TEAM PROJECT – 3RD TEAM

ARCHITECTURAL DESIGN OF ALIEXPRESS LIVE

Nov. 29, 2018

Team Members

Name	Email
Soolyun Kim	kimsuryeon@gmail.com
Dongmin Jang	ehdals338@gmail.com
Daegeun Choi	asdf123ggg@gmail.com
Minchang Choi	mraz.choi@gmail.com
Peiheng Lee	peihenglee1024@gmail.com
Yazhini Venugopal	yazhini.yazhl93@gmail.com

Revision History

Version	Date	Author	Revisions
0.1	2018.11.22	3rd Team	Extract overall architecture
0.2	2018.11.29	3rd Team	Iteration 1&2
0.4	2018.12.04	3rd Team	Iteration 3&4
0.7	2018.12.12	3rd Team	Iteration 5&6
1.0	2018.12.19	3rd Team	Iteration 7 & Final architecture

Index

1. Introduction

This section briefly describes the target system including a description of the technical terms to be used in the future.

1.1. System of interest

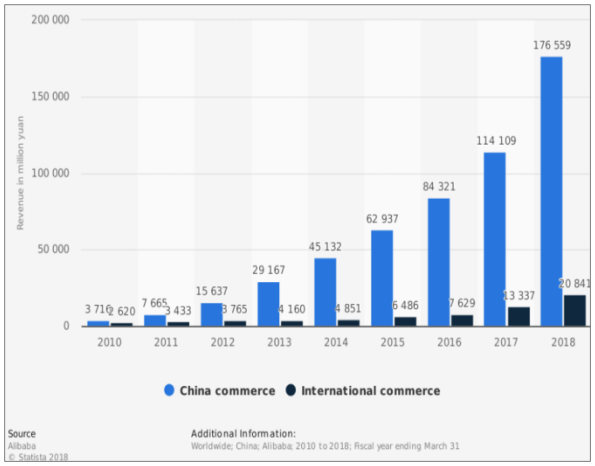


Figure 1. Annual e-commerce revenue of Alibaba from 2010 to 2018(in million yuan)

Due to the influence of the globalization era, there are many systems and platforms that can be purchased on the other side of the globe. Most of these systems are to purchase goods directly from foreign countries through Internet shopping malls. The representative system, AliExpress, is a B2C (Business to Consumer) site created by Alibaba Group, a Chinese Internet company in 2010. As of 2017, more than 100 million overseas customers are using it, and 20~30s are seen to use it a lot.

Figure 2. AliExpress Number of customers and distribution by age

The representative service of AliExpress is a live commerce system, which enables live video streaming to enable buyers to interact with and order from sellers in real time, and is a live broadcasting system for merchandise sales. The advantage of live streaming is that it is an introduction to live products, so users may have an immediacy to get the desired product information instantly. The buyer can solve the question of the product by asking the seller about the durability test and the wearing of the clothes. Even after the live streaming is finished, even if the buyer does not watch live while uploading the live video to the seller's store, he can check the product information while watching the uploaded video.

The seller of AliExpress can be a company or an individual, and the buyer can confirm the reliability of the seller with the seller rating.

2. System Architecture

This section describes architecture of entire system.

2.1. Architecture of AliExpress Live (Early version)

AliExpress Live has two types of server which are 'Web Server' and 'Streaming Server'. The 'Web Server' provides typical functions of online-shop. It manages information of products and payment transactions.

The 'Streaming Server' is only dedicated for video streams. It is physically separated from the Web Server because it needs to handle large sized multimedia. The data can be temporarily cached to proxy servers for providing better user experience.

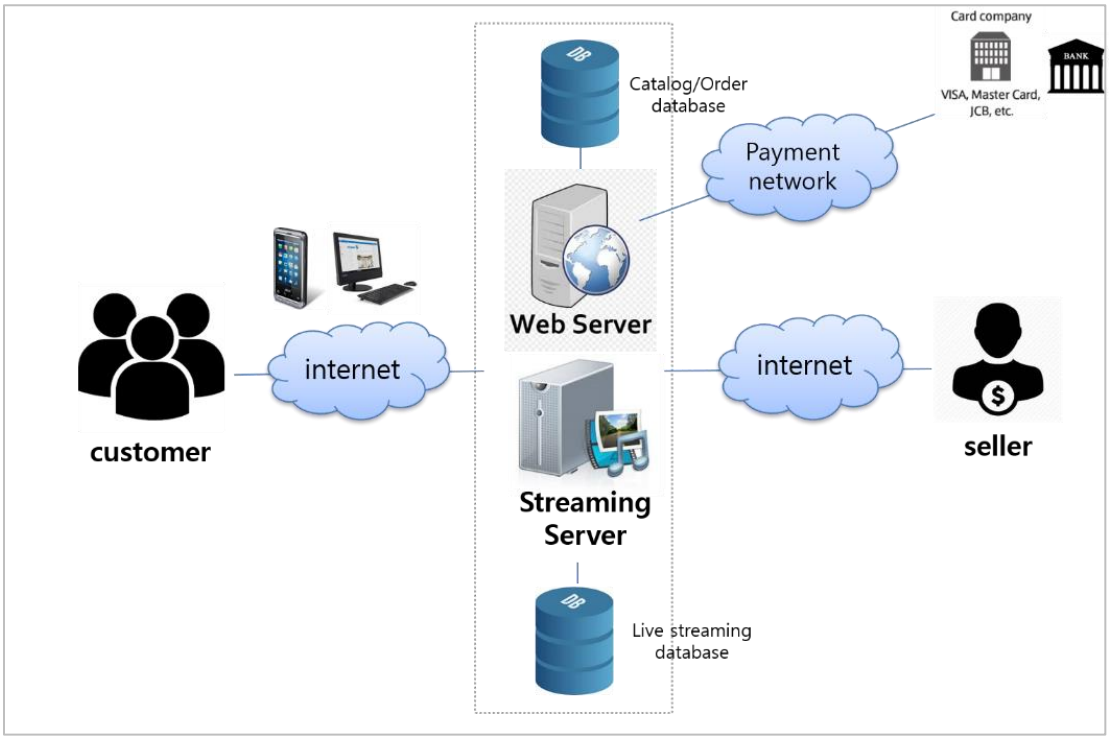


Figure 3. System Overview

3. Architectural drivers

3.1. Design Purpose

This is greenfield system in a relatively novel domain. The purpose is to produce a detailed design to support the construction of the system.

