SUNG KYUN KWAN UNIVERSITY

Software Engineering Technologies ECE5966

FINAL PROJECT REPORT

LiveShop

2019711010 Pham Cong Vinh

2019712410 Hyung-Joon Jeon

2013314340 Taekwon Ka

2018711276 Vo Van Vi

2019711011 Mwasinga Lusungu

Suwon

12th December 2019

Table of contents

I. REVISION HISTORY

Version	Date	Author	Revisions
0.1	2019.11.06	Team 1	System Requirements
0.3	2019.11.15	Team 1	Initiate the architecture
0.5	2019.11.28	Team 1	Architectural Drivers
1.0	2019.12.12	Team 1	Final project report

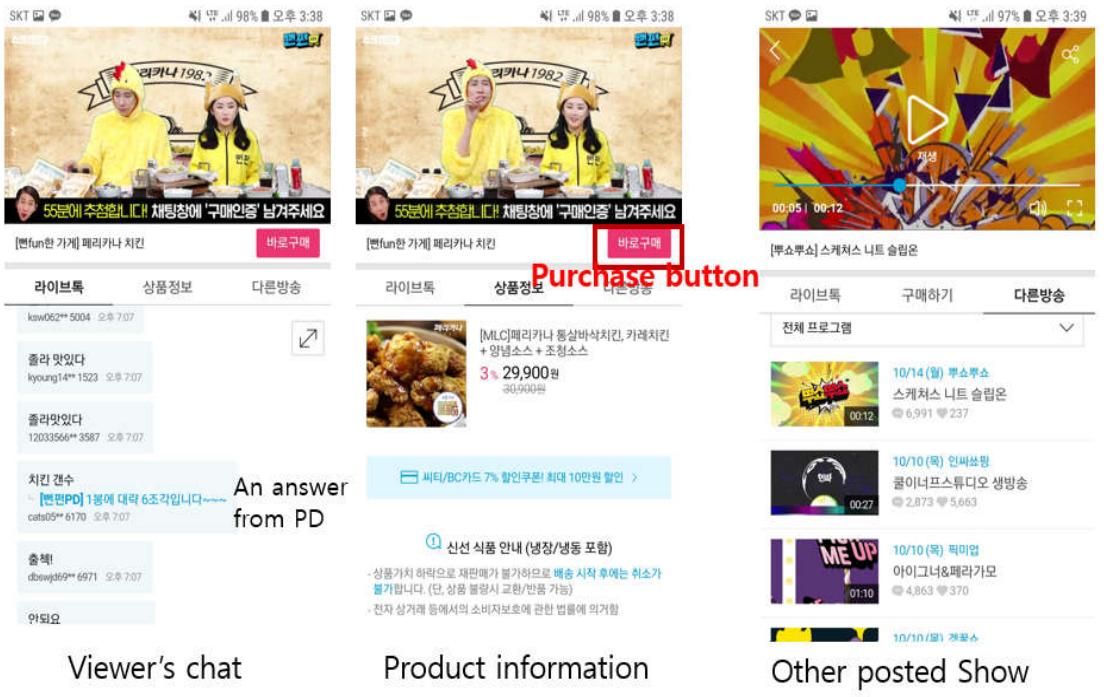
II. OVERVIEW OF THE TARGET SYSTEM

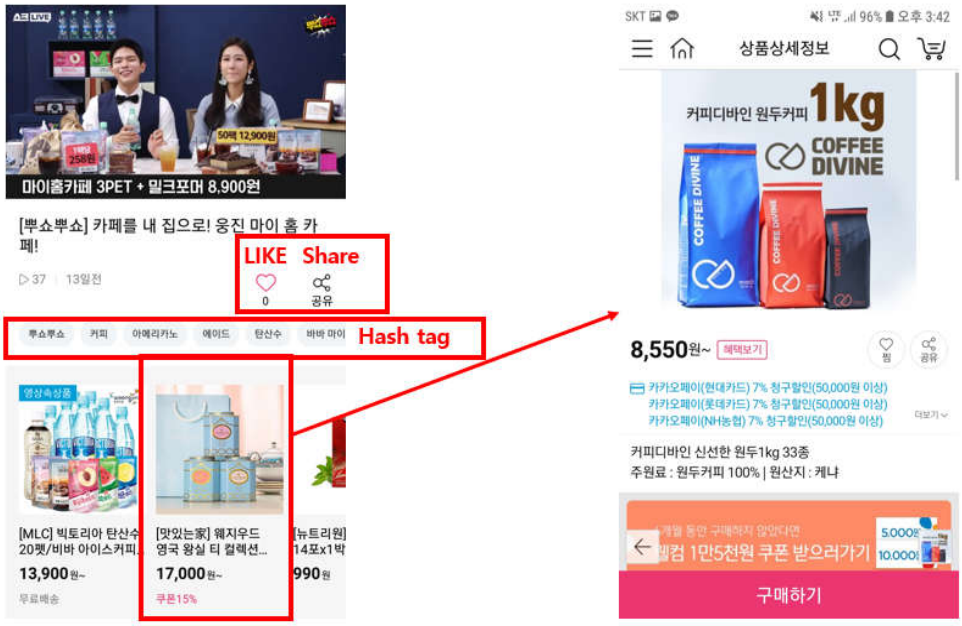
1. CJ Mall Shock Live

CJ Mall is one of the largest shopping malls in Korea, recording 124 billion won in operating profit in 2017. Shock Live is a live commerce platform of CJ Mall which is a combination of home shopping and live streaming. In the show, the show host (or MD) introduces the product to sell like home shopping. Since the show is live, the show host can receive viewers' feedback through online chat and have a conversation with viewers. The target customers of Shock Live are from 20s to 30s who are familiar with mobile devices. These people are highly influenced by influencer marketing, a marketing strategy in which a famous person on SNS (referred to by the term 'influencer') introduces product. Shock Live invites such 'influencers' as its show host to present a product to target customers.

An important feature of Shock Live is that sellers and customers are clearly divided. Like home shopping, Shock Live introduces only one product to customers at a certain time, as planned by the MD. As a result, a consumer can only see live streaming planned by the MD of CJ Mall and only buy the products that come out the show. So, the CJ Mall just becomes the seller and the end-users of Shock Live become customers.

After the show, video and chats are saved and posted with product information. They are used as advertising materials to introduce products. It is effective because, according to a survey, people are more likely to purchase the item when they watch video advertising related to the product rather than the related text.



