given in the following section.

Table 4.1 User Table

User	
UserID	Unique identification of users in user table
FirstName	User’s first name
LastName	User’s Last name
Email	Email of the user will be stored, and programmatically will be unique to avoid data replication
City	City of the user
Country	Country of the user
RegisterationDate	Database will maintain record of the users like when they have registered themselves

Table 4.2 Account Table

Account	
AccountID	Unique identification of accounts of the users for sign in
UserName	UserName of the users to sign in, this is an unique attribute of the Account table
Password	Password is to identify exact user for sign in
Status	Status is used to disable or enable account of the users
Token	Token attribute is used to confirm email of user and to enable user account; programmatically a link will be mailed to the users when they confirm that the account will be activated.

Relationships

a. Account table will also hold the primary key of user to show relationship of one-to-one that each user has only one account information from where he is signing in.

Table 4.3 Video Table

Video	
VideoID	Unique identification of the video from Video Table
Path	the path or name of SDP file, for access in web application
Location	The attribute location is used to show the

	location of user from where he is streaming video
Date	Date and time of when the streaming started

Relationships

a. Video table will hold primary key of user to show the relationship of one-to-many/one-to-zero, that each video will have exactly one owner but one user can stream many videos or none.

Table 4.4 Comment Table

Comment	
CommentID	Unique identification of comments
CommentText	CommentText will save the comment of user
CommentDate	Comment date and time will be saved

Relationships

a. Comment table will hold primary key of User Table to show relationship of one-to-many/one-to-zero, that each comment is given by one user and each user can give many comments or may give no comments.
b. Comment table will also hold primary key of Video table to show relation of one-to-many/one-to-zero, that each comment is given to one video and each video can have many or zero comments.

Table 4.5 Rate Table

Rate	
RateID	RateID is unique identification of Rate Table
RateNumber	Rating will be in numbers that will be given by user
Date	This attribute holds date and time when video was rated

Relationships

a. Rate table will hold primary key of User Table to show relationship of one-to-one, that each rating is given by only one user and each user can only rate the video once.

b. Rate table will also hold primary key of Video table to show relation of one-to-many/one-to-zero, that one rate is given to one video and each video can have many or zero rate by the users.

Table 4.6 Tags Table

Tags	
TagID	Unique ID of Tag Table
TagName	Videos have to categorized by tags

Relationships

a. Tags table will contain primary key of Video Table to show relationship of one-to-one/one-to-many, that each video have one tag or can have many tags, while one tag is given to one video

Table 4.7 Access Log

AccessLog	
ID	Unique ID of table
VisitedPage	This attribute will save name of page accessed
IP	IP field will store client IP
Browser	This Field will contain the name of browser from request is generated
Version	Browser version is saved in this field
Date	Date and time of request generation

Table view and key exchange

The figure below shows relationships between tables in the form of primary and foreign key exchange.

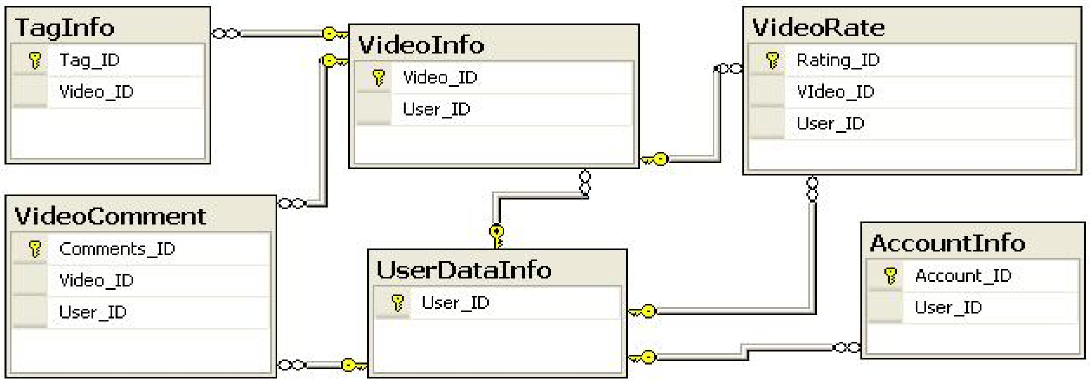


Figure 4.4 Table View and key Exchange

4.2.2 Content Model

Content model produces a model that comprises both the structural aspects (as class diagrams). The figure below shows the content model of the web application.

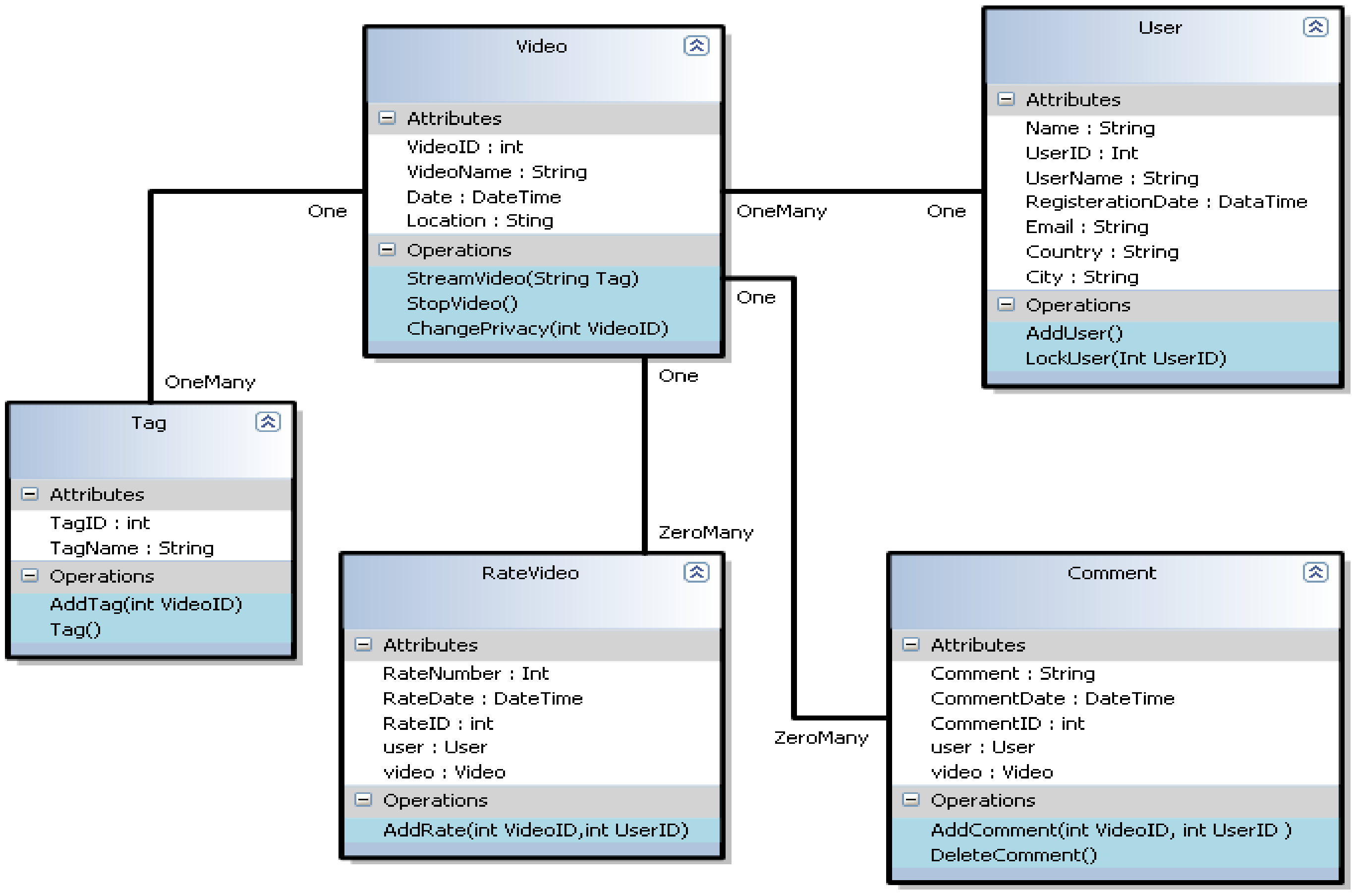


Figure 4.5 Content Model

4.2.3 Hypertext Structure Model

Hypertext Structure Model contains classes of the content model which can be visited by navigation. Hypertext structure model is specified as a view on the content model.

Figure below shows the hypertext structure model for the web application.

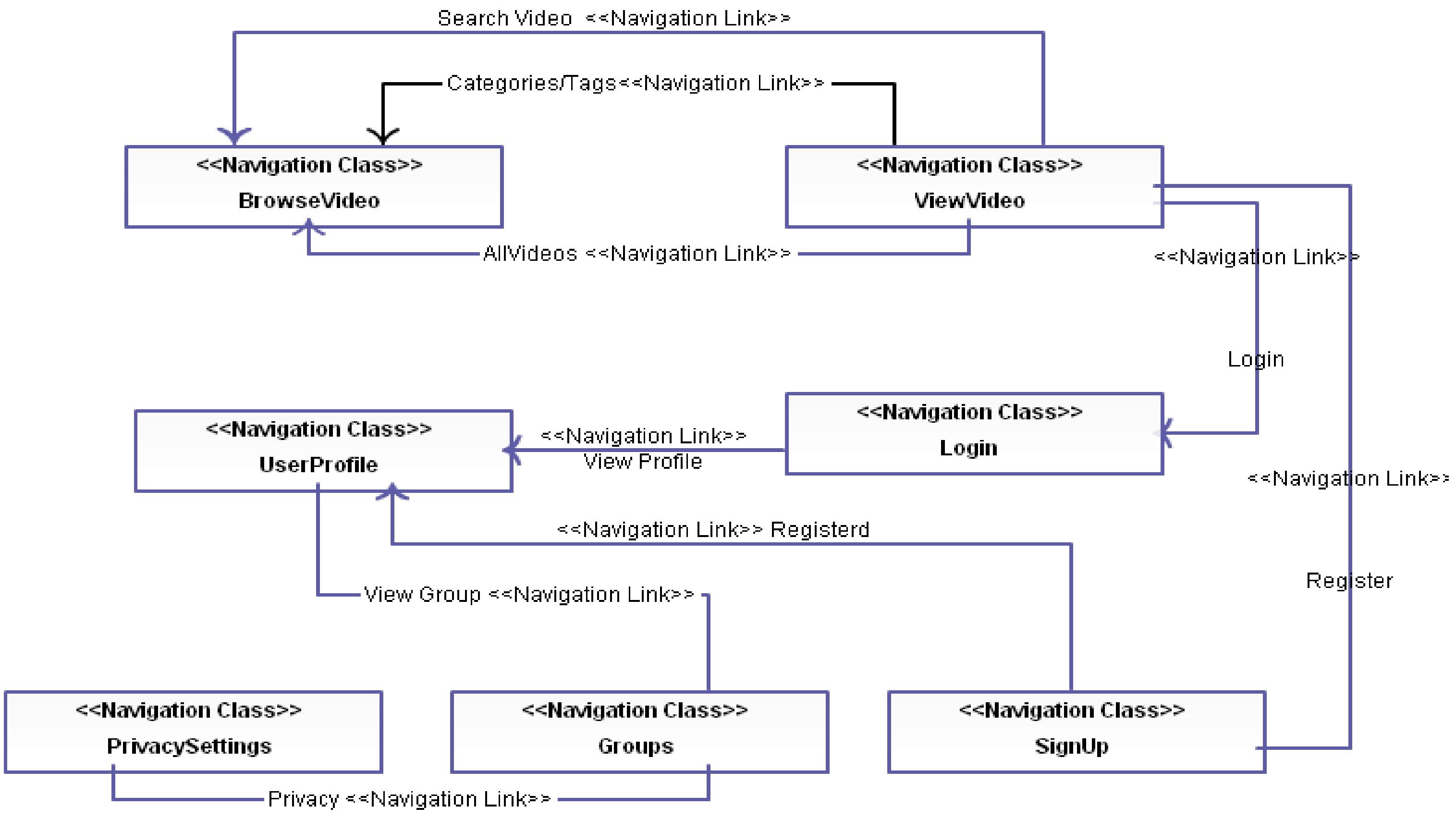


Figure 4.6 Hypertext Model

4.2.4 Access Model

Access structures refine the hypertext structure model to provide navigation and orientation aids. Access structures define various navigational structures such as index, guided tour, menu, home and landmark. The figure below shows the access model for the web application.

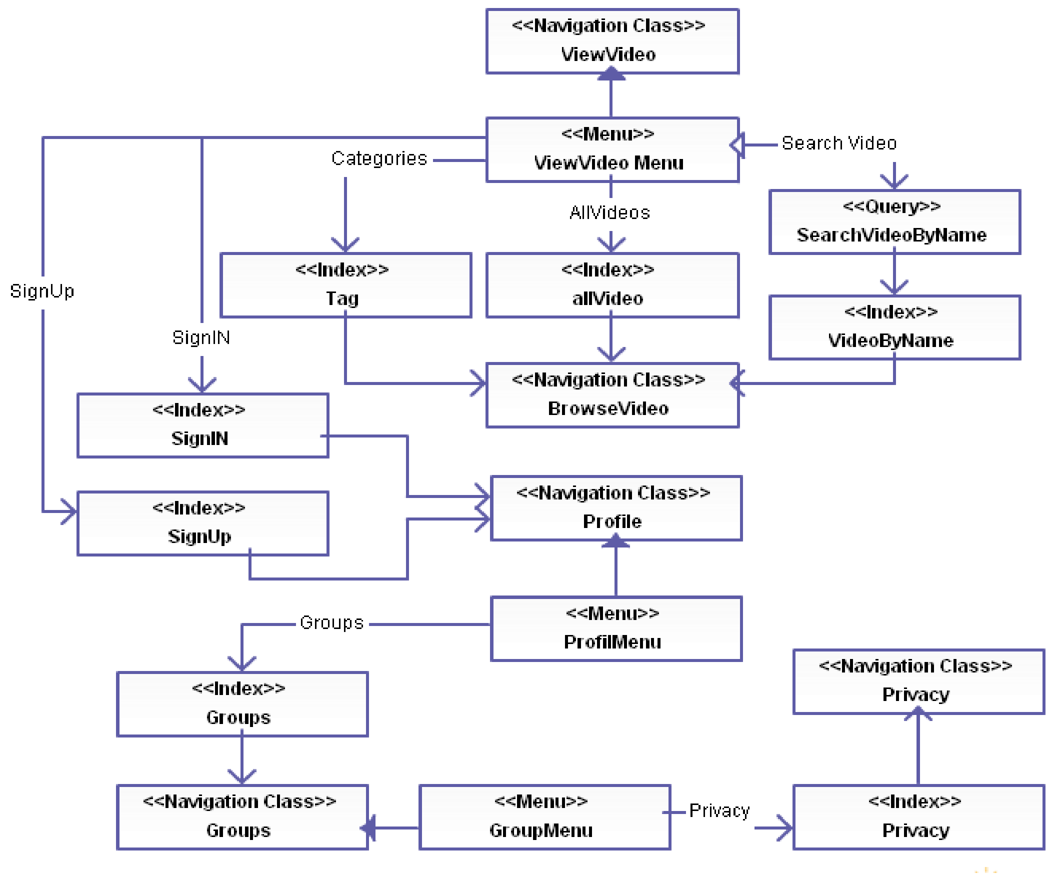


Figure 4.7 Access Model

4.3 Presentation Model

Presentation model deals with the user interface and look and feel of the Web application. Presentation page is a visualization unit presented as a page. The figure below shows the presentational model.