Business goal is to stimulates users' desire to purchase by exposing various products through media of live streaming. Customers who get a desire to purchase through video should be able to buy products easily. We should combine e-commerce system and multimedia system to satisfy business goal.

In detail, customers who use our system should be able to watch live video (or recorded one). When customers watch live video, they can interact with seller or other customers through live chat. Also, they can know the number of viewers who watch same live video. Customers who want to subscribe to a specific seller's broadcast can follow his/her channel. Customers can share video to others using variance social media and can give ‘like’ to video.

Seller who want to broadcast his/her live should be able to open his/her channel and start/stop live streaming. Seller also can interact with customers through live chat and know the number of viewers. Seller can upload his/her products and coupons to channel

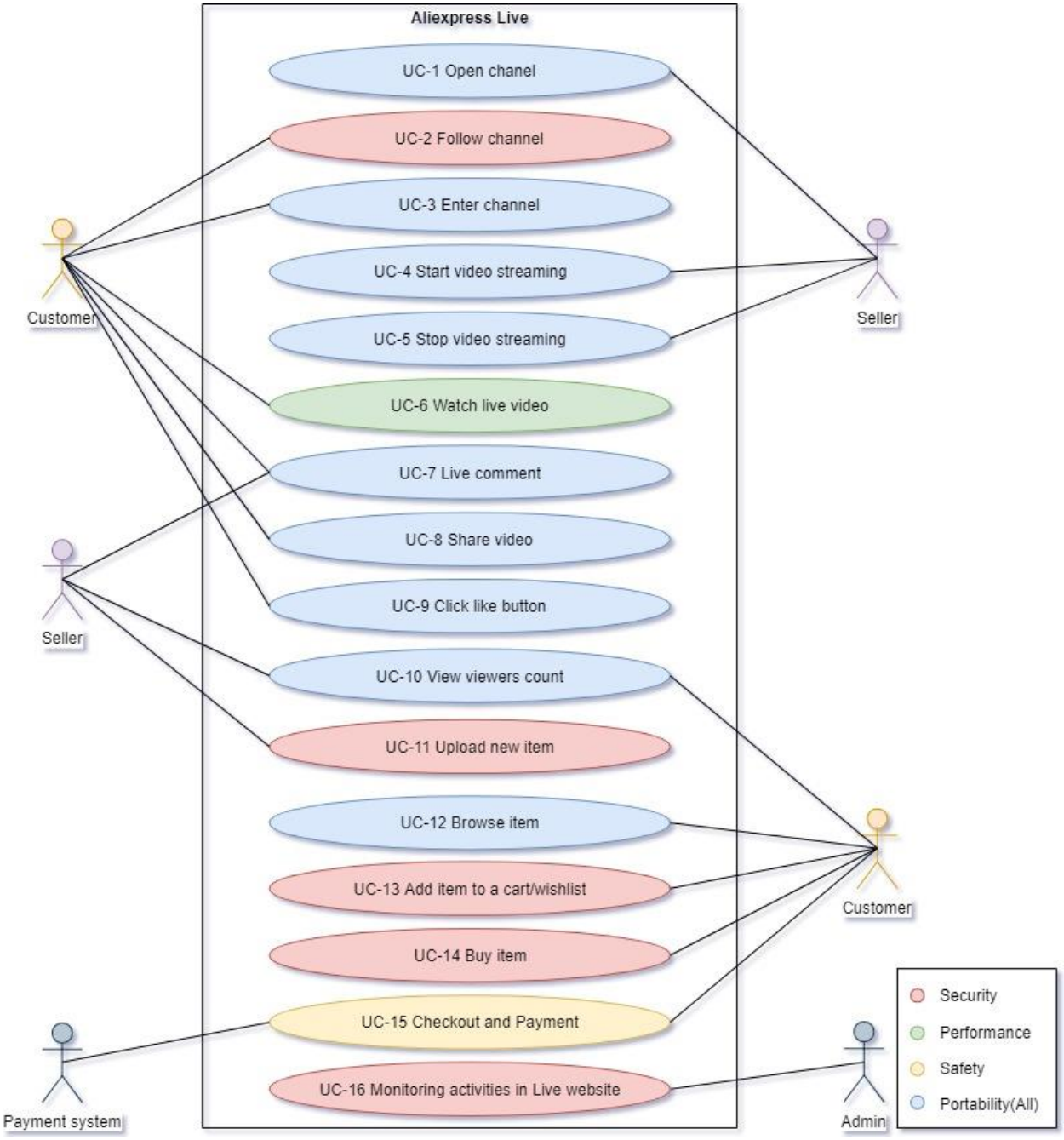
Customer can add products to wish list or shopping cart. Customer who want to buy products in the shopping cart (or directly buy) should be able to pay the price.

Our streaming system should provide enough quality and be capable to many customers. Our application system should protect information of user accounts. Payment system must safe from internal malfunction and external attacks.

To secure a large number of customers, customers should be able to access to our system using web(pc) and mobile. Also, our system should independent to operation system. But, for usability, our system should be able to run on popular version OS (e.g. Windows 7, 8, 10, Android Marshmallow, IOS10).

3.2. Use Case Model

This section contains the use case modeling for the AliExpress Live. Important use case scenarios based on the actors have been included here. This use case model depicts relationship between the actors -User, Seller, Admin and their role in the AliExpress Live site. Not only live streaming functionalities also order and payment functionalities are included in use case.