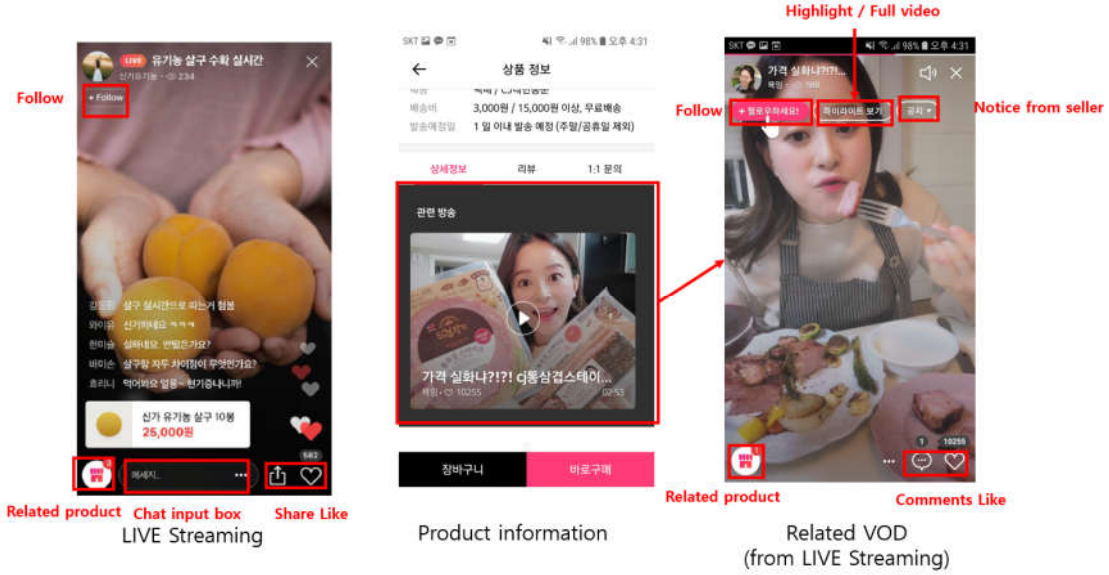


VOD and related products about VOD

Product information

2. Grip

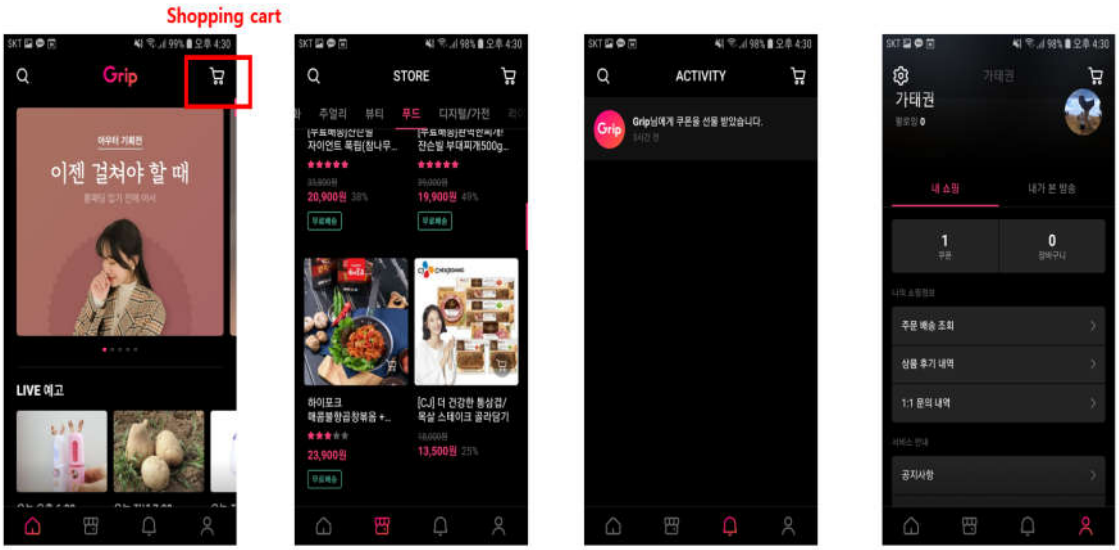
Grip is basically a Live commerce platform similar to CJ Mall's Shock Live. The difference between them is that Shock Live sells the product from CJ Mall through the show, whereas Grip is a sales platform



in which anyone can be a seller. In Grip, anyone who is registered as a seller can advertise the product through their own live streaming. As a result, unlike Shock Live, broadcast schedule and running times can vary depending on the seller.

Grip connects sellers and consumers. The administrator of Grip monitors the seller and the consumer on whether they are selling and buying properly in accordance with sales regulations.

Like Shock Live, video and chatting from live streaming are posted with product information. Specially, users in Grip can see the highlight videos specified by the seller as well as the full video.



Main page
(Live schedule and vod)