Table 4.8 Stereotype

Stereotype			
🗋	Presentation Page	⊡	Presentation Unit
—	Anchor	◙	Image
▷	Video	▷̲	Video List
=	Anchor List	⬬	Button
ab|	Input text	≋	Text

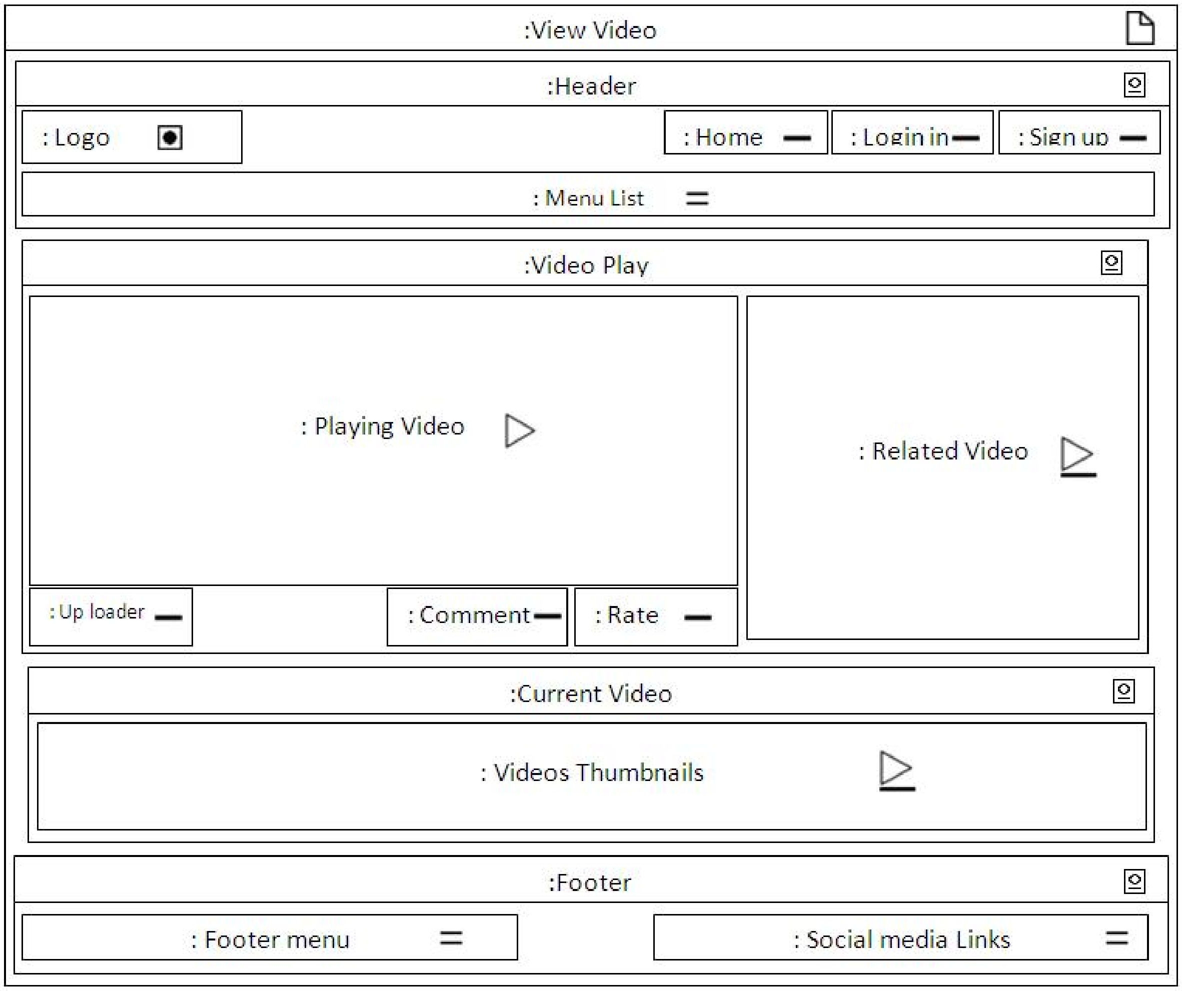


Figure 4.8 Presentational Model

4.4 UML Diagrams

UML diagrams represent two different views of a system model:

a. Static (or structural) view: It highlights the static structure of the system which includes objects, attributes, operations and relationships. Itcomprises class diagrams.

b. Dynamic (or behavioral) view: It highlights the dynamic behavior of the system. A change to the internal states of objects associations among objects is shown by this view. Itcontains sequence diagrams, activity diagrams and state machine diagrams.

4.4.1 Use case Diagram

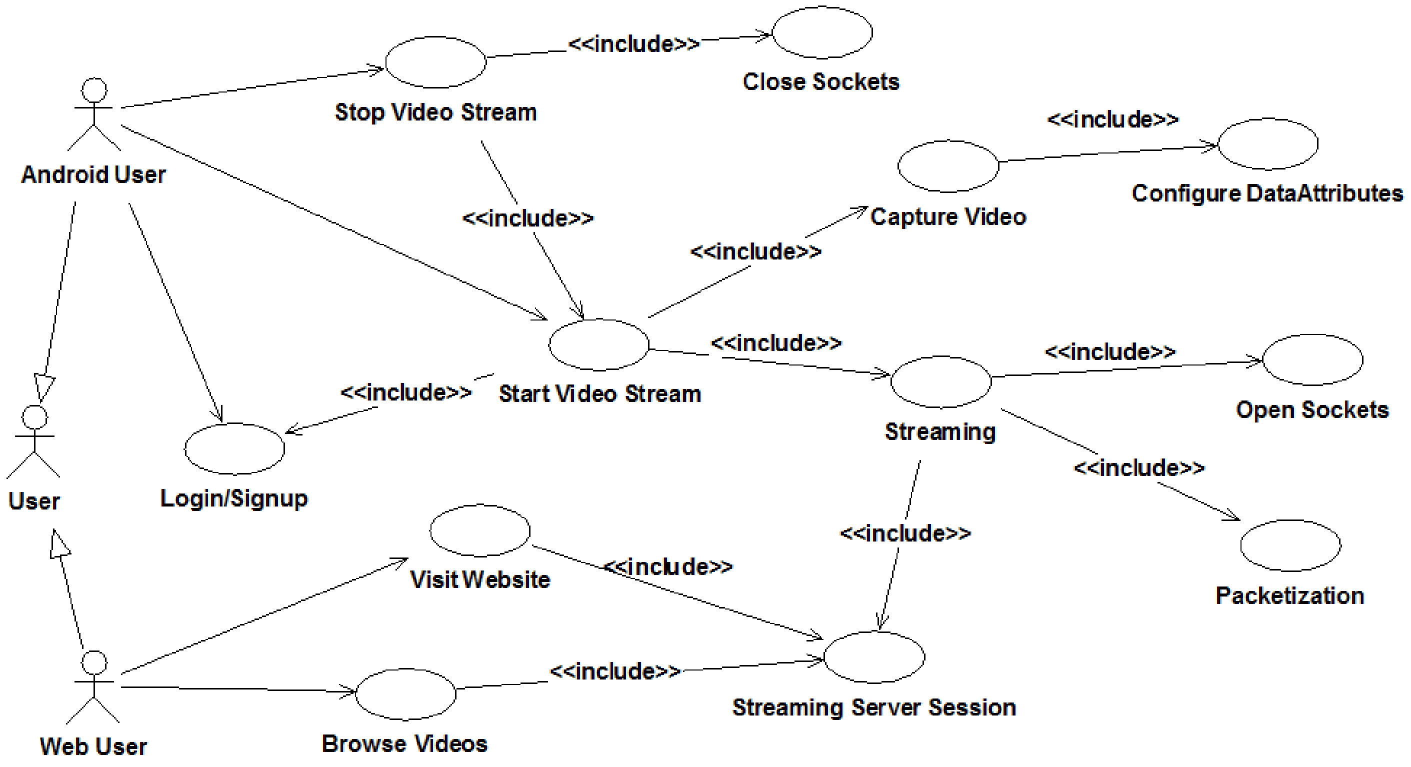


Figure 4.9 Use Case Diagram

4.4.1.1 Use case Specification

The use case specification for the use cases in figure 10 is elaborated in the section below.

a. LOGIN:

Brief Description:

This use case describes how a user logs into the Live StreamBox.

Actors:

Users can be android users and the web users.

Pre-conditions:

The system should be active.

Basic Flow of Events:

This use case starts when the actor wishes to log into the Live StreamBox.

a. The system requests that the actor enter his/her name and password.
b. The actor enters his/her name and password.
c. The system then validates the entered name and password and logs the actor into the system.

Alternative Flows:

a. Invalid Name/Password:

If, in the Basic Flow, the actor enters an invalid name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the Basic Flow or cancel the login, at which point the use case ends.

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the actor is now logged into the system.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

b. START VIDEO STREAMING:

Brief Description:

This use case describes how a user starts video streaming.

Actors:

Users can be android users.

Pre-conditions:

The system should be active and user must be logged onto the application.

Basic Flow of Events:

This use case starts when the actor wishes to log start the video stream.

a. The application launches and the video size, resolution, frame rates and the preview display are set.
b. The actor presses the START button on the application.
c. The application then creates a connection with the server and sets the audio and video attributes.
d. Then the application will create packets and the RTP sockets.
e. The application will now stream the video.

Alternative Flows:

a. Connection Failure:

If, in the Basic Flow, the connection with the server fails, an error message will be displayed. The actor can choose to either to wait the connection to be created or can exit the application.

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the actor is streaming the video.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

c. CAPTURE VIDEO:

Brief Description:

This use case describes how the application will set the required attributes for capturing a video.

Actors:

Users can be android users.

Pre-conditions:

The system should be active and user must be logged onto the application.

Basic Flow of Events:

This use case starts when the actor wishes to log start the video stream.

a. The application will be launched and will configure the data.

b. The video size, resolution, frame rates and the preview display are set.

Alternative Flows:

a. Video Size not supported:

If, in the Basic Flow, the user provides the video size which is not supported, an error message will be displayed. The actor can choose to either to provide other video size or to exit the application

b. Resolution not supported:

If, in the Basic Flow, the user provides the resolution which is not supported, an error message will be displayed. The actor can choose to either to provide other video size or to exit the application

c. Frame Rates not supported:

If, in the Basic Flow, the user provides the frame rates which is not supported, an error message will be displayed. The actor can choose to either to provide other video size or to exit the application

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the application will have configured the data attributes successfully.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

d. STREAMING:

Brief Description:

This use case describes how the application starts streaming the video.

Actors:

Users can be android users.

Pre-conditions:

The system should be active, user must be logged onto the application and the application should have configured the data.

Basic Flow of Events:

This use case starts when the actor presses the START button.

a. The application will open sockets i.e. RTP sockets.
b. RTP sockets will be opened for streaming video.
c. The application will then make packets. Packets will be of two types: video and audio packets.
d. Then these packets will start to stream.

Alternative Flows:

a. RTP Sockets not opened:

If, in the Basic Flow, the RTP sockets cannot be opened, an error message will be displayed. The actor can choose to either to wait for the sockets to open or to exit the application

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the application will have configured the data attributes successfully.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

e. STOP VIDEO STREAMING:

Brief Description:

This use case describes how a user stops the video stream.

Actors:

Users can be android users.

Pre-conditions:

The system should be streaming the video.

Basic Flow of Events:

This use case starts when the actor wishes to stop the video stream.

a. The actor presses the STOP button on the application.

b. The application destroys the surface and the session expires.
c. The application then closes the connection with server.
d. It then closes all the RTP sockets.
e. The application will go back to the initial screen.

Alternative Flows:

N/A

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the actor has stopped the video stream.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

f. VISIT WEBSITE

Brief Description:

This use case describes how a user views the live stream on the web.

Actors:

User can be any one who can use the web.

Pre-conditions:

The system should be active.

Basic Flow of Events:

This use case starts when the actor launches the web application.

a. The application will have a session already created with the streaming server.
b. The streaming server then sends the stream to the application.
c. Then the application will display the video stream coming from the streaming server.

Alternative Flows:

N/A

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the actor is viewing the video stream.

b. Failure Condition:

If not, the system state is unchanged.

c. Special Requirements:

N/A

g. BROWSE VIDEOS:

Brief Description:

This use case describes how a browse different live video streams.

Actors:

User can be any one who can use the web.

Pre-conditions:

The system should be active.

Basic Flow of Events:

This use case starts when the actor launches the web application.

a. The application will allow the user to browse for other live video streams.
b. On selection of a particular video, the application will connect to the streaming server.
c. Then the streaming server will send the stream to the web application so that user can view the selected video.

Alternative Flows:

a. Connection with the Streaming Server fails:

If, in the Basic Flow, the connection with the streaming server fails, an error message will be displayed. The actor can choose to either to wait for the connection to be created or to select another video.

Key Scenarios:

N/A

Post-conditions:

a. Successful Condition:

If the use case was successful, the actor is viewing the video stream.

b. Failure Condition:

If not, the system state is unchanged.

Special Requirements:

N/A

4.4.2 Sequence Diagrams of key use cases

a. Use Case Login/SignUp

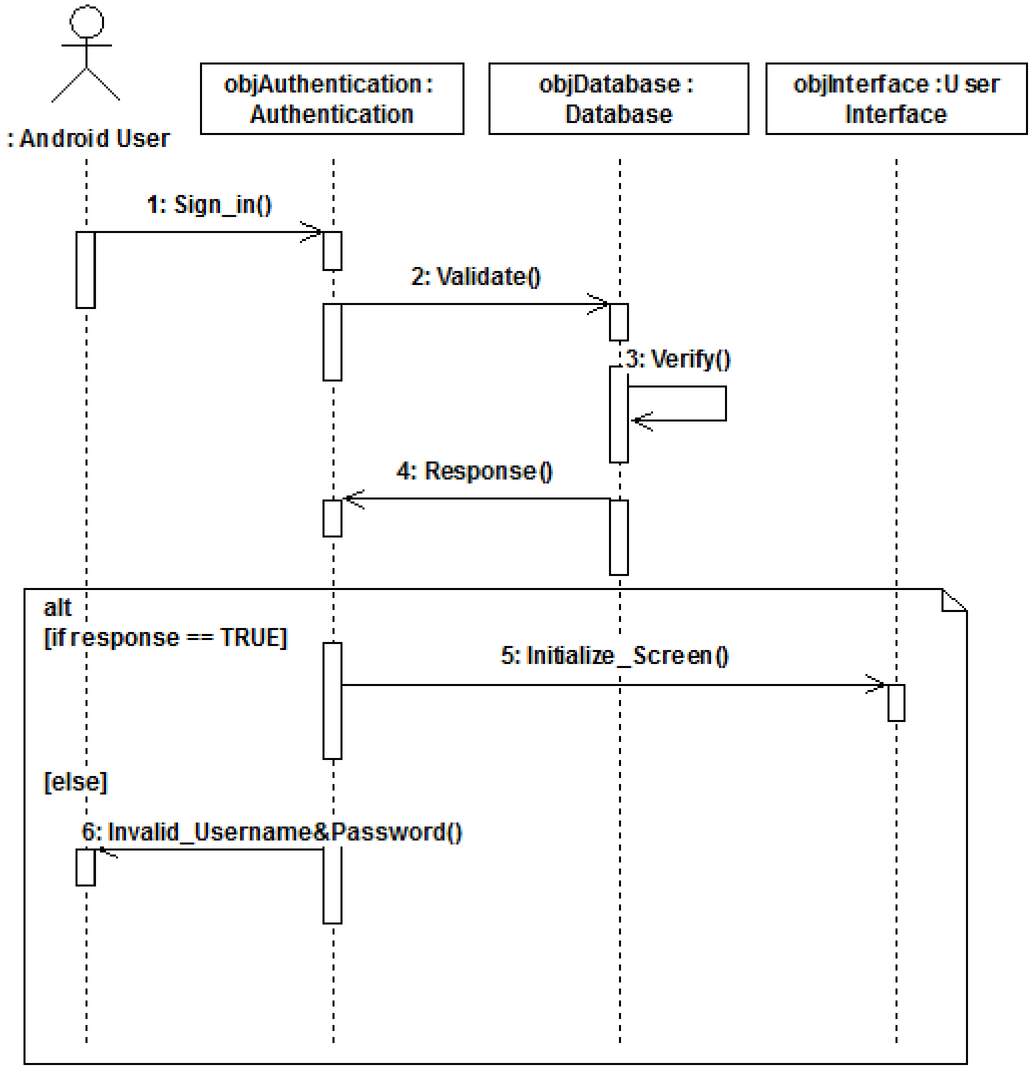


Figure 4.10 User Login Sequence Diagram

The figure below shows the sequence diagram of how a user sign up to the application. When a user is successfully registered, he goes to the Login screen.