Use Case	Description
UC-1 Open Channel	The seller opens a channel regarding to process.
UC-2 Follow Channel	The customer follows a channel so the customer can be informed
UC-3 Enter Channel	The customer enters a channel
UC-4 Start Video Streaming	The seller starts live video broadcasting.
UC-5 Stop Video Streaming	The seller stops live video broadcasting.
UC-6 Watch Video	The customer watches live video.
UC-7 Leave Live Comments	The customer leaves a comment while watching live video. The comment can be seen by everyone who watches the live as well as the seller.
UC-8 Share Video	The customer can share the live video to other users. The video can be shared with URL, QR code and through SNS.
UC-9 Click Like Button	The customer clicks like button while watching video. 'Like' is displayed as an icon on live video screen.
UC-10 View Number of Viewers	The seller and the customer who watches live video can see how many users are watching the video.
UC-11 Upload New Item	The seller adds new item to his shop
UC-12 Browse Items	The customer browses product items in the channel. The customer gets detailed information with selecting specific item.
UC-13 Add Item to Cart/Wish-list	The customer adds an item to cart/wish-list.
UC-14 Buy Item Now	The customer buys items in the cart. The customer can view total amount of payment and choose how he/she will pay.
UC-15 Checkout and Payment	The payment system checks out payment with provided payment information.
UC-16 Monitoring activities in live site	Admin role has all access to monitor activities happens in live website.

Quality Attributes

ID	Quality Attribute	Scenario	Associated Use Case
QA-1	Security	A user of the system performs any action in the system such as updating personal information, banking information or any purchase activity should be visible only to them.	UC-2, UC-11, UC-13, UC-14, UC-16
QA-2	Performance	Streaming video should be good in quality (high bit rate) and buffering time should be less when large number of users watch the live at a same time (lag ratio should be low).	UC-6
QA-3	Safety	When a user makes a payment for their purchase, navigation between AliExpress and payment sites should be safe.	UC-15
QA-4	Portability	A user should be able to access AliExpress live from different devices like Mobile, Desktop and Tab.	UC-1 ~ UC-16
QA-5	Scalability	The system should be able to increase its scale with relatively small effort	UC-6

3.3. Architectural Concerns

ID	Concerns
CRN-1	Establishing an overall initial system architecture.
CRN-2	Leverage the expertise of the development team's knowledge of Java and Java-related technologies.
CRN-3	Allocate tasks based on the strengths of each member of the development team.
CRN-4	Introduction of new functionality must, as much as possible, avoid modifications to the existing core functionality.
CRN-5	Emphasize parallelism maximization in development and reduce dependencies between modules and teams.
CRN-6	Budget based on preliminary design and cost estimation should not exceed.
CRN-7	The system should be deployed with small effort.

3.4. Constraints

ID	Constraints
CON-1	The system must be accessed from a web browser and compatible to Internet Explorer, Chrome, Firefox and Safari.
CON-2	The system should be able to work on, IOS - It should support from IOS 5 to IOS 12 Android - It should support from Gingerbread version to Marshmallow Windows - It should support XP, Vista, windows 7 to 10.
CON-3	Payment history of recent 30 days must be stored in database.
CON-4	A majority of modules should be unit tested

4. Review Inputs

<table>
<tr><td>Quality attribute scenarios</td><td>The scenarios were described in 3.3. They have now been prioritized as discussed in SRS.
<table>
<tr><th>Scenario ID</th><th>Importance to the customer</th><th>Difficulty of implementation according to the architect</th></tr>
<tr><td>QA-1</td><td>Medium</td><td>Medium</td></tr>
<tr><td>QA-2</td><td>Medium</td><td>High</td></tr>
<tr><td>QA-3</td><td>High</td><td>Medium</td></tr>
<tr><td>QA-4</td><td>Low</td><td>Low</td></tr>
<tr><td>QA-5</td><td>High</td><td>Medium</td></tr>
</table>
We have relatively small amount of quality attributes here, so all of quality attribute scenarios are selected as architectural drivers.</td></tr>
<tr><td>Constraints</td><td>All of constraints discussed in 3.5 are selected as architectural drivers.</td></tr>
<tr><td>Architectural Concerns</td><td>All architectural concerns discussed in 3.4 are selected as architectural drivers.</td></tr>
</table>

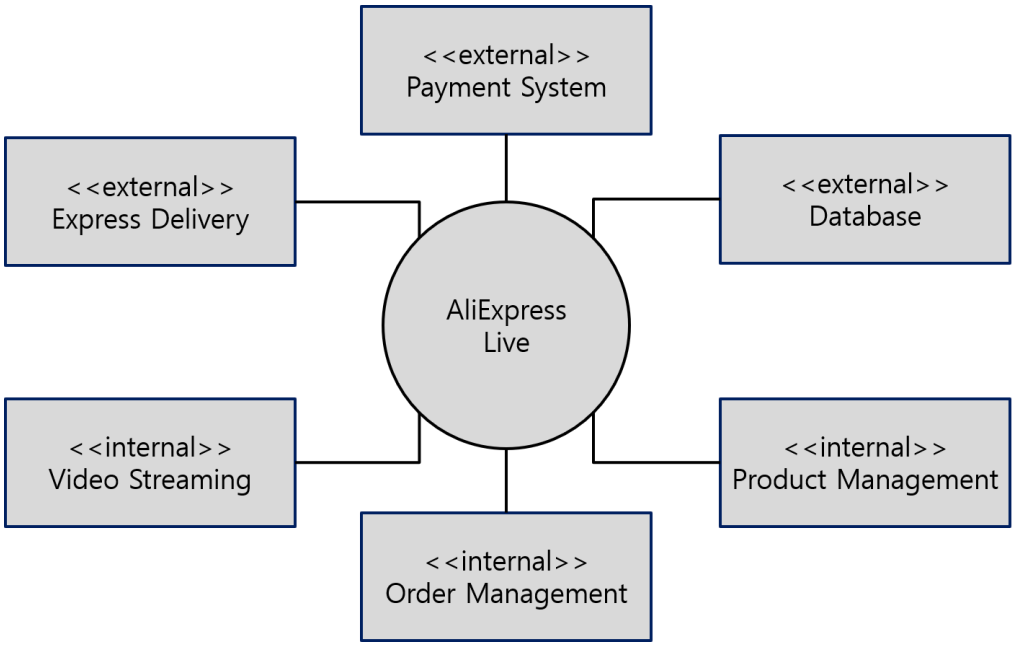
5. ADD Iteration 1 – Establishing overall system structure

5.1. Establish iteration goal by selecting drivers

Goal	Deriving overall structure of system with considering all of the drivers that may influence the general structure of the system.
Selected Drivers	QA-1: Security / QA-2: Performance / QA-3: Safety / QA-4: Portability CON-1: Web browser compatibility CON-2: OS compatibility CRN-1: Establishing an overall initial system architecture

5.2. Choose one or more elements of the system to refine

This is a greenfield system as live e-commerce system is well-known these days. So the items to be refined are all of internal systems.