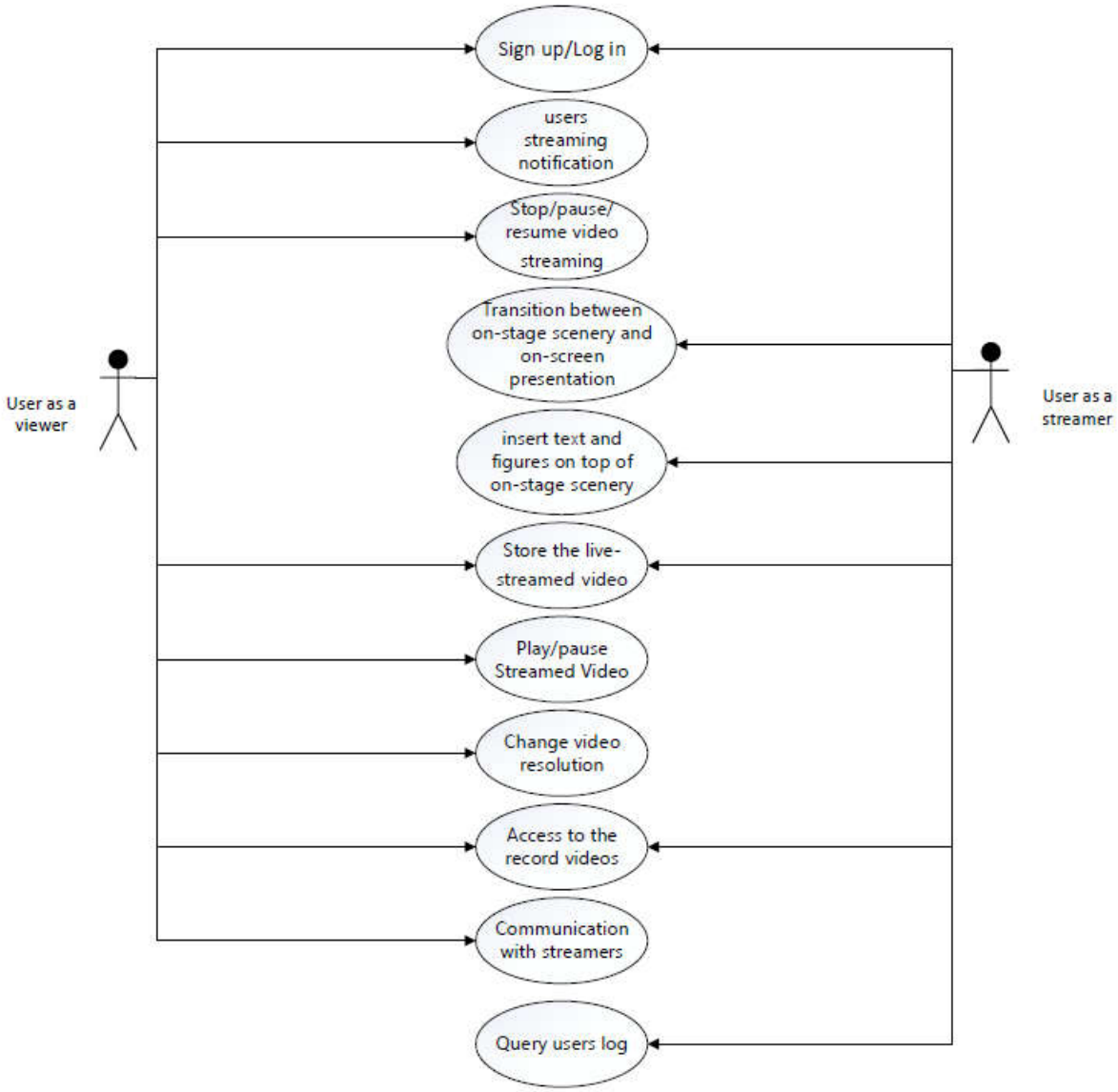
Store

My activity

My page

III. SYSTEM ARCHITECTURE

1. System requirements



Use Case	Description
UC-1: Start broadcasting Streamed Video and	The show host broadcasts the show via live streaming. As soon as a show starts, the users are notified about it, and notification procedure is to follow the constraint CON-1. While broadcasting, the host can see the show with

Notify online users about streaming	own mobile device to check if the show is properly being broadcast, as described in UC-3.
UC-2: Stop/pause/resume video streaming (Host side)	In case the show time expires, the host can have the utilities to stop video streaming. In case any technical problems occur, pausing is necessary, and resuming the show is also needed whenever such problems are resolved.
UC-3: Transition between on-stage scenery and on-screen presentation (or pre-recorded videos)	At on-stage scenery, the host gets to directly speak in front of off-line audience or online users watching the show. On the other hand, using on-screen presentation enables use of text and figures in slides (or perhaps pre-recorded videos) created from or loaded to digital devices such as computers. In many cases, the show host would want to effectively advertise a product by switching from on-stage scenery to on-screen presentation. UI for enabling this easily must also be given.
UC-4: insert text and figures on top of on-stage scenery	During the whole streaming session, text and figures will be utilized and put on top of on-stage scenery to display various information, especially the brief product descriptions and contact information (telephone number, email, etc.). UI for enabling this easily must also be given.
UC-5: Store the live-streamed video separately, and analyze (or selectively store) highlight scenes.	For further use of scenes from live-streamed advertisement, the host should be able to automatically save the streamed video. In addition, the highlights of the video must also be separately recorded. The host has two options; if set to “manual”, the host, while broadcasting, gets to choose which part of the video is the essential part that conveys core product information
UC-6: Play/pause Streamed Video (Client side)	Basic UI for playing or pausing the streamed video must be given. Doing this action should not have any malignant technical effects, as outlined in CON-2.
UC-7: Change video resolution	According to user’s preference (or perhaps change in network conditions), the user must be able to change the video resolution. Basically, if the resolution setting is not set to “auto”, the user is given a simple UI for enabling manual resolution changes. Meanwhile, change in network connection status can also affect the video resolution if the setting is set to “auto”. Details on how video resolution is automatically changed is described in QA-1.

UC-8: Access to the record videos	Users can access the recorded videos for a product anytime when they have missed the live streaming session.
UC-9: Communication with streamers	The system supports chatting. Viewers can interact with the streaming by texting comments while watching live-streamed advertisements.
UC-10: Sign up and log in	Users sign up or log in to the system, which is required for chatting, product purchase, or broadcasting their own advertisements.
UC-11: Query users log	Streamers queries time logs of their streaming to check all their schedule, their streaming duration or to make reports
UC-12: Chat mechanism	Audience will be chatting online with MD on the Player screen.

2. Quality Attribute

Quality Attribute	Category	Description	Associated Use Case
QA-1: Video Resolution	Performance	As far as limitations in hardware are negligible, available resolutions for video streaming are 240p, 360p, 720p, 1080p, and 4K. The user either gets to manually choose the resolution or select “auto” to watch the video with resolution automatically set according to hardware specifications and network conditions.	UC-7
QA-2: Low Latency between server (show host) and client (online audience) during live streaming	Performance	As the name of feature suggests, this streaming feature should be manifested in live manner; delays between show host and online audience should be as little as possible. To be specific, the latency between server side for broadcasting video packets and client side for receiving video packets should be no more than 5 seconds.	UC-2, UC-6

QA-3: Video Aspect Ratio adjustment based on device screen	Scalability	According to the screen resolution of user's device, the basic 16:9 ratio of the video being broadcast may not work properly. To cope with this, the video screen should be adjusted to fit the device screen.	UC-2, UC-6, UC-7
QA-4: Low latency between server and client in chatting	Performance	The latency between the timepoint of sending a message and displaying it on chatting screen should be no more than 1 second.	UC-9
QA-5: Synchronization between streaming and chatting	Performance	A time server management protocol will help to synchronize comments and display them on screen during streaming one by one.	UC-2, UC-6, UC-9
QA-6: Account management for advertisement and purchasing	Security	Although any users (even unregistered users) can view the products and their associated streamed videos, they must sign up / log in to purchase them. In addition, sellers must make their accounts to gain permission for streaming videos to advertise products.	UC-10
QA-7: Record video and store it with unconstrained access.	Availability	During the streaming, the video is to be recorded and securely stored without memory issues. Unless the broadcaster wants to delete or update the video, the lifetime of video being stored is unlimited, and users can access it whenever they want.	UC-8
QA-8: Notification of streaming session expiration	Performance	A new time server management protocol will be added to track the streaming duration of each seller. An alert will be shown when the streaming almost reaches the designated duration to notify the current seller to be aware.	UC-2
QA-9	Security	The system will obscure (replace) all personal information (sensitive) like phone number with	UC-12

		asterisk to prevent other users from using that information for malicious or other purposes	
QA-10	Performance	The system will display the posted comments within 500 msec (Assume that the users' network condition is good)	UC-12
QA-11	Usability	The system shall provide an option for users to translate foreign languages	UC-12