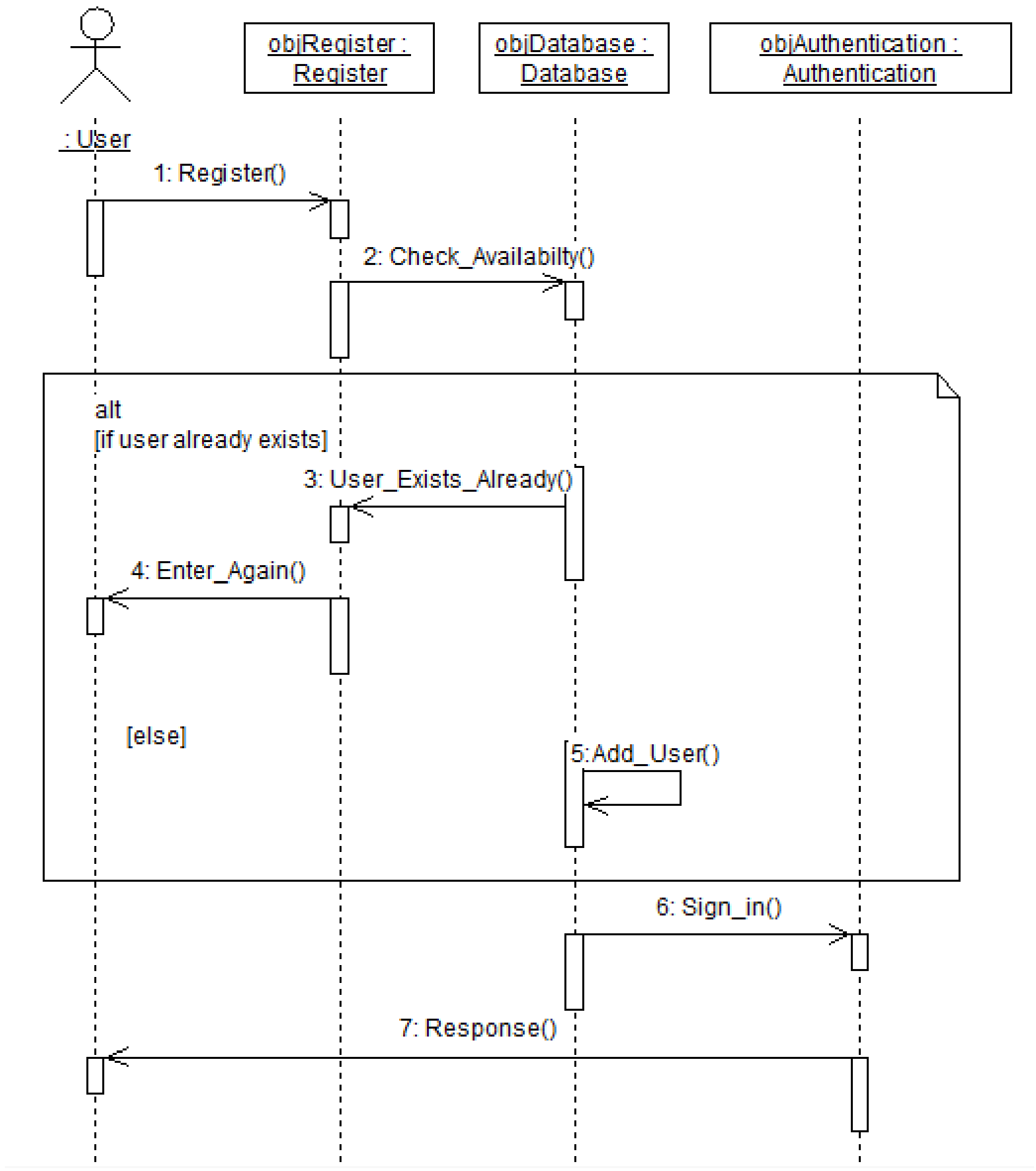
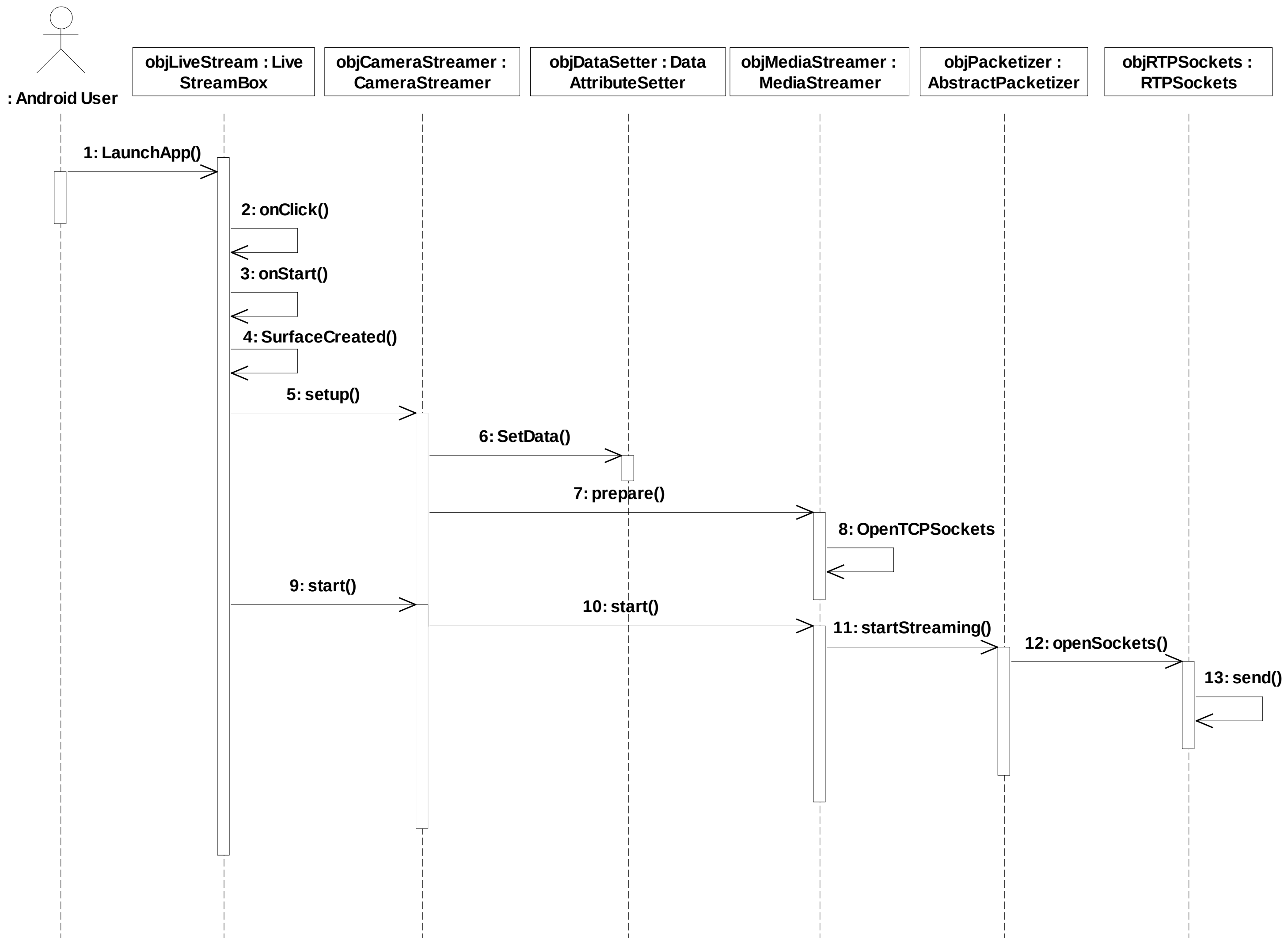


Figure 4.11 User SignUp Sequence Diagram

b. Use Case Start Video Stream

The sequence diagram for the use case Start Video Stream describes how a user starts video streaming. The figure below shows the sequence diagram for the start video stream use case



: Android User
objLiveStream : Live StreamBox
objCameraStreamer : CameraStreamer
objDataSetter : Data AttributeSetter
objMediaStreamer : MediaStreamer
objPacketizer : AbstractPacketizer
objRTPSockets : RTPSockets
1: LaunchApp()
2: onClick()
3: onStart()
4: SurfaceCreated()
5: setup()
6: SetData()
7: prepare()
8: OpenTCPSockets
9: start()
10: start()
11: startStreaming()
12: openSockets()
13: send()

Figure 4.12 Start Video Stream Sequence Diagram

c. Use Case Capture Video

The sequence diagram for the use case Capture Video describes how the application will set the required attributes for capturing a video. The figure below shows the sequence diagram.

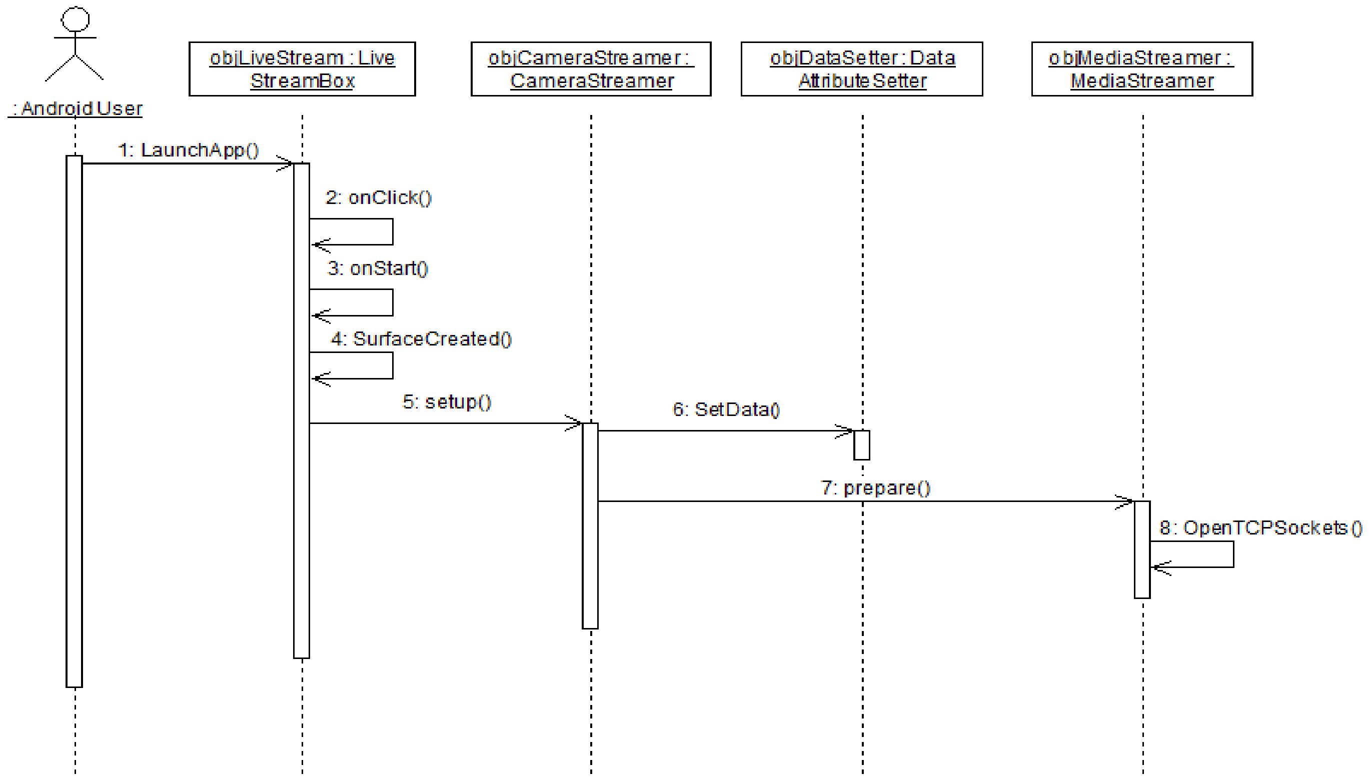

: Android User
objLiveStream : Live StreamBox
objCameraStreamer : CameraStreamer
objDataSetter : Data AttributeSetter
objMediaStreamer : MediaStreamer
1: LaunchApp()
2: onClick()
3: onStart()
4: SurfaceCreated()
5: setup()
6: SetData()
7: prepare()
8: OpenTCPSockets()

Figure 4.13 Capture Video Sequence Diagram

d. Use Case Streaming

The sequence diagram for the use case Streaming describes how the application starts streaming the video. The figure below shows its sequence diagram.

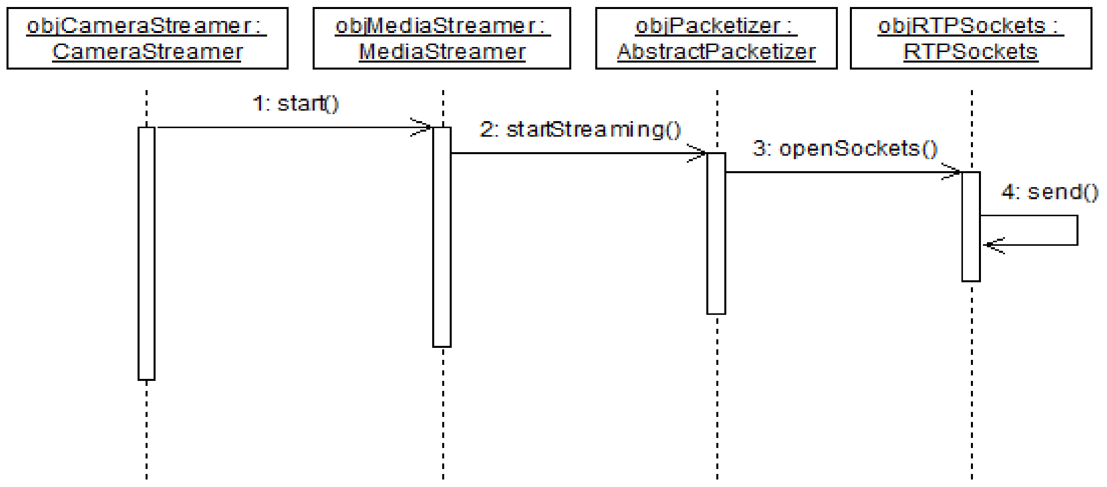
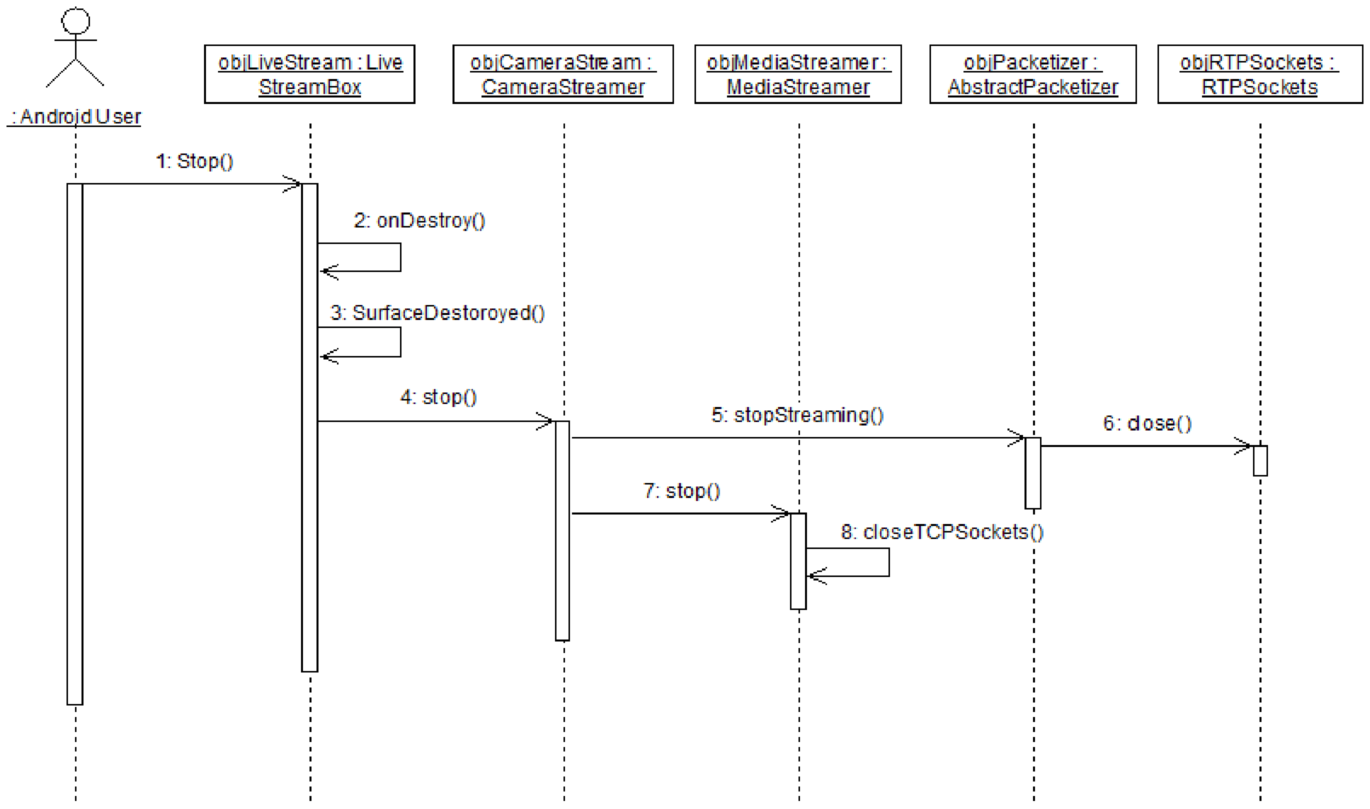

objCameraStreamer : CameraStreamer
objMediaStreamer : MediaStreamer
objPacketizer : AbstractPacketizer
objRTPSockets : RTPSockets
1: start()
2: startStreaming()
3: openSockets()
4: send()

Figure 4.14 Streaming Sequence Diagram

e. Use Case Stop Video Stream

The sequence diagram for the use case Stop Video Stream describes how a user stops the video stream. The figure below shows the sequence diagram for the use case.



: Android User
objLiveStream : Live StreamBox
objCameraStream : CameraStreamer
objMediaStreamer : MediaStreamer
objPacketizer : AbstractPacketizer
objRTPSockets : RTPSockets
1: Stop()
2: onDestroy()
3: SurfaceDestoroyed()
4: stop()
5: stopStreaming()
6: close()
7: stop()
8: closeTCPSockets()

Figure 4.15 Stop Video Stream Sequence Diagram

f. Use Case Visit Website

The sequence diagram for the use case Visit Website describes how a user views the live stream on the web. The user can be anyone on the web. The figurebelow shows the sequence diagram for the use case.