5.3. Choose one or more design concepts design concepts that satisfy the selected drivers

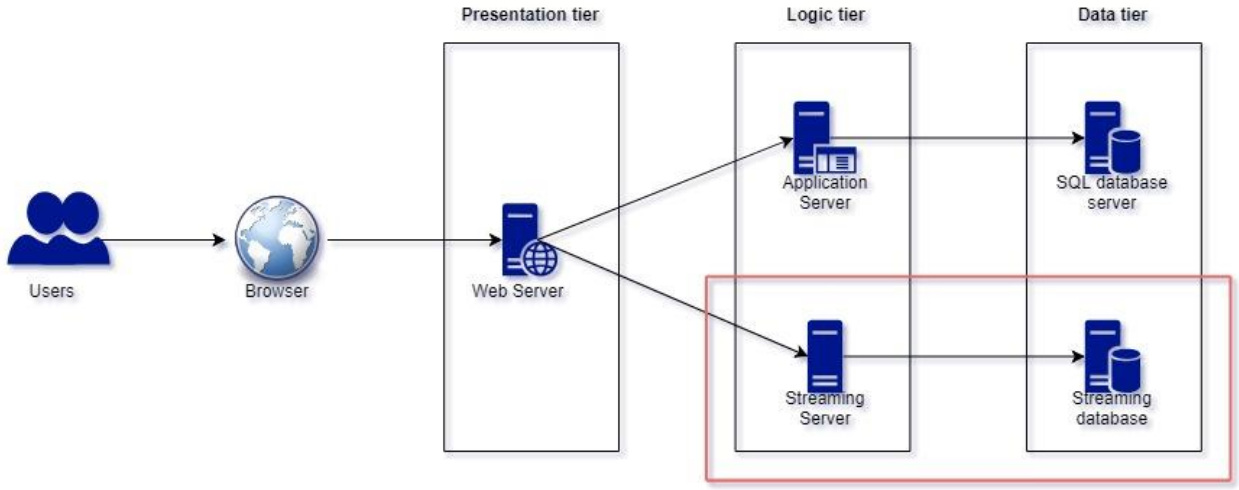
Design decisions and location	Rationale and assumptions
Using "N-tier architecture" as reference architecture.	"N-tier architecture" may include the web server that delivers web pages to the consumer's browser, application servers(UC-15), database servers(UC-11), and any other underlying servers or devices. We can also adopt new technologies like live video streaming service(UC-6) and add more components without having to rewrite the entire application or redesigning whole software, thus making it easier to scale or maintain(QA-5). Meanwhile, in terms of security, you can store sensitive or confidential information in the logic tier, keeping it away from the presentation tier, thus making it more secure. (QA-1)

5.4. Instantiate architectural elements, allocate responsibilities, and define interfaces

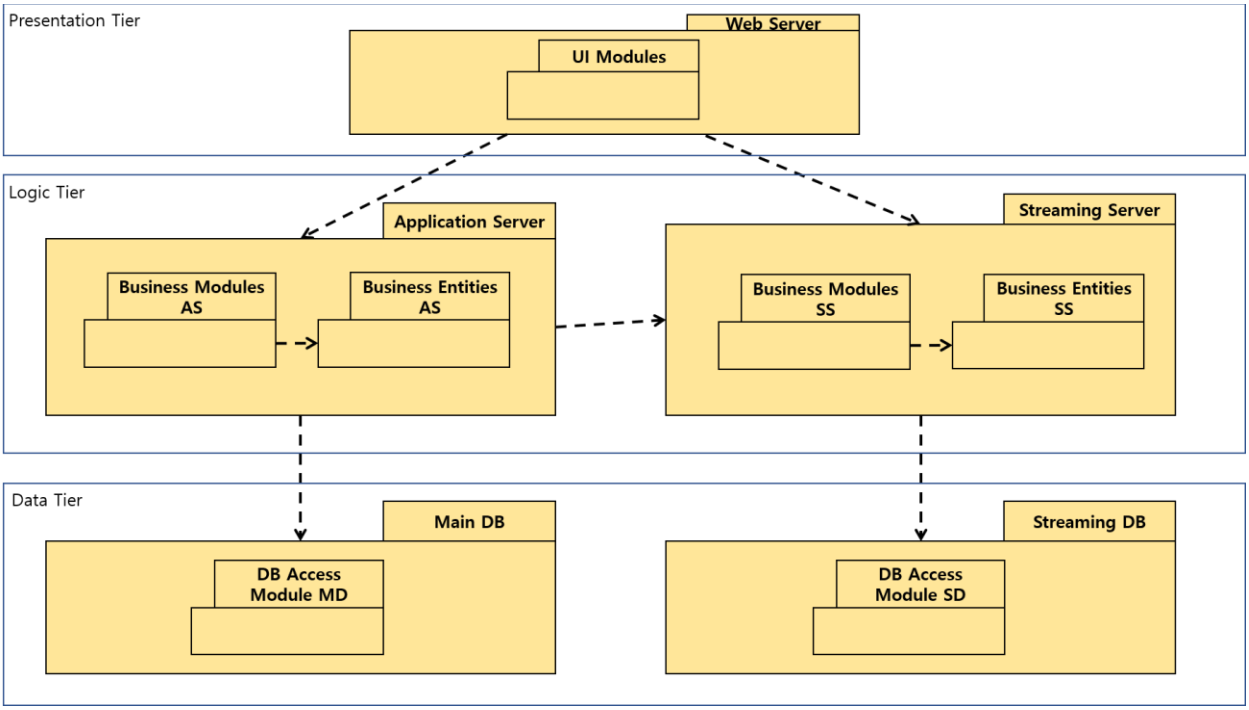
Design decisions and location	Rationale
Instantiate a presentation layer (web) , a logic layer (application), a data-storage layer.	We need typical 3 layers for e-commerce system. First, presentation layer (web server) are publicly accessible and are used to present information such as web pages, forms, advertisements, merchandise, and shopping cart contents to the consumer's web browser. Because web servers are publicly accessible, sensitive or confidential information—such as payment card data—must never be stored on web servers. Second, logic layer (application server) performs a variety of processing functions and should never be publicly accessible. In most cases, consumers do not interact directly with application servers, as the application servers receive requests from the web server, to process, format, and prepare data for storage or transmission. Application servers may also receive responses or retrieve content from database servers and subsequently pass the results back to the web server for presentation to the consumer. The data-storage tier includes database servers and any other system or media used to store data. Since databases may store sensitive information, including payment card data, database servers must never be publicly accessible. In a three-tier computing model, the database only accepts requests from and provides responses to properly formatted and authenticated requests, usually made by an application server.