3. Constraints

Constraint	Description
CON-1: Support for iOS and Android	The system must be accessible through web browsers or a mobile application with different platforms: iOS and Android
CON-2: Reliable Storage	All streaming video must be recorded and securely stored in the database without issues in memory limitations.
CON-3: Open source	The system shall be composed primarily of open source technologies for cost efficiency. For components whose effectiveness of using proprietary technology outweighs costs, proprietary technology may be used.
CON-4: Flexibility in video resolution change	Viewers can watch streamed videos even in networks with low bandwidth or poor connection conditions, having video resolution adjusted to low configurations whenever option is set to "auto".
CON-5: Latency-free video play request	Once a viewer invokes play action for a live-streamed video, requested video must be played within 15 seconds.
CON-6	The system shall store all comments in the database for analysis and shall discard them after 30 days.

4. Concerns

Concern	Description
CRN-1	Establishing an overall initial system structure
CRN-2	Choose a Programming language among various options such as JavaScript, python, HTML+CSS, etc., based on the team members' knowledge.
CRN-3	Allocate work to members of the development team
CRN-4	Leverage the team's knowledge about video streaming, e-commerce platform and ADD 3.0.

IV. FIRST ITERATION - System Architecture

Here we describe the first iteration of the ADD applied to the LiveShop project.

1. Step 1: Review Input

This is the first step at First Iteration so the input is the LiveShop's requirements had been presented.

2. Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to establish an overall system architecture. Here we consider complete system as an input to the selection of drivers shown in the following table.

Selected Drivers	Architecture concerns: CRN-1, CRN-3

3. Step 3: Choose One or More Elements of the System to Refine

In this step, we choose which element of the system will be the design focus in subsequent steps shown in the below table.

1. We firstly carefully examine the example CJmall Live in order to summarize which function, feature, quality level that we should achieve in implementing our project.
2. Then we considered team member knowledge in developing mobile application, client-server model, working ability in order to decide which concept architecture we should follow as design concept.

To the best of our knowledge, the client-server is the most suitable to our project. That means the Model–view–controller (MVC) that is better suited for our LiveShop as a web application. However, our all our team members didn't have background in web, network and mobile application. It turned out the MVC is quite difficult to us in order to adapt to.

This led us to other choice, the "Action Domain Responder" architectural pattern. Action Domain Responder is an alternative to the "Model 2" misappropriation (for server-side over-the-network request/response interfaces) of the original MVC user interface pattern (for client-side in-memory graphical user interfaces). ADR is a user interface pattern specifically intended for server-side applications operating in an over-the-network, request/response environment.

Design decisions and location	Rationale and assumptions
Action Domain Responder **A server-side alternative to Model View Controller.**	ADR more closely describes the current day-to-day practice and work of web interactions than "Model 2" MVC. A request comes in and gets dispatched to an action; the action interacts with the domain and emits a response. The response work, including both headers and content, is cleanly separated from the input collection and the domain logic. One drawback is we end up with more classes in the application. This drawback may not be so terrible in the longer term. Individual classes may lead to cleaner or shallower inheritance hierarchies. It may also lead to better testability of the Action separate from the Responder. These will play themselves out differently in different systems. Others have noted many classes may be more easily manageable via IDEs and editors than fewer classes but more methods since class lookups are frequently easier than method lookups. Another benefit is that a clean separation of business logic into the Domain makes it easier to test the domain logic without spinning up an entire user interface system. Likewise, a clean separation of presentation logic makes it easier to test the response-building work in isolation from the input collection and domain operations.

4. Step 4: Determining the Architecture: Sketch views and record design decisions

After gathering requirements and base on the examination of CJmall Live, we specify some sketch views that represent some main use-cases as follows:

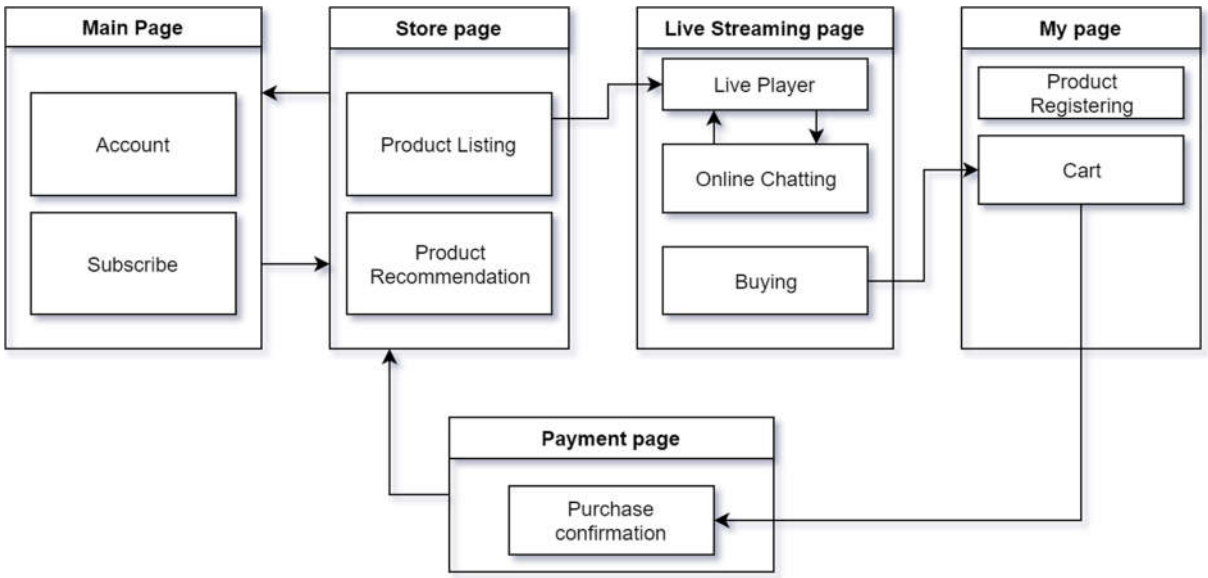
Main page: In main page, there are several Live streaming shows or recorded live streaming videos- based on the user's preferences.

Store page: In Store, there are several products registered by seller and introduced through Live Streaming.