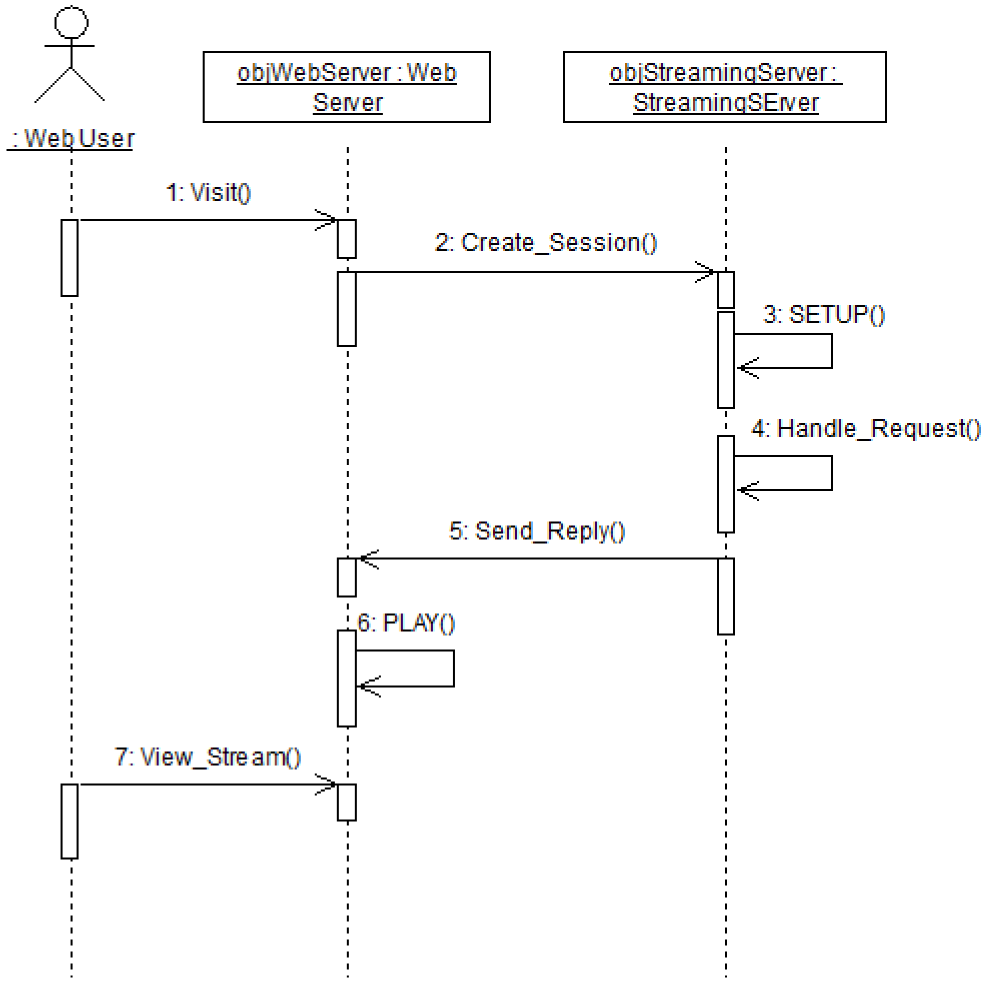


Figure 4.16 Visit Website Sequence Diagram

g. Use Case Browse Videos

The sequence diagram for the use case Browse Videos describes how a browse different live video streams. The users can be anyone on the internet. The figure below shows the sequence diagram for the Browse Videos use case.

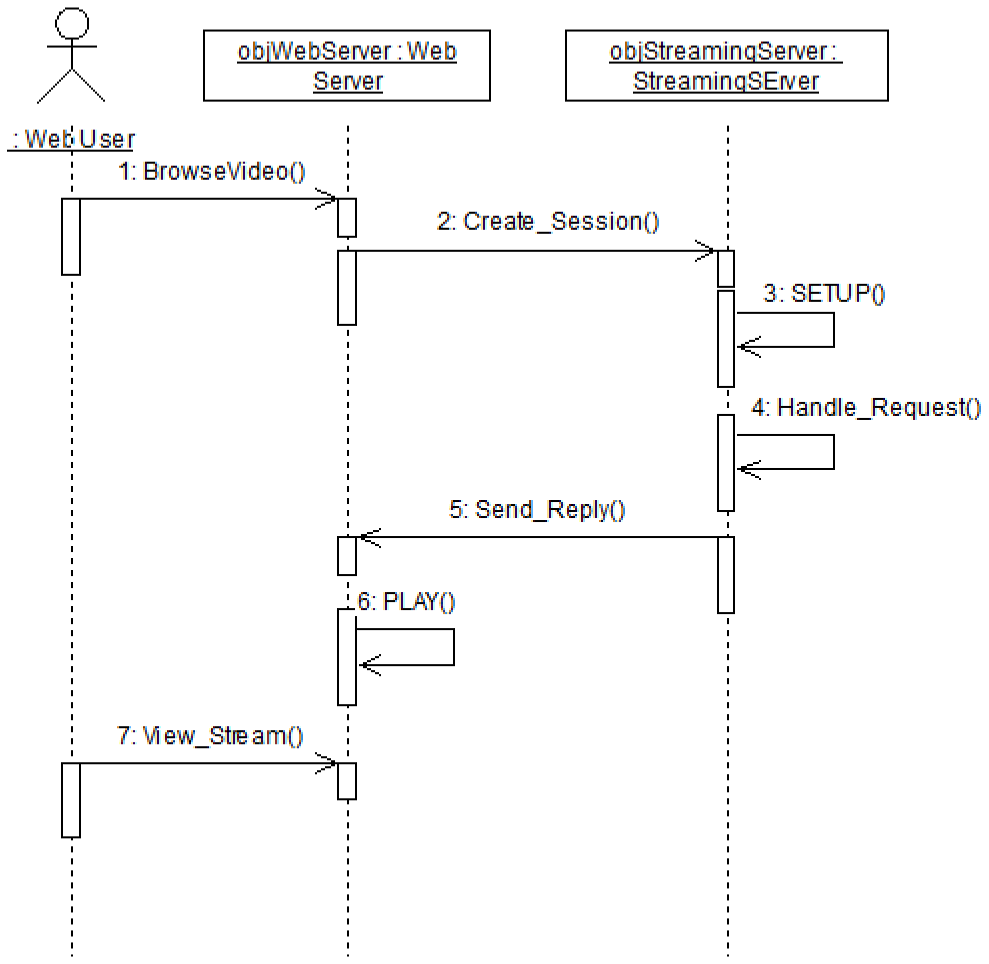


Figure 4.17 Browse Videos Sequence Diagram

4.4.3 Logical View

The logical view shows both the static and dynamic views of the system. The logical view concentrates on getting the best logical grouping of functionality into objects.

4.4.3.1 State machine models

a. System State Machine model

System state machine model is shown below in figure

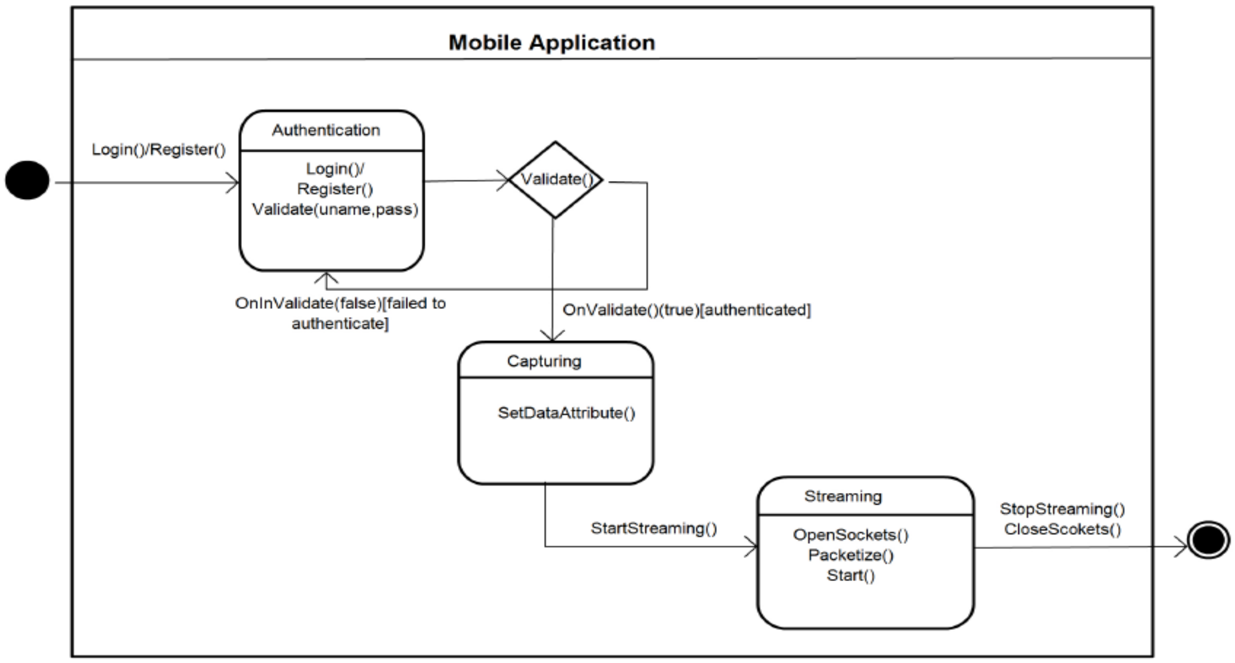


Figure 4.18 System State Machine Model

Description

a. Application gets Launched
b. User is asked to login into the system
c. Validate() function gets called for authentication
d. If the user enters correct username and password, validate function returns true and user gets authenticated. Now it continues with capturing.
e. If the user enters wrong username and password. Validate function return false and unauthenticated him by redirecting back.

f. During capturing setDataAttribute() function sets all Media Recorder data for recording.
g. On calling StartStreaming() function, sockets get opened, audio and video packets get created and streaming gets started.
h. User presses Quit button. StopStreaming() and closeSockets() functions are then called for stopping streaming and closing all connections.

b. Authentication State Machine model

The state machine model for authentication is given in the figure below.

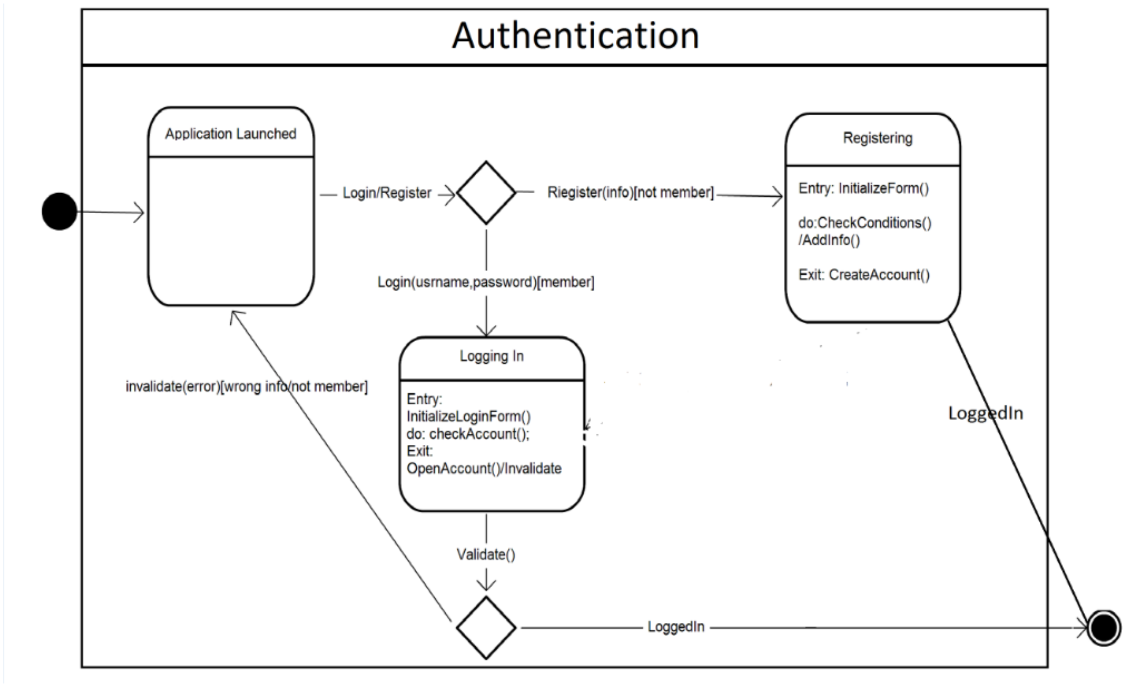


Figure 4.19 Authentication State Machine Model

Description

a. Application gets launched.
b. If user is not a member, it is directed to registering state. There user will be provided with registration form. He enters required information. Addinfo() enters info into the database on the basis of result from CheckConditions(). This CheckConditions() function returns true or false on the basis of some conditions like valid email, valid username. If returns true, user gets logged in to system else is not.

c. If user is already a member, he gets login into the system by entering correct username and password.
d. If entered username and password is misspelled and there is some error in login, user gets not logged in the system unless again login.

c. Capturing State Machine model

In the figure the state machine model is shown for the capturing state.

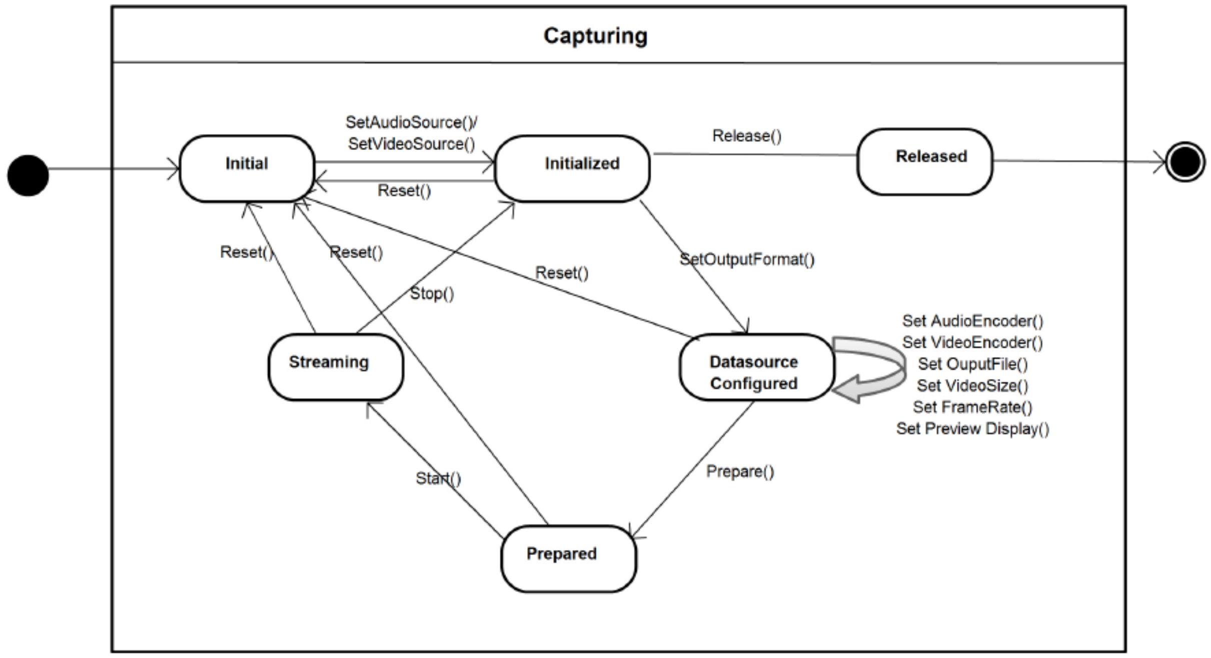


Figure 4.20 Capturing State Machine Model

Description

a. Source for both audio and video is set according to user's choice. Now mediarecorder is initialzed.
b. DataSource attributes for recording is configured by calling setting functions of mediarecorder.
c. Mediarecorder get prepared for recording on calling prepare()
d. Mediarecorder starts recording and streaming on calling start()
e. Capturing gets stopped by calling stop () method leading mediarecorder to the initialized state. Now user can again start recording in a different format and set data differently going through the same respective steps.

f. At any state, media recorder can be reset to an initial state by calling reset method.

d. Streaming State Machine model

The state machine model for streaming is shown in figure.

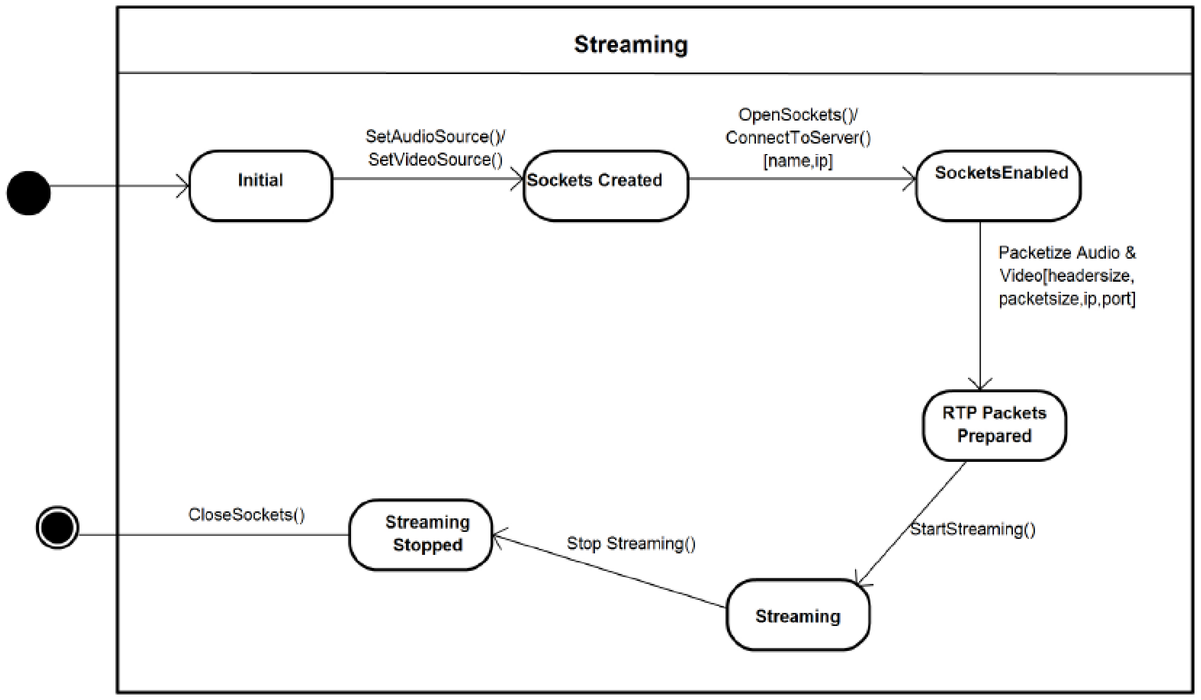


Figure 4.21 Streaming State Machine Model

Description

a. When all MediaRecorder gets prepared for capturing, Tcp connection gets established by creating socket connections and opening them for reliable session creation.
b. When sockets get enabled for streaming, create audio and video packets continuously for streaming and stream video.
c. Any time streaming can be stopped by the user by calling stopstreaming() method.
d. This method stops streaming and ends session by closing all connections.