Create new streaming server for live video streaming.	The system is a live e-commerce system, so the system should include streaming server for live video streaming service. (UC-6) Extend the module for streaming server and add new data storage to manage media data.(video…)

5.5. Sketch views and record design decisions



Element	Responsibility
Browser	It is responsible for the visual interface between the customer and the web server. (such as internet explorer, chrome, firefox…)
Presentation Tier	This tier contains modules that control user interaction and use case control flow.
Logic Tier	This tier contains modules that perform business logic operations that can be executed locally
Data Tier	This tier contains modules that are responsible for communication with the server.
Web Server	It is publicly accessible and are used to present information such as web pages, forms, advertisements, merchandise, and shopping cart contents to the consumer's web browser.
Application Server	It performs a variety of processing functions and should never be publicly accessible.
SQL database	It should store information about product items, payment and personal information.
Stream server	It performs live video streaming services.
Stream database	It should store media data(video) for video streaming service.



Element	Responsibility
UI Modules	These modules are for the user interface and receive user inputs.
Business Modules AS	These modules implement business operations. (User identity information, Payment and order management)
Business Entities AS	These entities make up the business model.(User account, Payment/Order information)
Business Modules SS	These modules implement streaming service. (Live video service, chatting service)
Business Entities SS	These entities implement streaming data.(Multimedia, interactive chat text)
DB Access Module MD	This module is responsible for people identity information, payment and order information. (UC-13,15)
DB Access Module SD	This module is responsible for storing video in the streaming service.

5.6. Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
		QA-5	Modules across the tiers and preliminary interfaces to support this use case have been identified.
		CRN-1	Overall initial architecture has been formed from the N-tier reference architecture.
	UC-6		Selected reference architecture establishes the modules that will support this functionality.
	UC-11		Selected reference architecture establishes the modules that will support this functionality.
	UC-15		Selected reference architecture establishes the modules that will support this functionality.
	QA-1		The elements that support the associated use case UC-11 have been identified.
QA-3			No relevant decision made, as it is necessary to identify the elements that participate in the use case that is associated with the scenario.

6. ADD Iteration 2 – Identifying structures to support primary functions

6.1. Establish iteration goal by selecting drivers

Goal	Identifying structures to support primary functionalities.
Selected Drivers	CRN-3: Allocate tasks based on the strengths of each member of the development team Channel related use-cases: UC-1 ~ UC-3 Video related use-cases: UC-4 ~ UC-6 Item management use-cases: UC-11, UC-12

6.2. Choose one or more elements of the system to refine

The elements that will be refined in this iteration are the modules located in the different tiers defined by the architecture derived from previous iteration.

6.3. Choose one or more design concepts design concepts that satisfy the selected drivers

Design decisions and location	Rationale and assumptions
Create a domain model for the application	Before starting a functional decomposition, it is necessary to create an initial domain model for the system, identifying the major entities in the domain, along with their relationships. There are no good alternatives. A domain model must eventually be created, or it will emerge in a sub-optimal fashion, leading to an adhoc architecture that is hard to understand and maintain.
Identify domain model that map to functional requirements	Each distinct functional element of the application needs to be encapsulated in a self-contained building block-a domain object. One possible alternative is to not consider domain objects and instead directly decompose sub-systems(servers) into modules, but this increases the risk of not considering a requirement.
Decompose domain objects into general and specialized components	Domain objects represent complete sets of functionalities, but this functionality is supported by finer-grained elements located within the sub-systems. The 'components' in the pattern are what we have referred to as modules. Specialization of modules is associated with the layers where they are located. There are no good alternatives to decomposing the layers into modules to support functionality.
Use Jetty framework[1] and Hibernate	Jetty framework is embedded web application server which enables a system to be stand-alone web server application without external Web Application Server(WAS). It supports