Live streaming player page: In this page, audience watch the live show, online chat with MD, follow seller or buy the product.

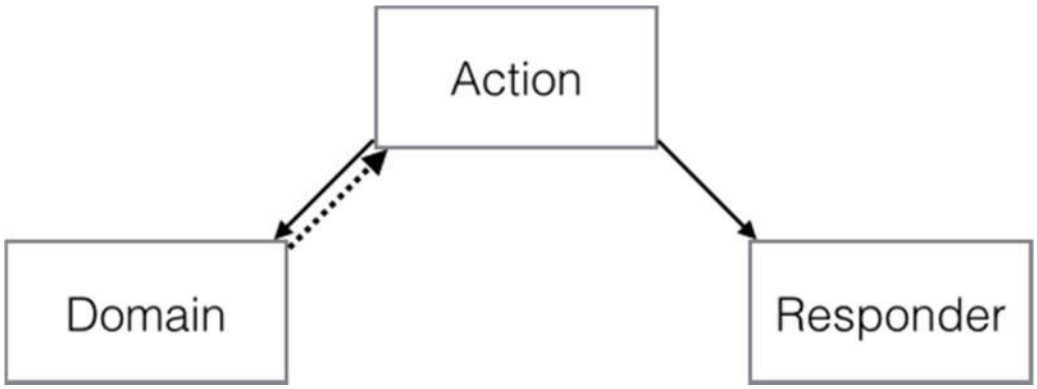
My page: In My Page, users can check information about purchased products and set options related to the application.

The diagram shown in Figure. 10, displays the visual and functional components of each page, and highlights the primary actions available to the user in order to navigate through the site to complete a purchase.



The diagram shown in Figure below, displays the visual and functional components of each page, and highlights the primary actions available to the user in order to navigate through the site to complete a purchase.

Determining the Architecture



Action Domain Responder organizes a single user interface interaction between an HTTP client and a HTTP server-side application into three distinct roles.

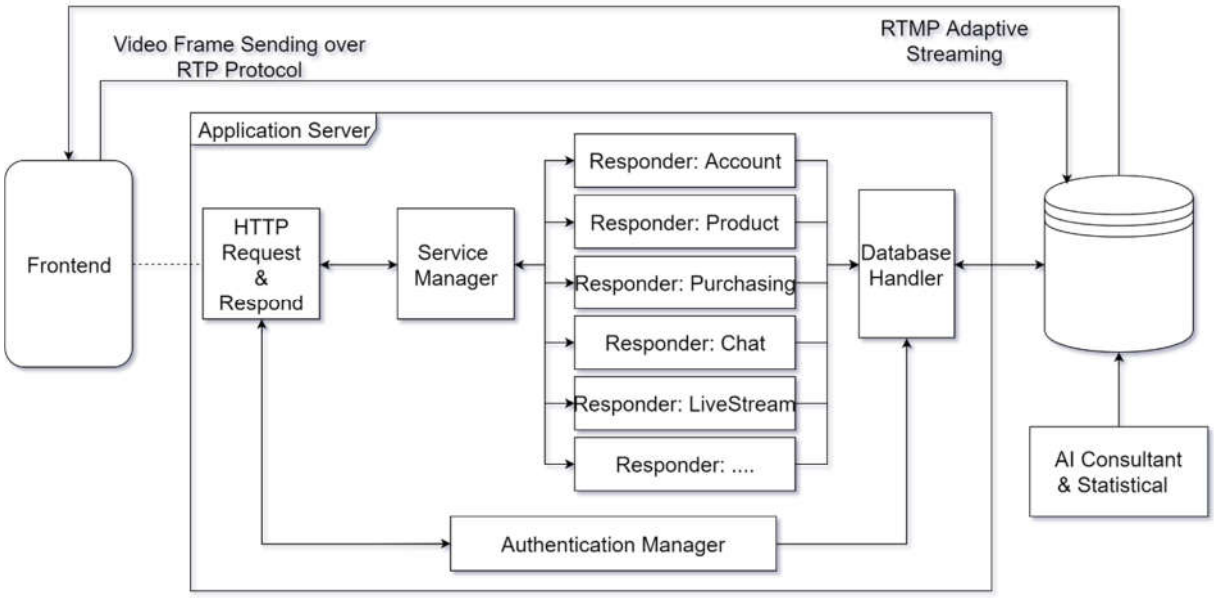
- Action is the logic to connect the Domain and Responder. It invokes the Domain with inputs collected from the HTTP Request, then invokes the Responder with the data it needs to build an HTTP Response.
- Domain is an entry point to the domain logic forming the core of the application, modifying state and persistence as needed. This may be a Transaction Script, Service Layer, Application Service, or something similar.
- Responder is the presentation logic to build an HTTP Response from the data it receives from the Action. It deals with status codes, headers and cookies, content, formatting and transformation, templates and views, and so on.

Collaborations

1. The web handler receives an HTTP Request and dispatches it to an Action.
2. The Action invokes the Domain, collecting any required inputs to the Domain from the HTTP Request.
3. The Action then invokes the Responder with the data it needs to build an HTTP Response (typically the HTTP Request and the Domain results, if any).
4. The Responder builds an HTTP Response using the data fed to it by the Action.
5. The Action returns the HTTP Response to the web handler sends the HTTP Response.

5. Step 5: Tabulation of the Sketch views and record design decisions

Figure below shows the sketch views and record design decisions of our “Action Domain Responder” architecture.



The table tells about each component and their responsibilities.

Components	Responsibilities
HTTP Request & Respond	This component will parse the HTTP request by client-side mobile application to Actions (and their attached data), then invoke the Service Manager (Domain) in order to feedback to these requests.
Service Manager	This is the core of Server-side application, it plays a role as Domain (the entry point to the business logic).
Responders	Each responder corresponds a logic that will handle a kind of request. Also, each responder will be constructed as a server-side application programming class.
Database Handler	Provide the database query operations to Responders.
Authentication Manager	Authenticate the Action (request) by its credentials before allow it correspond responder operate with Database.
AI Consultant	Mining the data generated by user in order to provide informative resource.

6. Step 6: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Partially addressed	Completed addressed	Design decisions made during the iteration
CRN-1		Replacing MVC by ADR design pattern, avoiding complexity of MVC.
	CRN-3	Leader will take care the initial architecture and technologies choice. Team 1 will focus on Video Streaming user cases. Team 2 who had network background will focus on Online Chat / Commenting user cases.
QA-6		Define “Authentication Manager” component in architecture, this will be place-holder for iterations that will use security QA, CON, CRN as architecture driver.
	QA-2	Each Action will be serviced by its corresponding Responder, this will reduce the latency for overall experience.