4.4.4 Deployment Diagram

Two devices are used for all the setup. One is Android Phone for video streaming and the other device is the PC having our servers. Servers include “Darwin Streaming Server”, “IIS Server” and “SQL Server”.

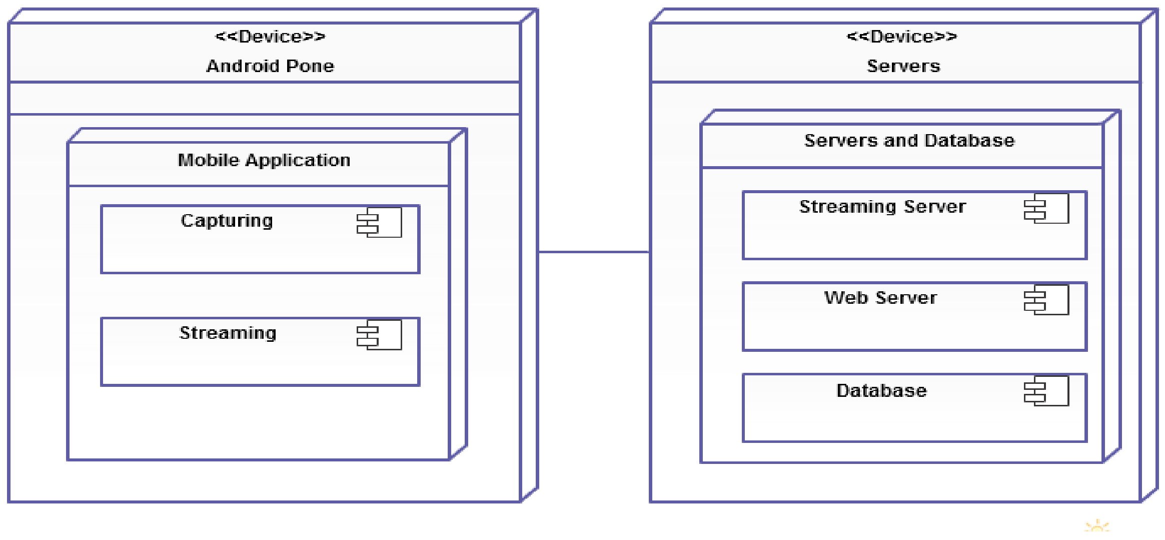


Figure 4.22 Deployment Diagram

4.4.5 Implementation View

The implementation view concentrates on taking the Logical view and dividing the logical entities into actual software components.

4.4.5.1 System Class Diagram

The application Live StreamBox comprises of many classes. The key classes are:

a. CameraStreamer
b. AbstractPacketizer
c. MediaStreamer
d. DataAttributeSetter
e. RTPSockets
f. Authentication
g. Media Recorder

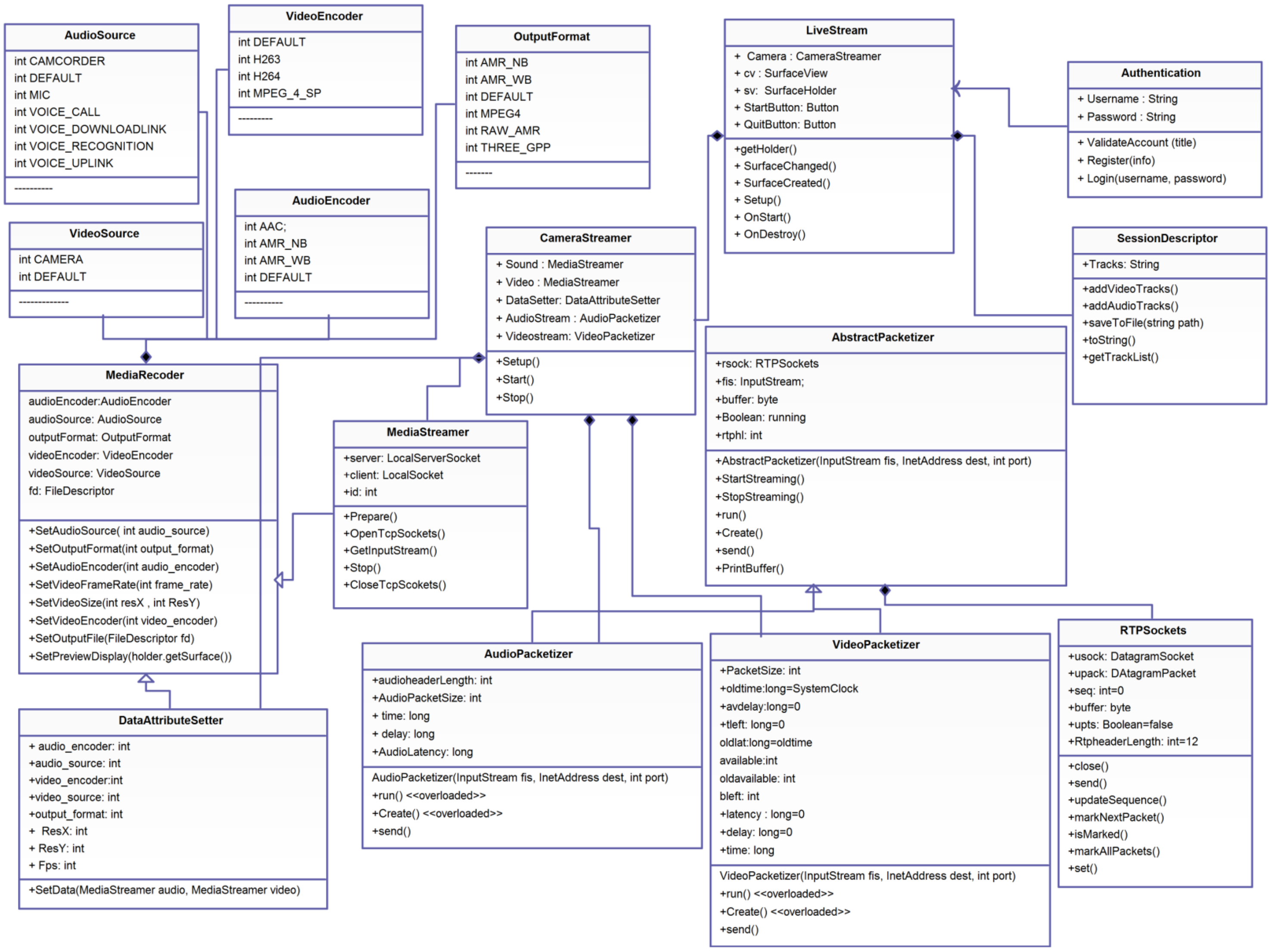


Figure 4.23 Class Diagram

4.4.5.2 System Classes Description

Table 4.9 Class Diagram Description

Name	**Description**
Authentication	Main Activity which loads the first Authentication interface for SignIn/Registration . This class validates entered username & password from database.
LiveStream	This class initializes and creates surface view window for recording and streaming video. It composes CameraStreamer class. It creates the object of class CameraStreamer and setups and starts streaming by calling its functions respectively. It also stop streaming on destroying surface
CameraStreamer	This class composes MediaStreamer, DataAttributeSetter, AudioPacketizer and VideoPacketizer. It first creates two audio and video objects of MediaStreamer class for recording audio and video. It then sets data attributes for recording by creating object of class DataAttributeSetter and calling its function setData(). It calls prepare method for creating Tcp sockets for connection. It creates object of classes AudioPacketizer and VideoPacketizer for audio and video packets streaming. After this whole setup, it finally starts recording as well as streaming of audio and video.
MediaStreamer	MediaStreamer class is a child class of MediaRecorder class. It opens and close TCP connection for session creation and get InputStream. It sets OutputFile with server filedescriptor of the file to be written into and then calls base class method prepare of MediaRecorder.
DataAttributeSetter	This class function setData() is concerned with setting audio and video data attributes for recording according to user choice. User can select different provided options of AudioSource, AudioEncoder, OutputFormat, VideoSource, VideoEncoder from interface. Options are then set by calling respective functions of base class that are setAudioSource(), setAudioEncoder(), setVideoSource(), setVideoEncoder(), setOutputFormat(). User can also select different resolution and frames per second from interface and set them through setVideoSize() and setVideoFrameRate() respectively.
MediaRecorder	This base class has child class MediaStreamer and nested classes AudioSource, AudioEncoder, OutputFormat, VideoSource, VideoEncoder. It has all basic functions of setting data attributes

Abstract Packetizer	This base class is concerned creating packets by creating and initializing object of class RTPSockets. Its functions StartStreaming() and StopStreaming() starts and stops the streaming thread on calling.
AudioPacketizer	This class creates audio packets and send them to the rtpSockets on a network
VideoPacketizer	This class creates video packets and send them to the rtpSockets on a network
RTPSockets	This class open RTP sockets, create RTP sockets and send them over the network. It also closes the connection. It update sequence number and check transmission of packets throught it. RTP sockets and packets are Datagram sockets and packets.
SessionDescriptor	This class generates sdp file on a specified path for running on a streaming server. It adds audio and video tracks.
AudioSource	This class has constants CAMCORDER, DEFAULT, MIC, VOICE_CALL, VOICE_DOWNLOADLINK, VOICE_RECOGNITION, VOICE_UPLINK. Anyone of these option can be passed to function setAudioSource(int audio_source)
AudioEncoder	This class has constants int AAC, AMR_NB, AMR_WB, DEFAULT. Anyone of these option can be passed to function setAudioEncoder(int audio_encoder)
OutputFormat	This class has constants AMR_NB, AMR_WB, DEFAULT, MPEG4, RAW_AMR THREE_GPP. Anyone of these option can be passed to function setOutputFormat(int output_format)
VideoSource	This class has constants CAMERA, DEFAULT. Anyone of these option can be passed to function setVideoSource(int video_source)
VideoEncoder	This class has constants DEFAULT, H263, H264, MPEG_4_SP. Anyone of these option can be passed to function setvideoEncoder(int video_encoder)

4.4.5.3 System Classes Attributes and Functions Description

Table 4.10 Authentication Class

Authentication	
AuthenticateAccount(uname, pass)	Check username and password in database
Register(info)	Enter user info to the database and set its status as member
Login(username, password)	Call AuthenticateAccount. If user has entered valid username and password, log him to the application and allow streaming

Table 4.11 LiveStream Class

LiveStream	
Camera	Object of CameraStreamer class.
Cv	Surfaceview for viewing window
Sv	Surfaceholder for holding view
startButton	For starting streaming
stopButton	For stoping streaming
getholder()	Get holder to receive information about the surface changed
SurfaceChanged(SurfaceHolder holder, int format, int width, int height)	This is called when some strucutral changes have been made to the surface
SurfaceCreated(SurfaceHolder holder)	This function initializes surfaceview object with cameraview instance from interface and initalizes surfaceholder by getting holder of this surfaceview object. It creates CameraStreamer object
OnStart()	This function calls setup(holder) of CameraStreamer obj for preparing audio and video, opening and closing tcp connections and creating audio, video packets. It then calls start() function of cameraStreamer obj for starting streaming.
OnDestroy()	This class destroy the surface and calls stopStreaming function of CameraStreamer obj for stopping streaming and closing connections

Table 4.12 CameraStreamer

CameraStreamer	
Sound	MediaStreamer object for preparing audio and opening sockets
Video	MediaStreamer object for preparing video and opening sockets
DataSetter	DataAttributeSetter object for calling setData() function for setting recording data attributes of audio and video
AudioStream	AudioPacketizer object for creating audio packets
VideoStream	VideoPacketizer object for creating video packets
Setup(SurfaceHolder holder, String ip)	This function calls datasetter function of object DataSetter for setting data attributes. It calls preapre function of sound and video

	object for preparing media and opening tcp connections. It then initializes audioPacketizer and VideoPacketizer constructors with inputstream and ip.
Start()	This function Start MediaRecorder for recording and start streaming by running thread of streaming audio and video.
Stop()	This function stops Recording and streaming of both audio and video.

Table 4.13 DataAttributeSetter Class

DataAttributeSetter	
audio_encoder, video_source, video_encoder, video_encoder	This attribute is set by user from interface menu options.
ResX	Width of video is set by user from interface menu options.
ResY	Hieght of video is set by user from interface menu options.
Fps	Frame sper second is set by user from interface menu options.
setData(MediaStreamer audio, MediaStreamer video)	This method initializes the above data attributes from the user selected options from interface. The options for these data are constant attributes of base nested classes AudioSource, AudioEncoder, OutputFormat, VideoSource, VideoEncoder. The above data attributes are then passed to respective funtions of base MediaRecorder class. These functions are setAudioSource(), setAudioEncoder(), setVideoSource(), setVideoEncoder(), setOutputFormat().

Table 4.14 RTPSockets Class

RTPSockets	
Rsock	DatagramSocket
Rpack	DatagramPacket
Seq	Sequence number for maintaining packets sequence.
Buffer	Buffer for data to hold
rtpHeaderLength	Rtp header
Create()	Creating rtp packets
updateSequence()	For maintaining seq number
Send()	Send packets on network

Table 4.15 MediaStreamer Class

MediaStreamer	
LocketSocket	Socket for client
LocketServerSocket	Socket for server
Prepare()	This function creare and open tcp sockets, get file descriptor from server opened sockets and setouputfile by calling this setOutputFile(filedescriptor) function.
setOutputFile()	Base class MediaRecorder function for setting filedescriptor of file the stream written into.
getInputStream()	This function get input stream from opened tcp sockets
closeSockets()	This class closes tcp connections on calling
Stop()	This function stops MediaRecorder recording and call closesockets()

4.5 Data Flow Diagram

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. The figure below shows the data flow diagram.

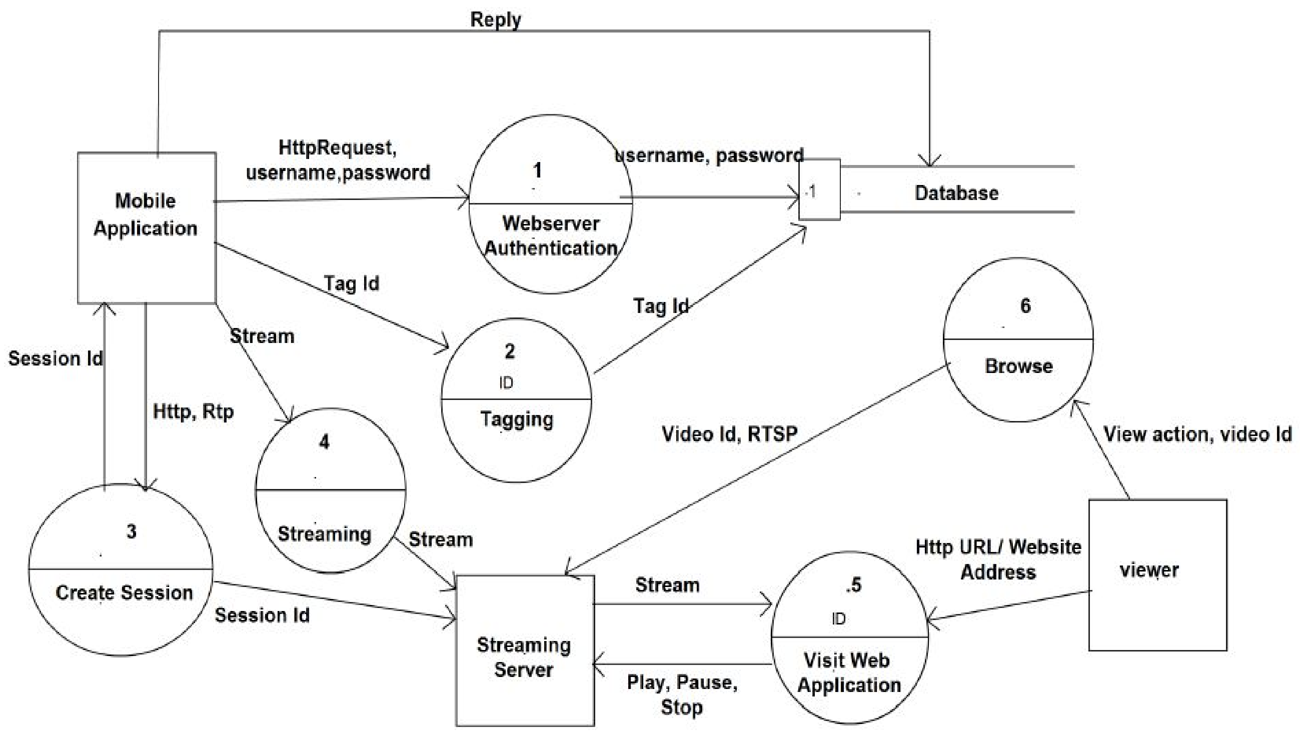


Figure 4.24 Data Flow Diagram

4.6 Design Patterns

4.6.1 Polymorphism

We have also used a pattern Polymorphism. It is a fundamental principle in designing how a system is organized to handle similar variations. According to this, make subclass objects responsible for their own behaviour and let proper behaviour be invoked automatically by subclasses.