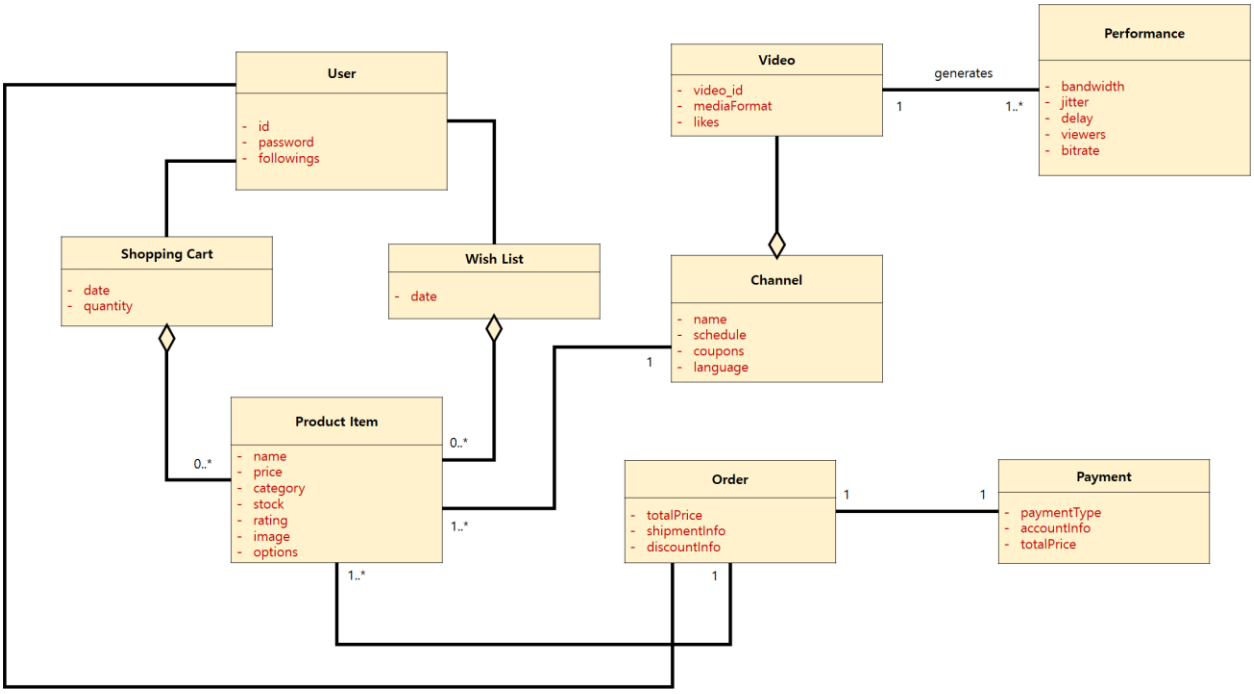
	servlet-based development which is friendly to development team. And it will reduce effort to deploy service to external WAS. Hibernate is a famous ORM(Object Relation Mapping) framework which enables us to map queried result from DB to Java entity. It will reduce effort exhaustive effort to generating entity classes.
Use PicoContainer[3] framework	PicoContainer is a framework for object injection technique. There are many other frameworks for this purpose, like Spring and Guice but PicoContainer is most light-weighted framework.
Use JSON RPC[4] to communicate among components	JSON RPC is a remote procedure call protocol encoded in JSON. It is very simple protocol, but the data can be formed in structured manner.

6.4. Instantiate architectural elements, allocate responsibilities, and define interfaces

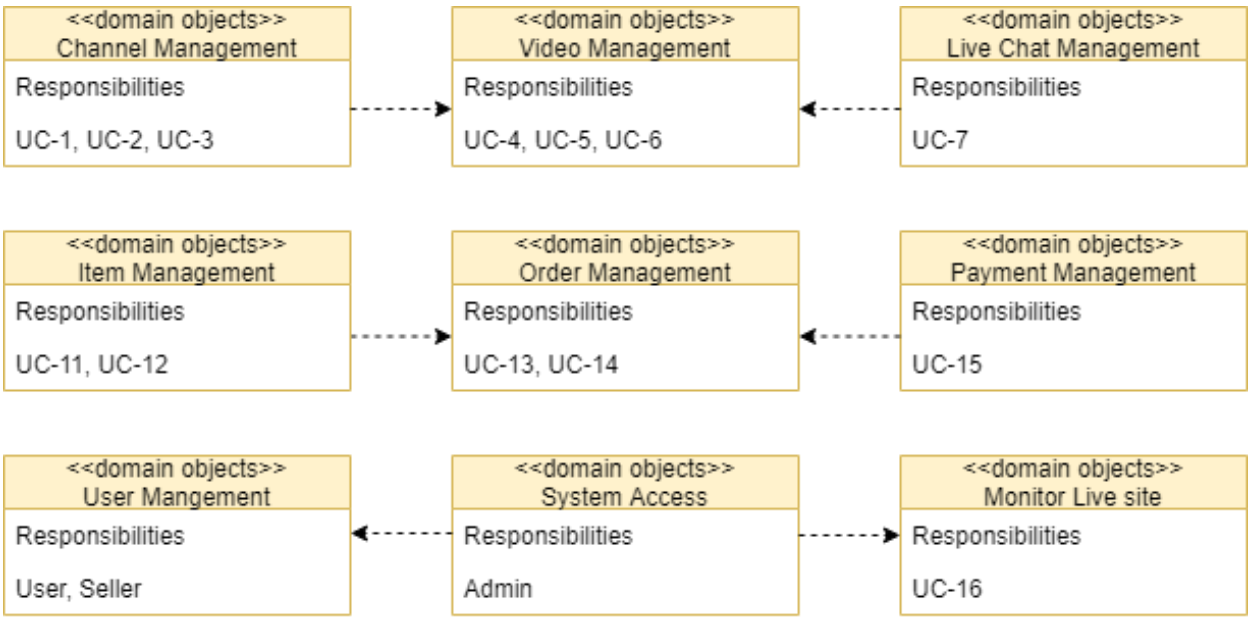
Design decisions and location	**Rationale**
Create only an initial domain model	The entities that participate in the primary use cases need to be identified and modeled but only an initial domain model is created, to accelerate this phase of design.
Map the system use cases to domain objects	An initial identification of domain objects can be made by analyzing the system's use cases. To address CRN-3, domain objects are identified for all the use cases in section 3.2.
Decompose the domain objects across the sub-systems to identify local modules with an explicit interface	This technique ensures that modules that support all the functionalities are identified. The architect will perform this task just for the primary use cases. This allows another team member to identify the rest of the modules, thereby allocating work among team members. Having established the set of modules, the architect realizes the need to test these modules, so a new architectural concern is identified here: CRN-4: A majority of modules should be unit tested. Only 'a majority of modules' are covered by this concern because the modules that implement user interface functionality are difficult to test independently.
Connect components associated with modules using PicoContainer	This framework uses an inversion of control approach that allows different aspects to be supported and the modules to be unit-tested(CRN-4)
Associate frameworks with a module in the data tier	ORM mapping is encapsulated in the modules that are contained in the data tier. The Hibernate framework[2] previously selected is associated with these modules.
Use JSON RPC for Interfaces for connecting between components	Interfaces the components across entire sub-systems uses are exported with JSON RPC