V.SECOND ITERATION - Selection of Technologies

Technology choices often influence the system architecture, meaning that we need to select technologies at the earliest stages of architecture design. There are 3 components that construct the live stream function which are:

1. Publisher (Recording video & packet sending)
2. Server (Encode video, Storing & streaming)
3. Viewer (Playing & Interacting with user)

Because the streaming function is the main key function of our project, we first select technologies that allow us to quickly implement the streaming function. Other technologies for mobile application constructing, chatting, web shopping, payment… will be decide at following iteration later.

2. Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to address CRN-1 (Establishing an overall initial system structure), also we must keep in mind the CRN-2 (Choose a Programming language based on the team members' knowledge) and CRN-4 (Leverage the team's knowledge).

3. Step 3: Choose One or More Elements of the System to Refine

Because the CJmall Live is closed-source application, we couldn't refer to it in term of technology. So, we have gathered some technologies, open source components which categorized into above 3 categories. Then we examined them in term of their feature, complexity, modification and combination ability… in order to select appropriate ones.

There are three areas that basically cover the live streaming architecture as mentioned in the beginning of this section. There are several technologies for this system, we can list some of them and discuss which one is most relevant for our project.

4. Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

The table below provides some discussion about technologies used for live stream systems.

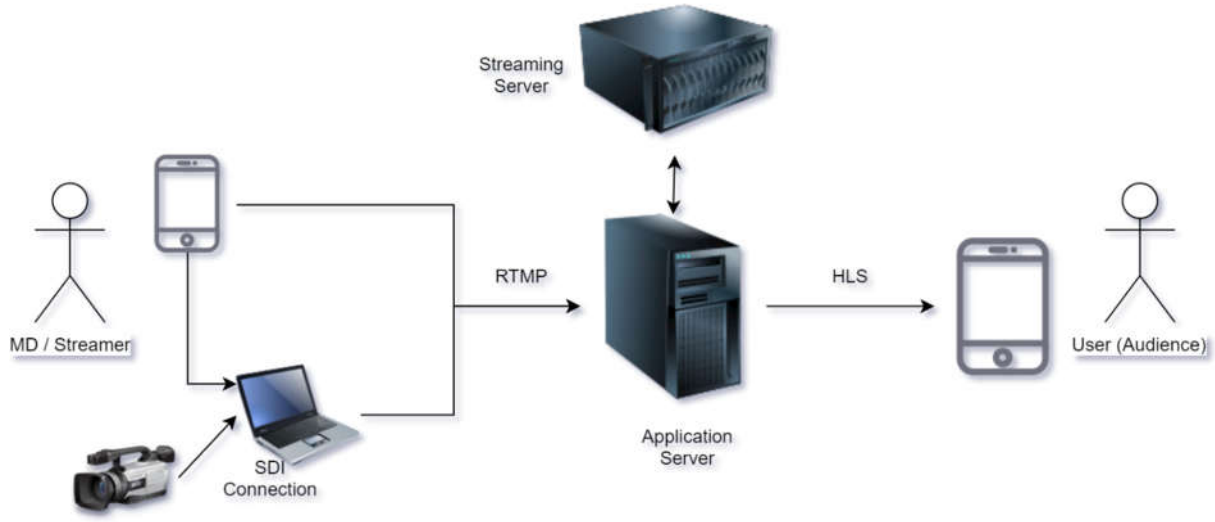
Design Decisions and Location	Rationale and Assumptions
Select the publisher technology for encode the live stream video	H.264 is a well-known video compression standard for high-definition digital video. The purpose of creating H.264/AVC was to pioneer a new digital video standard capable of delivering good video quality at a substantially lower bitrate. Most newer encoders today are compatible with this protocol and suitable for live streaming with compact files. In our case, particularly we want to stream the best, highest-quality video content, we will use this video codec as the format for recording, encoding & storing as well as streaming.
Select the technology for streaming server	RTMP is a TCP-based protocol which maintains persistent connections and allows low-latency communication. To deliver streams smoothly and transmit as much information as possible, it splits streams into fragments, and their size is negotiated dynamically between the client and server. We will employ this protocol to send the video content recorder by user (as streamer) to LiveShop streaming server.
Select the technology for the Viewer	HTTP Live Streaming (also known as HLS) is an HTTP-based adaptive bitrate streaming communications protocol developed by Apple Inc. and released in 2009. Support for the protocol is widespread in media players, web browsers, mobile devices, and streaming media servers. An annual video industry survey has consistently found it to be the most popular streaming format. HLS resembles MPEG-DASH in that it works by breaking the overall stream into a sequence of small HTTP-based file downloads, each download loading one short chunk of an overall potentially unbounded transport stream. A list of available streams, encoded at different bit rates, is sent to the client using an extended M3U playlist. To enable a player to adapt to the bandwidth of the network, the original video is encoded in several distinct quality levels. The server serves an index, called a "master playlist", of these encodings, called "variant streams". The player can then choose between the variant streams during playback, changing back and forth seamlessly as network conditions change. LiveShop mobile application will communicate with streaming server over this protocol and fetch the video data and playback to user (as audience).

5. Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The instantiation design decisions considered and made are summarized in the following table:

Decision	Rationale
Recorder & Packet Sender 	The Recorder module and Packet Sender module will be developed and embedded inside LiveShop mobile application. By leveraging the OpenCV and FFMPEG, these 2 highly extensible and open technologies (CON-3) will allow us implement sophisticated user-case (UC-2, UC-3, UC-4) with defined quality attributes (QA-3).
Streaming Server 	The Streaming Server will receive the video data from streamer then encode them into H.264 format video and store at storage in order to distribute audience user. Constructing an initial stream server will be overloaded to us. Because of that we will employ the Ant Media Server which is an open-source stream server (CON-3). Ant Media Server features Low Latency Adaptive Live Streaming as well as provide SDK for iOS, Android (CON-1).
Player 	hls.js is a JavaScript library which implements an HTTP Live Streaming client. hls.js does not need any player, it works directly on top of a standard HTML<video>element. The Player module will be implemented based on hls.js and embedded inside LiveShop mobile application. By playback the content from stream server to user's mobile device by HLS protocol which support the bitrate adaptability, we can ensure QA-2 (Low Latency) as well as UC-7 (Changing resolution) and QA-1 (Multiple Video Resolution).

6. Step 6: Sketch Views and Record Design Decisions



7. Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Partially addressed	Completed addressed	Design decisions made during the iteration
UC-2		By constructing the recorder module base-on open source library by ourselves, we can implement the Pause/Start function.
UC-3		Playing recorded videos while streaming will be implemented simulate OBS's features (open source).
	UC-4	OpenCV and FFMPEG allow developer freely manipulate the recording content, we will
	UC-7	The video supplied by streamer will be encode to multiple resolution versions. Audience can freely select resolution appropriate to their device.