In our scenario, AbstractPacketizer class has a behaviour function **"CREATE"** (refer to Figure). Create function will perform differently in creating audio and video packets.

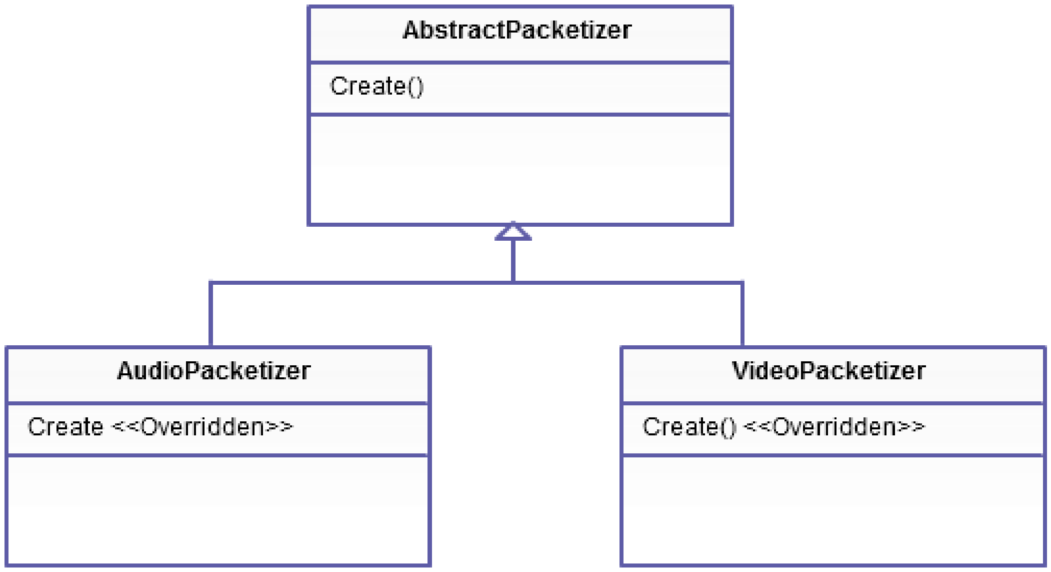


Figure 4.25 Polymorphism Pattern

4.6.2 Creator Pattern

We have used the creator pattern. This pattern is based on the principle for the assignment of creation responsibilities. By this, low coupling, increased simplicity, encapsulation, and reusability is supported by this design.

If one of following gets true (the more the better), Assign class B the responsibility to create an instance of class A ():

- B "contains" or compositely aggregates A
- B records A

- B closely uses A
- B has the initializing data for A that will be passed to A when it is created.
- Thus with respect to creating A, B is an Expert.

Generallyselect a class B which aggregates or contains class A, if more than one option applies,

There are different scenarios in which the creator pattern has been used:

Scenario 1:

Refer to figure 30 **LiveStream** class is creating the objects of the following classes:

a) **CameraStreamer** and
b) **SessionDescriptor**

Therefore,

- LiveStream "contains" or compositely aggregates CameraStreamer and SessionDescriptor
- LiveStream records CameraStreamer and SessionDescriptor
- LiveStream closely uses CameraStreamer and SessionDescriptor
- Thus LiveStream is an expert with respect to creating CameraStreamer

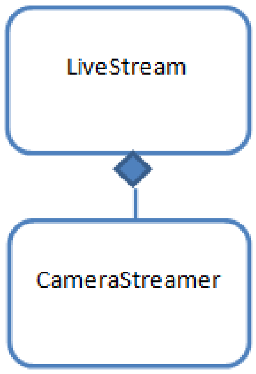


Figure 4.26 Creator Pattern Scenario 1

Scenario 2:

Refer to figure 31 **CameraStreamer** class is creating the objects of the following classes:

a) MediaStreamer

b) AudioPacketizer

c) VideoPacketizer

Therefore,

- CameraStreamer "contains" or compositely aggregates MediaStreamer, DataAttributeSetter, AudioPacketizer, VideoPacketizer
- CameraStreamer records MediaStreamer, DataAttributeSetter, AudioPacketizer and VideoPacketizer.
- CameraStreamer closely uses MediaStreamer, AudioPacketizer VideoPacketizer and DataAttributeSetter.
- CameraStreamer has the initially data for AudioPacketizer and VideoPacketizer that will be passed to these when it is created

Thus CameraStreamer is an expert with respect to creating MediaStreamer, AudioPacketizer, VideoPacketizer and DataAttributeSetter.

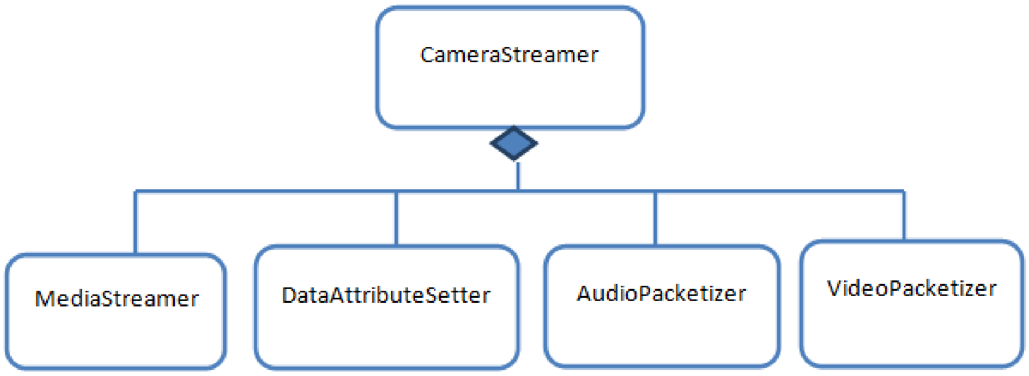


Figure 4.27 Creator Pattern Scenario 2

Scenario 3:

Refer to figure 32 **AbstractPacketizer** class is creating the objects of the following class RTPSockets.

Therefore,

- AbstractPacketizer “contains” or compositely aggregates RTPSockets.
- AbstractPacketizer records RTPSockets.
- AbstractPacketizer closely uses RTPSockets.
- Thus AbstractPacketizer is an expert with respect to creating RTPSockets.

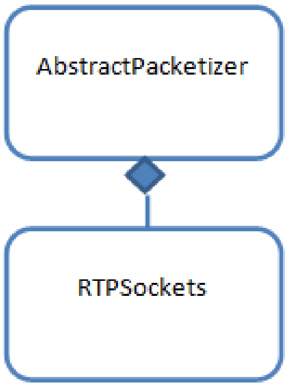


Figure 4.28 Creator Pattern Scenario 3

Scenario 4:

Refer to figure 33 **MediaRecorder** class is creating the objects of following classes:

- AudioSource
- AudioEncoder
- VideoSource
- VideoEncoder
- OutputFormat

Therefore,

- MediaRecorder “contains” or compositely aggregates AudioSource, AudioEncoder, VideoSource, VideoEncoder, OutputFormat.

- MediaRecorder records AudioSource, AudioEncoder, VideoSource, VideoEncoder, OutputFormat.
- MediaRecorder closely uses AudioSource, AudioEncoder, VideoSource, VideoEncoder, OutputFormat.

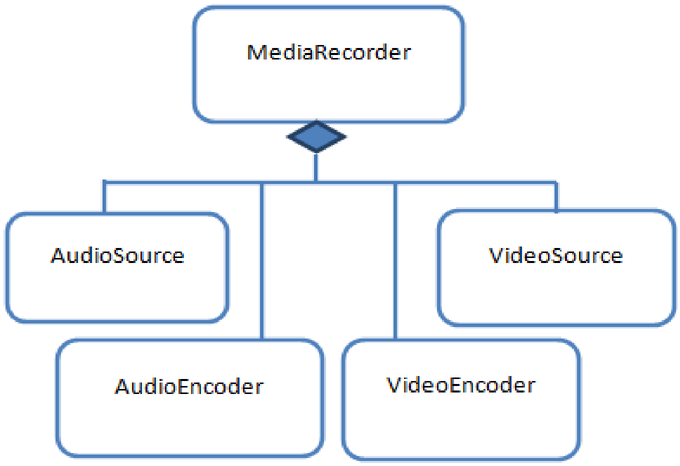


Figure 4.29 Creator Pattern Scenario 4

Chapter 5: System Implementation

5.1 Live Streaming

Streaming video is compressed form content sent over the Internet and presented by the viewer in real time. To play a streaming video file, a Web user does not have to wait for it to play. As an alternative, the media is played as it arrives as is sent in a continuous stream of data.

5.2 Live StreamBox

Our project LiveStreamBox is a system that allows the users to upload live videos from their cell phones to a streaming server on the internet. These videos then can be viewed live by anyone in the world from our website.

Project has three main modules

1. Android Application
2. Streaming Server Session Handling
3. Web Service

5.2.1 Android Application:

5.2.1.1 Devices, development Tools &API's

Android:

Android is a Linux-based operating system for mobile devices. The Android Open Source Project (AOSP) is developed by the Open Handset Alliance is led by Google and has been updated frequently, with each fixing bugs and adding new features. Now Android 4.0.4 Ice Cream Sandwich is the latest. Our application is using the version 2.3.3 Gingerbread.

Android have a special Java virtual machine called Dalvik Virtual Machine which it used to run Java applications.It isaugmented for mobile devices. To develop java applications for Android, Google has published a software development kit (SDK) that comes with a Java library.

SDK:

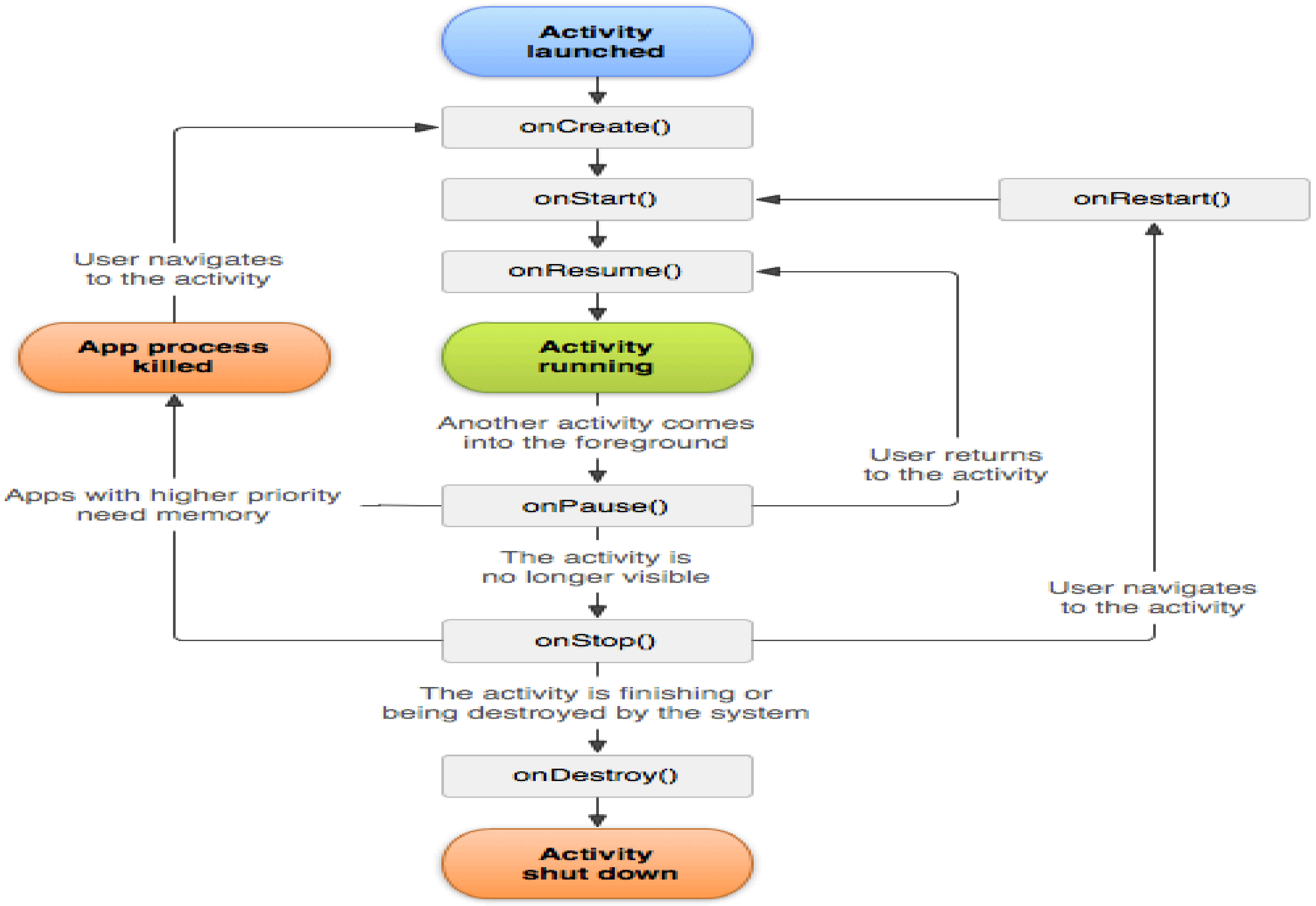
The SDK make available tools for building mobile applications. There is also a built-in emulator is someone don’t have a real Android device.

Libraries:

The Libraries offers a rich framework for developing mobile applications. It offers GUI elements along with other elements to control the tasks of the phone. For this thesis, the important libraries and classes are activity, android. View, android.io, android.net and very important android media library.

Activity:

An Activity is a main class of the Android framework. This class characterizes a visual user interface and code to maintain all. Every user interface element showed on the screen has background controlling Activity object. Its phases of life are:



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Figure 5.1 Activity launching Android

Eclipse:

Eclipse is a multi-language supported software development environment (IDE). It is originally established and still financed by IBM. It is an open source and the Eclipse SDK (which includes the Java development tools) is expected for Java developers.

Our application has been built in Eclipse Helios 3.5.

Android Media Player:

Android has anintegral built in media player framework for playing video and audio. It supports many playbacks like of RTSP URLs, local files and also HTTP URLs. From device to device and to emulator, thesepublically supported file formats and codecs differ and are pronounced by Google in very universal terms.

5.2.1.2 Description

Mobile application comprises video capturing, encoding, packetization and delivering of this content over the network.

1. Authentication& Tagging:

The application user has to be authenticated by the web server before starting streaming.

User enters his/her username and password. Http connection is made with the web service using android http libraries in the apache package. Username and password are sent as connection parameters along. The web application validates the user on the basis of correct username and password else declares “not a member”.

After authentication, tagging screen provides the user with different tagging options. This tag option will also be sent to web service. This streamed video will then be available to the visitors of the website under the same tag category as selected.

2. Video Streaming :

After installing, when the user launches Live StreamBox application, after authenticationmain stream activity gets start and sets the following recording parameters.

- Ip
- ResolutionX
- ResolutonY
- FramesPerSecond
- BitRate

The very first time when the application gets launched in the mobile, it tests the mobile for H.264 support by running a test.

H.264:

H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) is a standard for video compression, and is now one of the most frequentlyused formats for the recording and compression. It is also used for sharing of high definition video. It is a block-oriented as well as motion-compensation-based codec. It is established by the ITU-T Video Coding Experts Group (VCEG) mainly in support with International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) joint working group, the Moving Picture Experts Group (MPEG).

The commitment of the H.264/AVC project was to create a standard actually that would accomplished providing good video quality. The quality would be at significantly lower bit rates than preceding standards (i.e., half or less the bit rate of MPEG-2, H.263, or MPEG-4 Part 2). In order to avoid impracticality and much expensiveness of its implementation, it would be deprived ofaggregating the complexity of design so much. To permit the standard to be functional on an extensive diversity of applications and on a range of networks and systems, plus low and high bit rates, varied resolution video, DVD storage, RTP/IP packet networks, and ITU-T multimedia telephony systems, it would provide adequate flexibility.

An additional goal was to provide enough flexibility to allow

It sets up the Media Recorder and record the dummy video for 500msec. While testing it parse the dummy video file with MP4Parser and if correctly parsed it finds the stsd box that's contains H.264 encoding parameters. Profile level information and these encoding parameters SPS (sequenceParameterSet) and PPS (pictureParameterSet) are stored in theavcC box.

AVC:

AVC specifies a set of Network Abstraction Layer (NAL) units, which contain different types of data. Thissub clause specifies the format of the elementary streams for storing such AVC content within the AVC file format. Two types of elementary streams are defined for this purpose.