6.5. Sketch views and record design decisions

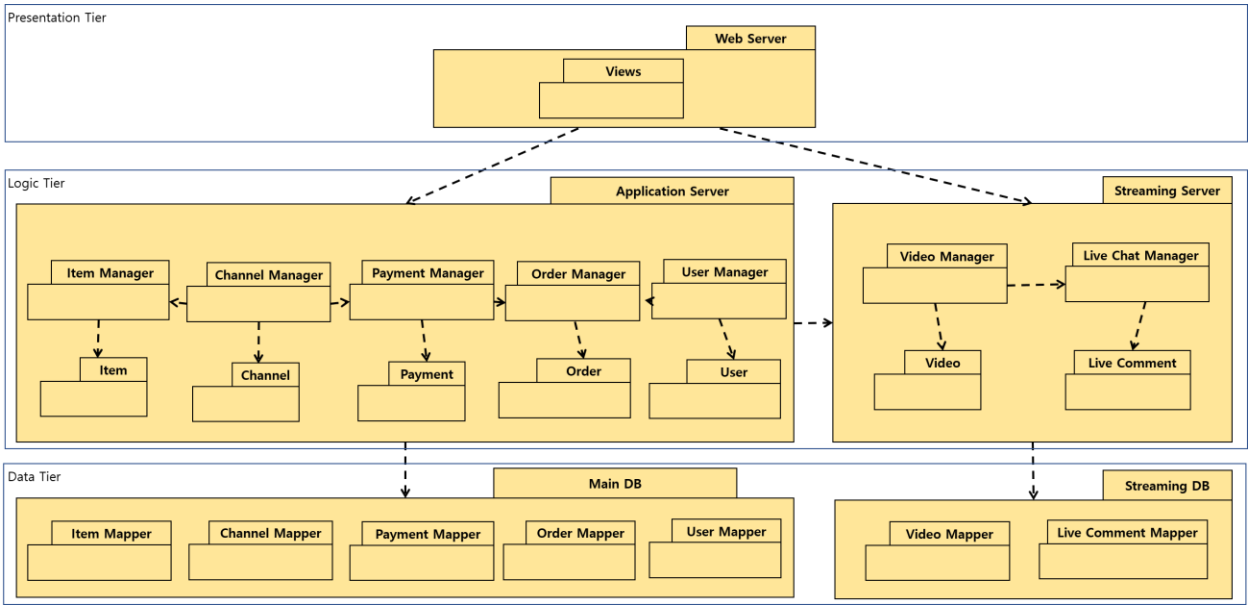
Initial domain model



Domain objects associated with the use case model



Modules that support the primary use cases



Element	Responsibility
Views	Constructs views which users will faces
Item Manager	Item Manager is used to perform activities such as Upload item and browse item.
Channel Manager	Channel Manager contains modules that controls activities such as open channel, follow channel and enter channel.
Payment Manager	Payment Manager where all the payment process will be handled.
Order Manager	Order Manager contains modules that activities related with user's order.
User Manager	User Manager contains modules that add/remove/modify user information.
Video Manager	Video Manager contains modules that performs Live streaming operation.
Live Chat Manager	Live Chat Manager is used to control chat operation during live stream.
Entities	Every entity like Item, Channel, etc. are representation of data in DB.
Object Mappers	Object Mappers like Item Mapper, Channel Mapper, etc. queries data from DB and maps it to class.

6.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
	UC-1 ~ UC-16		Modules across the sub-systems preliminary interfaces to support this use cases have been identified.
CRN-4			Modules need to be unit tested are not identified
	QA-1		The elements that support the associated use cases have been identified.
	QA-2		The elements that support the associated use cases have be en identified.
	QA-3		The elements that support the associated use cases have be en identified.
	QA-4		The elements that support the associated use cases have be en identified.

* Definitions

[1] JETTY FRAMEWORK

Eclipse Jetty is a Java HTTP server and Java Servlet container. While Web Servers are usually associated with serving documents to people, Jetty is now often used for machine to machine communications, usually within larger software frameworks. It uses less memory and is more lightweight thus offering speed and Scalability. Jetty can be embedded with ease in any Java web application, small devices line phones and setup boxes as well as serve as asynchronous server. It is open source with good community backing and support. Jetty has small footprint and can be quickly launched and restarted with ease. Widely used but still less known when compared to Tomcat. It is pluggable and extensive resulting in high degree of customizability. Built into several frameworks such as GWT, JRuby, Grails, Scala/Lift, etc. It's small and efficient with low maintenance and total cost of ownership

[2] HIBERNATE FRAMEWORK

Hibernate in short is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.

Hibernate is free software that is distributed under the GNU Lesser General Public License 2.1. Hibernate's primary feature is mapping from Java classes to database tables, and mapping from Java data types to SQL data types. Hibernate also provides data query and retrieval facilities. It generates SQL calls and relieves the developer from the manual handling and object conversion of the result set.

[3] PICOCONTAINER

PicoContainer is a mature dependency injector framework similar to spring-core or Guice. It presents the advantage to be quite small and yet very complete.

The advantages of dependency injection are numerous,

- Unloads the code from managing instance creation. So, no need to use the new keywords in the code, simply asking for the dependency will do.
- Minimize coupling between classes

PicoContainer is a small, simple container for arbitrary components. It is embeddable and extentensible. PicoContainer is designed for server and client side, for small and enterprise applications. There are already quite a few diverse containers that deliver the PicoContainer design. Components can be created for it without importing or extending any interfaces or classes outside of the java.* package. Components also have no meta-info specification. i.e. no XML. Most importantly of all, components for PicoContainer can be used without PicoContainer for good old-fashioned hard-coded deployments.

[4] JSON RPC

JSON-RPC is a remote procedure call protocol encoded in JSON. It is a very simple protocol (and very similar to XML-RPC), defining only a few data types and commands. JSON-RPC allows for notifications (data sent to the server that does not require a response) and for multiple calls to be sent to the server which may be answered out of order.

7. Iteration 3 – Addressing Quality Attribute Scenario Driver (QA-2)

7.1 Establish Iteration Goal by Selecting Drivers

Based on the fundamental structural decisions made in iterations 1 and 2, we can now start to reason about the fulfillment of some of the more important quality attributes

7.2 Choose One or More Elements of the System to Refine

For this iteration, the architect focuses on the QA-2 quality attribute scenario:

- Streaming video should be good in quality (high bit rate)

- Buffering time should be less when large number of users watch the live at a same time (lag ratio should be low).