QA-3		HLS protocol will automatically be changing the resolution video base on audience's internet bandwidth and device capability.

VI. THIRD ITERATION – Refining elements for Video streaming functionality

The drivers for this iteration are related to live broadcasting of the show to prospective customers. Continuing with the progress made until the second iteration, the aim of this iteration procedure is to add and refine elements related to video streaming functionality. In addition, this iteration deals with manifesting architectural constraints and concerns related to such functionality.

2. Step 2: Establish Iteration Goal by Selecting Drivers

In this iteration, we focus on services related to reliable video streaming.

Quality Attribute	Category	Description	Associated Use Case
QA-1: Video Resolution	Performance	As far as limitations in hardware are negligible, available resolutions for video streaming are 240p, 360p, 720p, 1080p, and 4K. The user either gets to manually choose the resolution or select "auto" to watch the video with resolution automatically set according to hardware specifications and network conditions.	UC-7
QA-2: Low Latency between server (show host) and client (online audience) during live streaming	Performance	As the name of feature suggests, this streaming feature should be manifested in live manner; delays between show host and online audience should be as little as possible. To be specific, the latency between server side for broadcasting video packets and client side for receiving	UC-2, UC-6

		video packets should be no more than 5 seconds.	
QA-3: Video Aspect Ratio adjustment based on device screen	Scalability	According to the screen resolution of user's device, the basic 16:9 ratio of the video being broadcast may not work properly. To cope with this, the video screen should be adjusted to fit the device screen.	UC-2, UC-6, UC-7

3. Step 3: Choose One or More Elements of the System to Refine

The elements related to video streaming service are chosen for refinement.

4. Step 4: Choose One or More Design Concepts that Satisfy the Selected Drivers

Live streaming system is a greenfield system for a mature domain. So, in this iteration, we choose design concepts from existing software solutions related to video streaming to satisfy drivers such as performance, reliability and scalability.

Refinement for	Service provider Name	Coverage (Cost| License |Available)

Live Streaming	Ant Media	$49 per month | Enterprise edition | mobile OS including Android, IOS
	Open Broadcaster	Free | GNU | Web Based (Linux)
	Plex	Free | Freemium & GPLv2 | All OS
	VLC media Player	Free | GNU | All OS

5. Step 5 & 6: Instantiate Architectural Elements, Allocate Responsibilities, Define Interfaces and Sketch Views and Record Design Decisions

<table>
<tr><th colspan="2">Design Decisions and Location</th><th>Rationale and Assumptions</th></tr>
<tr><td colspan="2">Use Ant Media Enterprise Edition as a server for video streaming.</td><td>SDK for iOS, Android in ant media makes it easy to develop application where users can broadcast and watch. Ant media also has low latency with 0.5 seconds End-to-End latency. Cluster supporting for scaling and adaptive bitrate are advantageous for scaling.</td></tr>
<tr><td></td><td>Alternative</td><td>Reason for Discarding</td></tr>
<tr><td></td><td>Open Broadcaster</td><td>Open Broadcaster is available for PC. Because It support for Windows, Linux and mac OS. If we choose Open Broadcaster as live streaming system, we'll need to modify several elements to suit the mobile OS.</td></tr>
</table>

	VLC media Player	VLC media Player doesn't support for live streaming server. So, if we use VLC media Player, we will need our own server.
	Plex	Plex offers live streaming service through its own mobile applications. So Plex is not suitable for applying to our application.

6. Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		QA-1, UC-7	Ant Media Server is always aware of the network speed, the end-user has on his side. Regardless of the resolution in the Adaptive settings, a bitrate selection is made either upward or downward, depending on the bit rate information and the instantaneous network speed.
		QA-2	Ant media use WebRTC as video streaming protocol which is more expensive than other protocols but has a lower latency of less than a second. Ant media enterprise edition has low End-to-End latency less than 0.5 seconds.

QA-3			Embedded player in Ant media can't scaling video automatically according to user's device. So, we need to address this feature later.
		UC-1	The ClipBucket provides commenting/ Chatting feature during the video streaming
		CON-1	Our database server will store data within 30 days. The data later than 30 days will be deleted
	QA-1		There is a security using in ClipBucket but it does not fully satisfy the requirement

		QA-2	ClipBucket uses advance AJAX techniques for the Comments & Ratings and in ClipBucket commenting system we can reply to comments & Like/Dislike them As well. This technique makes the comments posting really fast
		QA-3	ClipBucket is multilingual. We can generate a language pack with just one click.

VII. FOURTH ITERATION – Commenting

This section enables commenting feature for users who want to solicit additional information or review products based on their experiences. This is one of the core processes for the system to make it interactive between show hosts and viewers (potential customers). In this section, we provide the feature which improves the system's reliability, usability, convenience for the viewers.

2. Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration is to add the commenting feature to the system by leveraging the team's knowledge in terms of programming languages and other technologies. We will combine QA-9, QA-11, CON-6 along with QA-10 to augment the feature.

Selected Drivers	**Quality attributes**	**QA-9, QA-10, QA-11**
	Constraints	CON-6

3. Step 3: Choose One or More Elements of the System to Refine

In this section, we focus on refining the element that are related to the chat controller of the system which helps the viewers to interact with the show hosts during the streaming section.

4. Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

This section outlines some of the design concepts that we have considered during the development of this architecture. Performance, usability, and reliability quality attributes were adopted as the baseline criteria for selecting tactics used in the design process of this architecture.

In the table below, we list some service providers that we choose for refinement purposes. These service providers are open source so that we can add more functions to the commenting features that were defined in the previous architectural drivers section.