1. **Video elementary streams** shall contain all video coding related NAL units (i.e. those NAL unitscontaining video data or signaling video structure) and may contain non-video coding related NAL units such as SEI message and access unit delimiter NAL units.
2. **Parameter set elementary streams** shall contain only sequence and picture parameter set NAL units

The **box structure** is described below:

```
aligned(8) class AVCDecoderConfigurationRecord {

unsignedint(8) configurationVersion = 1;

unsignedint(8) AVCProfileIndication;

unsignedint(8) profile_compatibility;

unsignedint(8) AVCLevelIndication;

bit(6) reserved = â€˜111111â€™b;

unsignedint(2) lengthSizeMinusOne;

bit(3) reserved = â€˜111â€™b;

unsignedint(5) numOfSequenceParameterSets;

for (i=0; i<numOfSequenceParameterSets; i++) {
```

```
unsignedint(16) sequenceParameterSetLength ;

bit(8*sequenceParameterSetLength) sequenceParameterSetNALUnit;

}

unsignedint(8) numOfPictureParameterSets;

for (i=0; i<numOfPictureParameterSets; i++) {

unsignedint(16) pictureParameterSetLength;

bit(8*pictureParameterSetLength) pictureParameterSetNALUnit;

}

}
```

Once these parameters get generated, our test become successful showing phone supports H.264 encoding. Application then writes these parameters in session description file .sdp

Session Description File:

The Session Description Protocol (SDP) is a presentation for describing streaming media in terms of its initialization parameters.SDP is proposed for the description of multimedia communication sessions. This session include session announcement, session invitation, and parameter negotiation. SDP does not send media itself but is used for negotiation of media type, format, and all related properties between end points. These set of properties and factors are often called a session profile. SDP is intended to be extensible to the provision of new media types and formats.

SDP is taking place as a factor of the Session Announcement Protocol (SAP), but found other uses in aggregation with Real-time Transport Protocol (RTP), Real-time Streaming Protocol (RTSP), and Session Initiation Protocol (SIP). It is also even found as a separate format for describing multicast sessions.

A session is described by a sequences of fields, one per line. The arrangement of each field is as follows.

<Character>=<value>

An SDP message has three main sections.

- Detailing the session
- Timing
- Media descriptions

Each message may compriseseveral timing and media descriptions. Names are only distinctive within the accompanying syntactic build, i.e. within the session, time, or media.

Our **sdp file** contains:

"v=0"

"o=- 15216309810768491287 15216309810768491287 IN IP4"+senderip+"\r\n"

"s=Unnamed\r\n"

"i=N/A\r\n"+

"c=IN IP4 192.168.13.176\r\n"

"t=0 0\r\n"

"a=tool:vlc 2.0.1\r\n"

"a=recvonly\r\n"

"a=type:broadcast\r\n"

"a=charset:UTF-8\r\n"

"m=video 5006 RTP/AVP 96\r\n"

"b=RR:0\r\n"

"a=rtpmap:96 H264/90000\r\n"

"a=fmtp:96 packetization-mode=1;profile-level-id="+profile+";sprop-parameter-sets="+b64sps+","+b64pps+";\r\n"

Now the application is ready for the user to start streaming. This all is done in starting application.

When the user presses the start button, Media Recorder attributes will get set.

Media Recorder:

This is the media recorder class of media library. It has 5 final nested classes provide constant variables for setting different media attributes for recording.

These are named as:

a. AUDIO SOURCE
b. AUDIO ENCODER
c. OUTPUT FORMAT
d. VIDEO SOURCE
e. VIDEO ENCODER

Media Recorder class has respective following functions for setting Media Recorder by these classes' different attributes.

a. SetAudioSource()
b. SetAudioEncoder()
c. SetOutput Format()
d. SetVideoSource()
e. SetVideoEncoder()

This application is using following:

a. **"CAMCODER"** as Audio Source
b. **"RAW_AMR"** as audio Output Format
c. **"AMR_NB"** as Audio Encoder
d. **"CAMERA"** as Video Source

e. **“THREE_GPP”** as video Output Format
f. **“H264”** as Video Encoder.

Media Recorder class also has built in functions for setting other media attributes.

a. SetVideoSize(intresX,intresY)
b. SetVideoFrameRate(int r)
c. SetAudioChannels(int n)
d. setVideoEncodingBitRate(int b)

User can change resolution, frames per second and data bit rate while from quality option provided in the application.

Over low bandwidth networks, video resolution, frames per second and data bit rate can be decreased for improving efficiency in terms of delay and quality of video during streaming.

Application sends the audio and video streams through UDP/RTP for real time streaming.

As recording parameters get set, there is a file like Media Recorder that writes in a local socket instead of a file so that you can modify data on-the-fly with getInputStream().

Audio Packets:

For audio, input stream from camera is continuously get with getstream and packetize. AMRNB packets will be made and sent in the udp stream. For AMR, the sampling frequency is 8 kHz, corresponding to 160 encoded speech samples per frame from each channel so timestamp gets updated at the rate of 160.

Video Packets:

For video, input stream from camera is continuously get with getstream and packetize. For H.264 packets, NAL units are read in some file and send over the rtp network. If there are NAL units in the buffer ready to be sent, we send one.

NAL:

The coded video data is ordered into NAL units. Every unit is efficiently a packet that holds an integer number of bytes. The header byte ,the first byte of each NAL unit indicates the type of data in the NAL unit and accordingly the remaining bytes containing payload data.

Socket connections will be made on the defined Streaming Server “ip” and “port” for establishing connection with it. It sets OutputFile at the server with socket filedescriptor of the file to be written into.

RTP Packets:

The video is transmitted through the RTP. As cellular networks are not extremely reliable and signal may fluctuate, the RTP Conversion takes place on the Android device. This requires more processing power and battery life than doing it on the server but will provide a continuous stream if the connection is temporarily lost or does not have the required bandwidth to transmit the full video by skipping packets at the server that never arrive and creating an image from the packets received at the server.

In order to produce the RTP packets, conversions from the camera devicesneeds to occur. The phone sends the H.264 provided by the Android API to alocal TCP port on the phone; the phone receives the packets from this localport then converts the packets to RTP H.263 UDP packets by wrappingsequential frames with a RTP header as shown in figure 2. The raw video issplit into packets around 1500 bytes. The data is sent in blocks with themarker set in the header at the beginning of each block. The RTP packets arethen forwarded to the server on the corresponding video UDP port receivedearlier.
The RTP header contains pieces of information so that the frame may bedecoded on the server and viewing client end to produce a viewable video.This conversion to RTP is required because the H.263 video codec does notsend information about the video such as length until the closing of arecording and it stores this information footer of the file. Thus, RTP packetscontain the time length of the packet, or timestamp along with a sequencenumber to remedy this problem. Other information that is contained in theRTP packet consists of a source identifier, a marker, a version and the payload type, in our case H.264.

These rtp packets are directly streamed into the sdp file placed at the DARWIN streaming server media folderthrough sockets mentioned in sdp file specification.

5.2.2 Streaming Server

5.2.2.1 Darwin Streaming Server (DSS):

It is the first open sourced RTP/RTSP streaming server. It was outon March 16, 1999. It is proficient of streaming range of media types including MPEG-4 Part 2 , H.264/MPEG-4 AVC, and 3GP.

It is the open source correspondent of QuickTime Streaming Server. It is developed by Apple and is based on its code.The original DSS source code release compiled only on Mac OS X, but external developers and programmers swiftly ported the code to Linux, Tru64 Unix, FreeBSD, Solaris, Mac OS 9 and Windows.

5.2.2.2 Streaming Protocols:

RTSP:

The Real Time Streaming Protocol (RTSP) is a network control protocol. It is aimed for controlling streaming media servers for use in entertainment and communications systems. The protocol is used for establishing and monitoring media sessions between end points. In order to assist real-time control of playback of media files from the server, clients of media servers issue VCR-like commands, such as play and pause.

The RTSP servers are not concerned with the broadcast of streaming data itself. For media stream delivery, many RTSP servers use the Real-time Transport Protocol (RTP) in combination with Real-time Control Protocol (RTCP).

With the rtspurl “rtsp:serverip:554/filename.sdp” embedded in the web page, web server will directly get the stream from user sdp file in the Darwin media folder.

5.2.3 Web Service

Live stream box system has a web User Interface Part. Where users can browse and can watch live videos. This web application built using different technologies. The important thing in web application is web user interface.

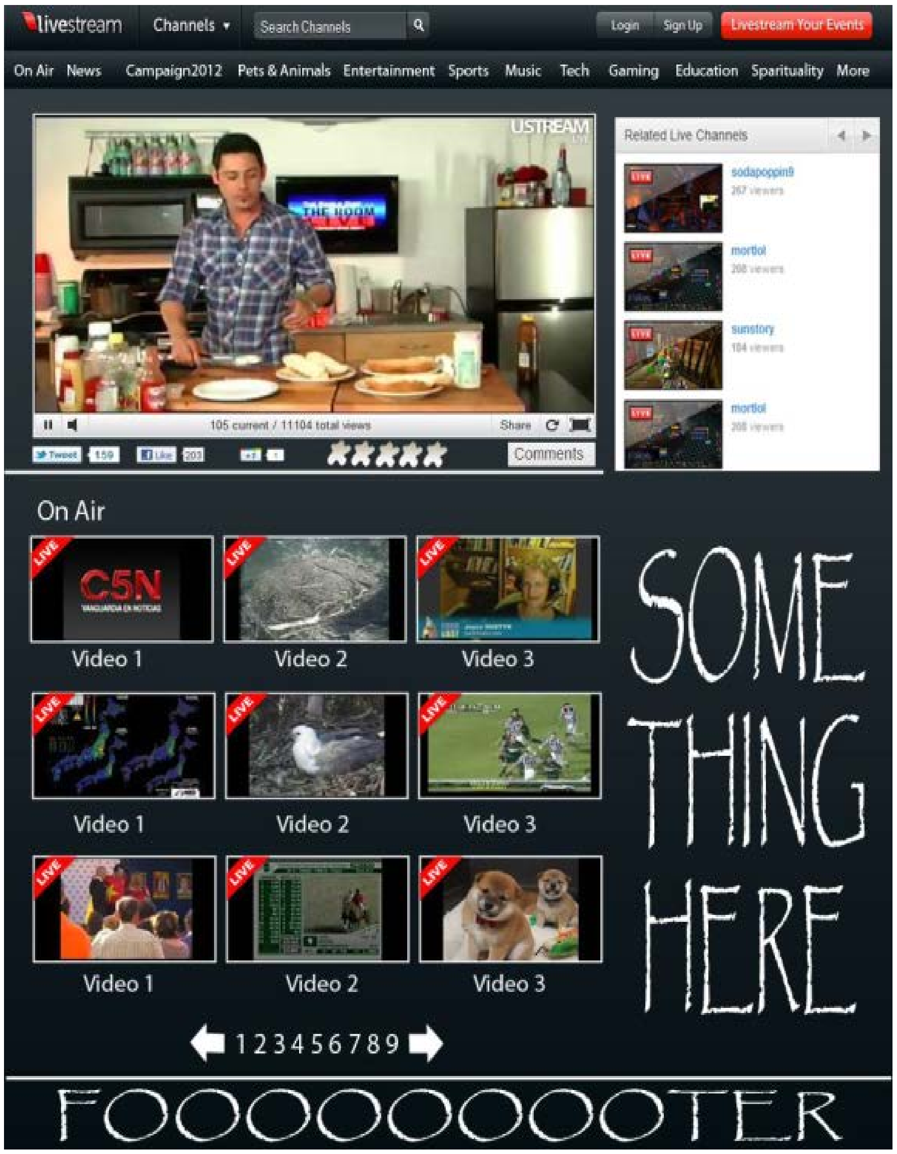
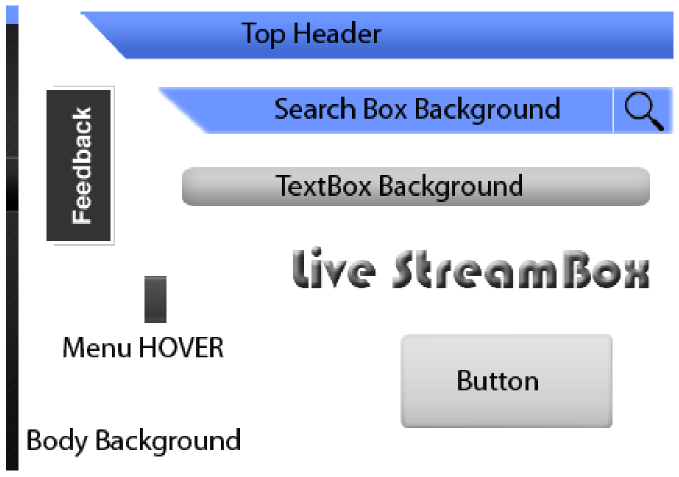


Figure 5.2 Webend

5.2.3.1 The implementation of web service



Development tools and database services

Adobe Photoshop CS3/CS5

We used Adobe Photoshop for creating and designing UI for web application. We tried with different colors scheme and layouts. We changed and tested our design for different layouts and then finalized. The finalized user interface was sliced (slicing is said to cut images from main user interface and then used for building web using CSS to make web UI as designed)

Dreamweaver CS5

Dream Viewer tool is used for building static HTML webs and setting cascading style. We used to develop HTML form of our web application, and wrote Cascade Style Sheet according to design of web interface.

Visual Studio 2010

Microsoft Visual Studio 2010 Professional is the important tool for thosewho perform basic development tasks. It shortens the creation, debugging, and deployment of applications on a range of platforms containing SharePoint and the Cloud. Visual Studio 2010 Professional arises with integrated support for test-driven development, as well as debugging tools that help guarantee high-quality solutions. (Uses .NET Framework 4)

.NET Framework 4

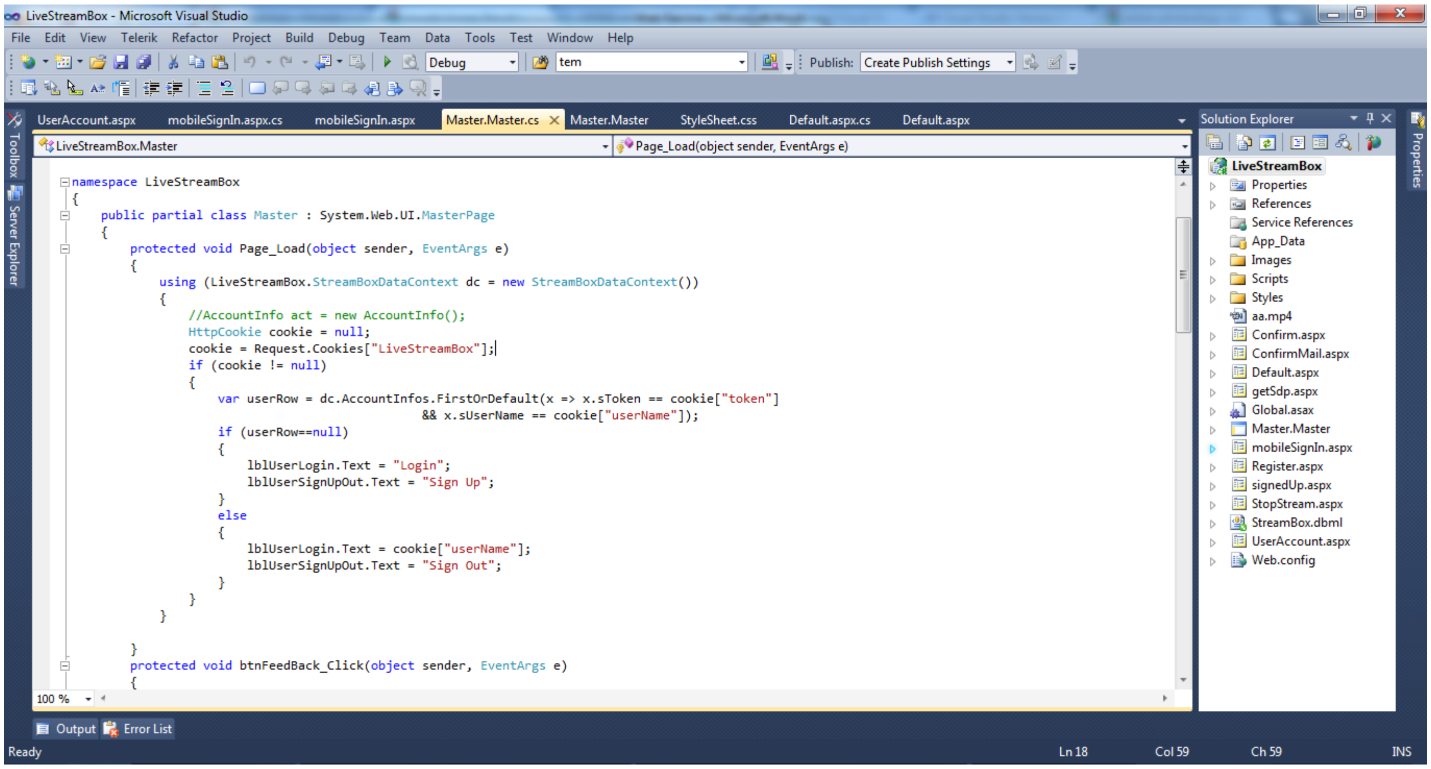
The .NET Framework is an application development platform. Itoffers services for building, deploying, and running desktop, web, and phone applications as well as web services. It comprises of two major constituents: the common language runtime (CLR), which affords memory management and other system services. It provides comprehensive class library for testing and reusing code for all key areas of application development.

ASP.net

ASP.NET, a part of the .NET Framework, provides classes and tools for creating dynamic Web applications. The topics in this section guide you in creating, programming, and deploying ASP.NET Web applications.

ASP.NET is a service providing based web platform for building enterprise-class server-based Web applications. ASP.NET is manufactured on the .NET Framework, so all .NET Framework structures are available to ASP.NET applications. Our applications can be written in any language that is well-matched with the common language runtime (CLR), containing Visual Basic and C#.

We used Visual Studioto create ASP.NET Web applications,. The tools and choices provided in the Visual Studio are considered for creating Web applications and these are together stated as Visual Web Developer. In addition, a free standalone product—Visual Web Developer Express—is presented that comprises the principal set of Web-design features from Visual Studio.

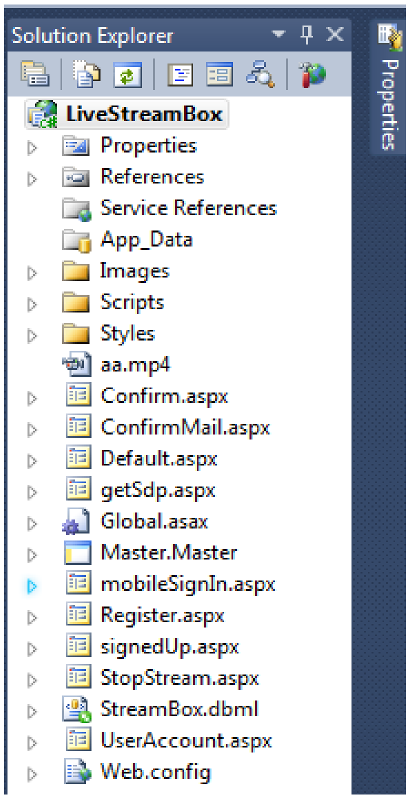


```
namespace LiveStreamBox
{
    public partial class Master : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            using (LiveStreamBox.StreamBoxDataContext dc = new StreamBoxDataContext())
            {
                //AccountInfo act = new AccountInfo();
                HttpCookie cookie = null;
                cookie = Request.Cookies["LiveStreamBox"];
                if (cookie != null)
                {
                    var userRow = dc.AccountInfos.FirstOrDefault(x => x.sToken == cookie["token"]
                                                && x.sUserName == cookie["userName"]);
                    if (userRow==null)
                    {
                        lblUserLogin.Text = "Login";
                        lblUserSignUpOut.Text = "Sign Up";
                    }
                    else
                    {
                        lblUserLogin.Text = cookie["userName"];
                        lblUserSignUpOut.Text = "Sign Out";
                    }
                }
            }

        }
        protected void btnFeedBack_Click(object sender, EventArgs e)
        {
```

5.2.3.2 Implementation of Web Application ASP.net

We have different web pages in our application, used for different requirements. And every page of ASP.net has frontend & backend code.



Master.Master

This is master page containing common UI things. The purpose of this Master page is to avoid from redundancy of common used thing like header footer, or authentication of users for pages to be used by authenticated users.

In our frontend of Master Page Header and Footer is programmed.

While in backend we have checked authentication of user, if authenticated then display name in header. And also maintains cookies

protectedvoidPage_Load(object sender, EventArgs e)

```
        {
using (LiveStreamBox.StreamBoxDataContext dc = newStreamBoxDataContext())
            {
HttpCookie cookie = null;
                cookie = Request.Cookies["LiveStreamBox"];
if (cookie != null)
                {
varuserRow = dc.AccountInfos.FirstOrDefault(x =>x.sToken == cookie["token"]
&&x.sUserName == cookie["userName"]);
if (userRow==null)
                    {
lblUserLogin.Text = "Login";
lblUserSignUpOut.Text = "Sign Up";
                    }
else
                    {
lblUserLogin.Text = cookie["userName"];
lblUserSignUpOut.Text = "Sign Out";
                    }
                }
            }

        }
```

Default.aspx

This page is default page or index page. And inherited to Masrer.Master

Frontend contains video play plug-in of VLC and related videos. Whenever user browses to this page; the latest video will be started streaming. And he can also browse other videos from this page’s menu (i.e. Health, Entertainment etc). This page will be redirected to itself containing some arguments (Name of Tag) in URL using Get Method. Attributes identify to database which video to start playing in VLC plug-in.

In backend we check that if any argument in URL then loads latest streaming video's path according to Tag from database and embed it to VLC plug-in. else browse the latest video to Default.aspx. The path set to VLC plug-in will be targeting to streaming server, because video has to stream from streaming server. And the path will be like rtsp://192.168.13.13/liveVideo.sdp.

```
<embedtype="application/x-vlc-
plugin"pluginspage="http://www.videolan.org"version="VideoLAN.VLCPlugin.2"
width="100%"height="100%"id="vlc"loop="yes"autoplay="yes"target="<%# "rtsp://" +
Eval("filePath") + ".sdp" %>">
```

UserAccount.aspx

This page is used for Signing In to our Web application. Frontend contains textboxes and button for authentication. Backend coding takes username and password from frontend and matches with database.

```
using (LiveStreamBox.StreamBoxDataContext dc = newStreamBoxDataContext())
        {
AccountInfoacc = newAccountInfo();
UserDataInfo user = newUserDataInfo();
stringuserName = txtUserName.Text.Trim();
string password = txtPassword.Text.Trim();
varaccRow                                                                    =
dc.AccountInfos.FirstOrDefault(x=>x.sUserName==userName&&x.sPassword==passwo
rd);
if (accRow != null)
           {
HttpCookie cookie = newHttpCookie("LiveStreamBox");
cookie.Values.Add("userName", userName);
cookie.Values.Add("token", accRow.sToken);
Response.Cookies.Add(cookie);
Response.Redirect("default.aspx");
            }
else
            {
Response.Redirect("Useraccount.aspx");
            }
        }
```

Register.aspx

This Page has functionality of creating New Users, this page contains sign up form and inserts into database. And also send mail to user's given email address to confirm email address. 32 byte random string is generated and inserted against every user, also embedded in URL (get method) which is sent to user for confirming email.

Random String generation:

```
RandomNumberGenerator random = RandomNumberGenerator.Create();
byte[] token = newbyte[32];
random.GetBytes(token);
stringtokenNumber = BitConverter.ToString(token);
```

ConfirmMail.aspx

This page contains only message that Email has been sent, and Message of thanks for being part of Live Stream Box.

Confirm.aspx

This page is has no frontend functionality, and redirected from link sent to user for confirming email. The functionality of this page is to extract argument from URL (32 byte random) and matches in database and confirms that user’s email is valid.

```
string token= Request.QueryString["id"].ToString();
using (LiveStreamBox.StreamBoxDataContext dc = newStreamBoxDataContext())
            {
UserDataInfo user = newUserDataInfo();
AccountInfoactInfo = newAccountInfo();

varuserRow = dc.AccountInfos.FirstOrDefault(x =>x.sToken == token);
if (userRow.Equals(null))
                {
//Response.Redirect("regit");
                }
else
                {
stringsUserName = userRow.sUserName;

HttpCookie cookie = newHttpCookie("LiveStreamBox");
cookie.Values.Add("userName", sUserName);
cookie.Values.Add("token", userRow.sToken);

Response.Cookies.Add(cookie);
Response.Redirect("default.aspx");
                }
            }
```

MobieSignIn.aspx

This page is no frontend, and only redirected from mobile when a user is signing in to Live Stream Box. Username and Password is embedded using post method and extracted at MobileSignIn.aspx and matched to database, if valid returns to mobile true else false.

```
using (LiveStreamBox.StreamBoxDataContext dc = newStreamBoxDataContext())
            {
UserDataInfo user = newUserDataInfo();
AccountInfoactInfo = newAccountInfo();
varuserRow = dc.AccountInfos.FirstOrDefault(x =>x.sUserName ==
userName&&x.sPassword == userPass);
if (userRow == null)
               {
Response.Write("F%");
               }
else
               {
Response.Write("T%");
               }
            }
```

StopStream.aspx

This page is also redirected from mobile when one user stops streaming. This page request contains post argument by which user is identified and then changes status of Stream On to Stream OFF.

MS SQL Server 2008

For the implementation of Web Service we required a database for which we used SQL Server 2008.

Microsoft SQL Server, developed by Microsoft, is a relational database management system. It is a software product as a database and isprime function is to store and retrieve data as requested by other software applications.

This is also product of Microsoft and used with ASP.net. Our database contains different tables for different records…

LINQ

Language Integrated Query (LINQ) is a programming model that introduces queries as a first-class concept into any Microsoft .NET language. However, complete support for LINQ requires some extensions in the language used. These extensions boost productivity, thereby providing a shorter, meaningful, and expressive syntax to manipulate data.LINQ is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages.We fetched data from database using this component. LINQ is simpler to use comparing ADO.net.

IIS

IIS (Internet Information Server) is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server)

IIS is Microsoft's entry to compete in the Internet server market that is also addressed by other Companies and Web servers. With IIS, Microsoft includes a set of programs for building and administering Web sites, a search engine, and support for writing Web-based applications that access databases. IIS helps to host server side web application, as we used ASP.net which is server side language, & needs to be hosted by server.

Chapter 6: Testing and Results Analysis

6.1 Functional Testing (Black box)

The software program or system under test is viewed as a "black box". The selection of test cases for functional testing is based on the requirement or design specification of the software entity under test. Functional testing emphasizes on the external behavior of the software entity.

Test case ID: A 001
Test Description: Project overall testing
Date: 16/ 05/ 12
Environment: Android app installed, Darwin Server running on Window OS, IIS &Mysql running
Test Execution:
1. Run Darwin streaming server.
2. Open Live StreamBox app.
3. Logged in on web server
4. Select Tag.
5. Start Streaming.
6. Browse website.

Expected Result: video live stream coming on website
Actual Result: error**:** "Can't open stream"
Status: Failed

Test case ID: A 002 Test Description: Project overall testing
Date: 16/ 05/ 12
Environment: Android app installed, Darwin Server running on Window OS, IIS &Mysql running
Test Execution:
1. Run Darwin streaming server.

2. Open Live StreamBox app.

3. Logged in on web server

4. Select Tag.

5. Start Streaming.

6. Browse website.

Expected Result: video live stream coming on website

Actual Result: Up loader stream coming live

Status: PASS

6.2 White Box Testing (white box):

The software entity is viewed as a “white box”. The selection of test cases is based on the implementation of the software entity**,** on module by module basis.

Authentication A:

1. **Test case ID: A 001**

   Test Description: Check mobile connection with the web server

   Date: 16/ 06/ 12

   Function to be tested: executeHttpPost()

   Environment: Android Mobile

   Test Execution:

   1. Start Live StreamBox application

   2. Enter username and password

   3. Press login button.

   Expected Result: Status: Online

   Actual Result: **Connection not Established**

   Status: FAIL

2. **Test case ID: A 002**

   Test Description: Validates user

Date: 16/ 06/ 12

Variable to be tested: RESPONSE is “T”

Environment: Android Mobile

Test Execution:

1. Start Live StreamBox application
2. Enter username and password
3. Press login button.

Expected Result: Status: Online

Actual Result: **Sorry!! Incorrect Username or Password**

Status: FAIL

3. **Test case ID: A 003**

Test Description: Validates User

Date: 16/ 06/ 12

Variable to be tested: RESPONSE is “F”

Environment: Android Mobile

Test Execution:

1. Start Live StreamBox application
2. Enter username and password
3. Press login button.

Expected Result: Status: Online

Actual Result: **Online**

Status: PASS

4. **Test case ID: A 004**

Test Description: Tag Video

Date: 17/ 06/ 12

Variable to be tested: RES is “T”

Environment: Android Mobile

Test Execution:

1. Select tagging option from interface

2. Pres button “TAG”

Expected Result: notification “Tagging not Done”

Actual Result:Tagging Done”

Status: FAIL

5. **Test case ID: A 005**

   Test Description: Tag Video

   Date: 17/ 06/ 12

   Variable to be tested: RES is “T”

   Environment: Android Mobile

   Test Execution:

   1. Select tagging option from interface

   2. Pres button “TAG”

   Expected Result: notification “Tagging Done”

   Actual Result: Tagging Done”

   Status: PASS

6. **Test case ID: A 006**

   Test Description: H.264 support test

   Date: 26/ 01/12

   Variable to be tested: dummy video parsing

   Environment: Android Mobile

   Test Execution:

   1. Prepare Media Player

   2. Record Dummy video

   3. Parse it and get encoding parameters

   Expected Result: Test successful, H264 Supported

   Actual Result: H264 not supported

   Status: FAIL

7. **Test case ID: A 007**

   Test Description: H.264 support Test

   Date: 26/ 01/ 12

   Variable to be tested: dummy video parsing

   Environment: Android Mobile

   Test Execution:

   1. Prepare Media Player

   2. Record Dummy video

   3. Parse it and get encoding parameters

   Expected Result: Test successful, H264 Supported

   Actual Result: H264 supported

   Status: PASS

8. **Test case ID: A 008**

   Test Description: set media attributes and prepare media recorder

   Date: 15/ 11/ 11

   Fucntion to be tested: prepare()

   Environment: Android Mobile

   Test Execution:

   1. Set audio attributes

   2. Set video attributes

   3. call prepare function of media recorder

   Expected Result: media prepared for recording

   Actual Result: Prepare failed, cant stream video

   Status: FAILED

9. **Test case ID: A 009**

   Test Description: set media attributes and prepare media recorder

   Date: 15/ 11/ 11

   Fucntion to be tested: prepare()

Environment: Android Mobile

Test Execution:

1. Set audio attributes

2. Set video attributes

3. call prepare function of media recorder

Expected Result: media prepared for recording

Actual Result: media prepared

Status: PASS

6.2.1 Media Quality Tests:

No	**Frames per second**	**Delay if Resolution** 640x480	**Delay if Resolution** 320x240	**Delay if Resolution** 176x144
1.	10	7 secs	6 secs	4 secs
3.	20	9 secs	7 secs	5.25 secs

6.2.2 Session Hanlding B

1. **Test case ID: B 001**

   Test Description: running Darwin server

   Date: 16/ 09/ 11

   Environment: Window OS

   Test Execution:

   1. Run perl file

   2. open server in browser

   3. At admin page, enter username password

   4. main page appears

   Expected Result: status “Server running”

   Actual Result: connection failed or timeout

   Status: Failed

6.2.3 Webservice C

1. **Test case ID: C 001**

   Test Description: logging web application

   Date: 16/03/ 12

   Environment: Window OS

   Test Execution:

   1. Browse website

   2. Register yourself

   3. At admin page, enter username password

   4. main page appears

   Expected Result: status “Server running”

   Actual Result: connection failed or timeout

   Status: Failed

Chapter 7: Conclusion and Future Work

Live Stream Box is an android live streaming application using android built in encoder for encoding purposes so quality of streaming in terms of video quality and delay can be improved a lot by implementing efficient encoding algorithms by ourselves.

In Android, Google introduces H.264 AVC codec. H.264 is higher quality but consumes more uploading bandwidth as well as more phone and server power. The delay could be reduced near to phone processing time, which should become more negligible in the future as processors designed for cell phones increase in speed and efficiency.

A 3G and Pre 4G 3GPP Long Term Evolution, or LTE, is now being deployed in test cities by mobile service providers. LTE offers significant improvements in data transfer speed giving video streaming the ability to transfer high definition content from and to the phone but this technology is not yet present in Pakistan.

APPENDIX A: GLOSSARY

RTSP	Real Time Streaming Protocol
RTP	Real-time Transport Protocol
RTCP	Real-time Transport Control Protocol
EDGE	Enhanced Data Rates for GSM Evolution
DSL	Digital subscriber line
GSM	Global System for Mobile(communications)
MPEG	Motion Picture Experts Groups
VCR	Videocassette Recorder
3GPP	Third Generation Partnership Project
API	Application Programming Interface
TCP	Transmission Control Protocol
AVC	Advanced Video Coding
UDP	User Datagram Protocol

APPENDIX B: REFERENCES

J.M. Boyce and R. D. Gaglianello. Packet loss effects on MPEG video sent over the public Internet. In *Proc. ACM Multimedia*, September 1998

Derek Eager, Mary Vernon, and John Zahorjan. Minimizing bandwidth requirements for ondemand data delivery. In *Proc.* 5*th Inter. Workshop on Multimedia Information Systems*, October 1999.

Dmitri Loguinov and HayderRadha. Measurement study of low-bitrate Internet video streaming. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, November 2001. http://web.cs.wpi.edu/~claypool/courses/525-S02/slides/LR01.pdf

AXIS Communication. H.264 video compression standard. New possibilities within video surveillance.

Thomas Wiegand, Gary J. Sullivan, Senior Member, IEEE, GisleBjøntegaard, and Ajay Luthra, Senior Member, IEEE. In *Proc. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, VOL. 13, NO. 7,JULY 2003

Gary J. Sullivan, PankajTopiwala and Ajay Luthr. The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions. In *Proc. e SPIE Conference on Applications of Digital Image Processing XXVII Special Session on Advances in the New Emerging Standard: H.264/AVC*, August 2004

Dialogic Corporation. (2009). Mobile Video — A New Opportunity. http://www5.dialogic.com/products/docs/whitepapers/11296-mobile-video-wp.pdf

AXIS Communication. An explanation of video compression techniques. http://www.axis.com/files/whitepaper/wp_videocompression_33085_en_0809_lo.pdf

Tracy V. Wilson. How Streaming Video and Audio Work. *Introduction to How Streaming Video and Audio Work*. Retrieved June, 2011 from http://computer.howstuffworks.com/internet/basics/streaming-video-and-audio.html

Tracy V. Wilson. How Streaming Video and Audio Work. *Streaming Servers*. Retrieved June, 2011 from http://computer.howstuffworks.com/internet/basics/streaming-video-and-audio.html

Tracy V. Wilson. Introduction - How to Create Streaming Video. Retrieved June, 2011 from http://www.mediacollege.com/video/streaming/overview.html

Dmnelson. (Feb, 2011). Windows Media Server or Web Server?. Retrieved June, 2011 from http://learn.iis.net/page.aspx/454/windows-media-server-or-web-server/