Refinement for: Commenting during video streaming	Name of Service provider	Package feature
	Red5 Open source media server	Free | Open source
	MistServer Open Source	Free | Open source | Supported on all major OS
	Kurento Media server	Free | Open source | Supported on all major OS and Mobile platforms
	Clipbucket	**Free | Open source | Supported on all major OS and Mobile platforms**
	Plex Media Server	Free | Open source | can be used with Web applications and Mobile platforms

5. Step 5: Instantiate Architectural Elements, Allocate Responsibilities and Define Interfaces

We adopt one of the open source platforms which were listed in the table above (step 4). The basic feature of commenting – satisfying the performance and usability of quality attribute - is based on this platform (summarized in the following table)

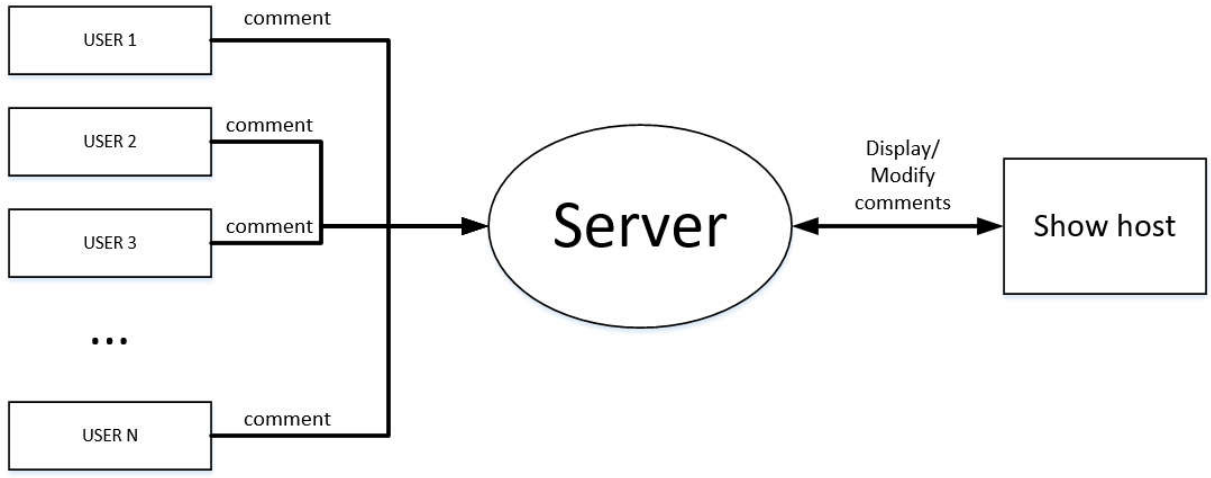
Design decisions	Rationale and Assumptions
Clipbucket is 100% Open source & Free, simply download/Install it. We will use it as a video streaming server	ClipBucket is an OpenSource Multimedia Management Script which lets us manage Videos, Photos & Audios from one platform. We can create websites or share websites like Youtube,

	Metacafe, or any other. This platform offers internal management systems that allow users to communicate and send messages via a built-in messaging service. Clipbucket Video streaming server offers advanced modules like FFMPEG to make on fly Video Conversions. So, users can stream it straight away using HTML 5 Players.

By leveraging the ClipBucket platform, we will create more functions on commenting to make the system satisfy with reliability of selected quality attributes. We will develop the features that could asterisk some personal information.

6. Step 6: Sketch Views and Record Design Decisions

The figure below depicts an overview of the commenting feature. During a livestream session, users can comment using a textbox at the bottom of the user interface window. The comments will be displayed in real-time based on the QA-10. As for the show host, he/she will be able to view or modify the comments depending on their nature.



7. Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not Addressed	Partially Addressed	Completely Addressed	Design Decisions Made During Iteration
		UC-12	The Clipbucket provides commenting/ Chatting feature during the video streaming
		CON-6	Our database server will store data within 30 days. The data later than 30 days will be deleted
	QA-9		There is a security using in ClipBucket but it does not fully satisfy the requirement
		QA-10	ClipBucket uses advance AJAX techniques for the Comments & Ratings and in ClipBucket commenting system we can reply to comments & Like/Dislike them as well. This technique makes the comments posting really fast
		QA-11	Clip Bucket is multilingual. We can generate a language pack with just one click.

VIII. SUMMARY OF THE ARCHITECTURE

The overall architecture design is based on the principle of Action Domain Responder architecture. The decision to adopt this design principle was necessitated by the need to separate components in order to eliminate the possibility of single point of failure instances.

Action Domain Responder is an alternative to the "Model 2" misappropriation (for server-side over-the-network request/response interfaces) of the original MVC user interface pattern (for client-side in-memory graphical user interfaces). ADR is a user interface pattern specifically intended for server-side applications operating in an over-the-network, request/response environment. Aligning expectations and factoring concerns away from the modern derivations of "Model 2" MVC toward Action Domain Responder is not difficult.

In general, the architecture is composed of the following components:

1. Domain (versus Model): There are few if any significant differences here, other than the Responder does not interact with the Domain. The Responder might use domain objects like entities and collections, perhaps wrapped in a Domain Payload, but only for presentation purposes. It does not request new information from the domain or send information back to the domain. Thus, the main difference is in the name. Using the word Domain instead of Model is intended to make implementors think of PoEAA domain logic patterns such as Service Layer and Transaction Script, or domain-driven design patterns such as Application Service or Use Case.

2. Responder (versus View): In a "Model 2" MVC system, a Controller method will usually generate body content via a View (e.g., a Template View or a Two Step View). The Controller then injects the generated body content into the response. The Controller action method will manipulate the response directly to the status code, headers, cookies, and so on. Some Controller action methods may present alternative content-types for the same domain data. Because these alternatives may not be consistent over all the different methods, this leads to the presentation logic being somewhat different in each method, each with its own preconditions. However, in a server-side web application, the presentation being delivered as output is not merely the body of the HTTP response. The presentation is the entire HTTP response, including the HTTP status, headers, cookies, and so on. As such, doing any sort of HTTP response building work in a Controller indicates a mixing of concerns. To fully separate the presentation logic, each Action in ADR invokes a Responder to build the HTTP response. The Responder is entirely in charge of setting headers, setting the body content, picking content types, rendering templates, and so on.

3. Action (versus Controller): In common usage, most "Model 2" MVC Controller classes contain several methods corresponding to different actions. Because these differing action methods reside in the same Controller, the Controller needs additional wrapper logic to deal with each action method properly, such as pre- and post-action hooks. Additionally, different action methods may have different dependencies, leading to over-long constructors and/or attempts at "action injection" of dependencies. In an ADR system, a single Action is the main purpose of a class or closure. Each Action would be represented by an individual class or closure. The Action interacts with the Domain in the same way a Controller interacts with a Model but does not interact with a View or template system. It sends data to the Responder and invokes it so it can build the HTTP response. All other logic, including all forms of input validation, error handling, and

so on, are therefore pushed out of the Action and into the Domain (for domain logic concerns) or the Responder (for presentation logic concerns).

In our initial architectural design, we adopted a very simple model which comprised a single apache2 server on which all services such as the video streaming RTMP. This architecture had various drawbacks including the issue of a single point of failure. In subsequent evolutions, we decided to solve the problem by adopting the attribute driven design (ADD). Applying ADD principles in design the architecture helped us to come up with a better architectural design.