7.3 Choose One or More Design Concepts that Satisfy the Selected Drivers

The elements that will be refined are the physical nodes that were identified during the first iteration:

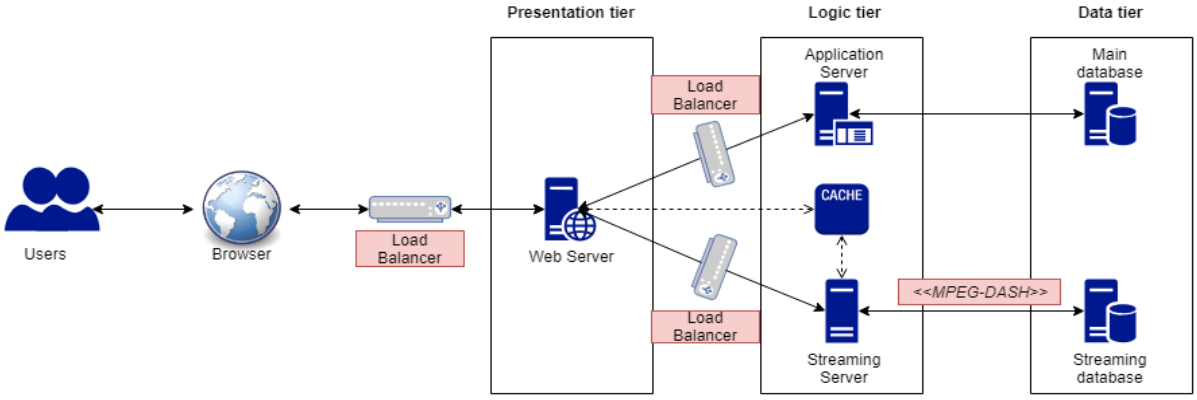
- Streaming Server / Application Server / Web Server
- Streaming database

Design decisions and location	**Rationale and assumptions**
Introducing **redundancy** tactic by replicating the streaming/application/web server and other critical components such as database	Using multiple components instead of a single component may increase reliability and availability through redundancy. By duplicating of critical components or functions of a system with the intention of increasing reliability of the system, usually in the form of a backup or fail-safe, or to improve actual system performance.
Introduce the **caching** tactic by deploying cache memory	By placing previously requested information in temporary storage, or cache, a cache server both speeds up access to data and reduces demand on an enterprise's bandwidth.
Introducing **adaptive bit-rate streaming** for streaming data	Adaptive bit-rate streaming is a technique used in streaming multimedia over computer networks. It works by detecting a user's bandwidth and CPU capacity in real time and adjusting the quality of the media stream dynamically.

7.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

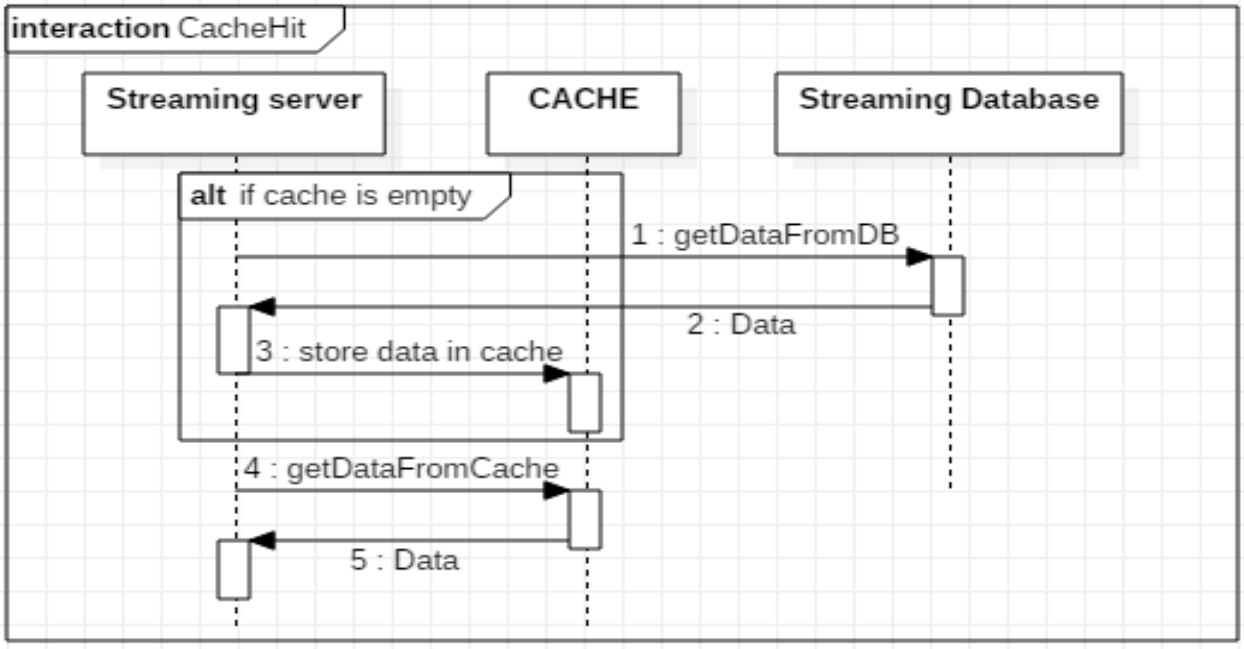
Design decisions and location	Rationale
Use **active redundancy** and **load balancing** in the streaming/ application /web server	Choose the **active redundancy** eliminates performance declines by monitoring the performance of individual devices, and this monitoring is used in voting logic. The voting logic is linked to switching that automatically re-configures the components. Because multiple servers are active in any time, it should distribute and balance the load between them.
Use Cache memory for streaming data	Choose **server side cache** to act of caching data on the server. It is common to cache commonly used data from the DB to prevent hitting the DB every time the data is required.
Implement **MPEG-DASH** protocol for streaming data	One of the newest protocols on the scene. While not widely used, this protocol has some big advantages. First, it supports adaptive bit-rate streaming. That means viewers will always be **delivered the best video quality that their current internet connection speed can support**. This can fluctuate second-to-second, and DASH can keep up.
Use **WOWZA** for streaming engine	The server is used for streaming of live and on-demand video, audio, and rich Internet applications over IP networks to desktop, laptop, and tablet computers, mobile devices, IPTV set-top boxes, internet-connected TV sets, game consoles, and other network-connected devices. The server is a Java application deployable on most operating systems.

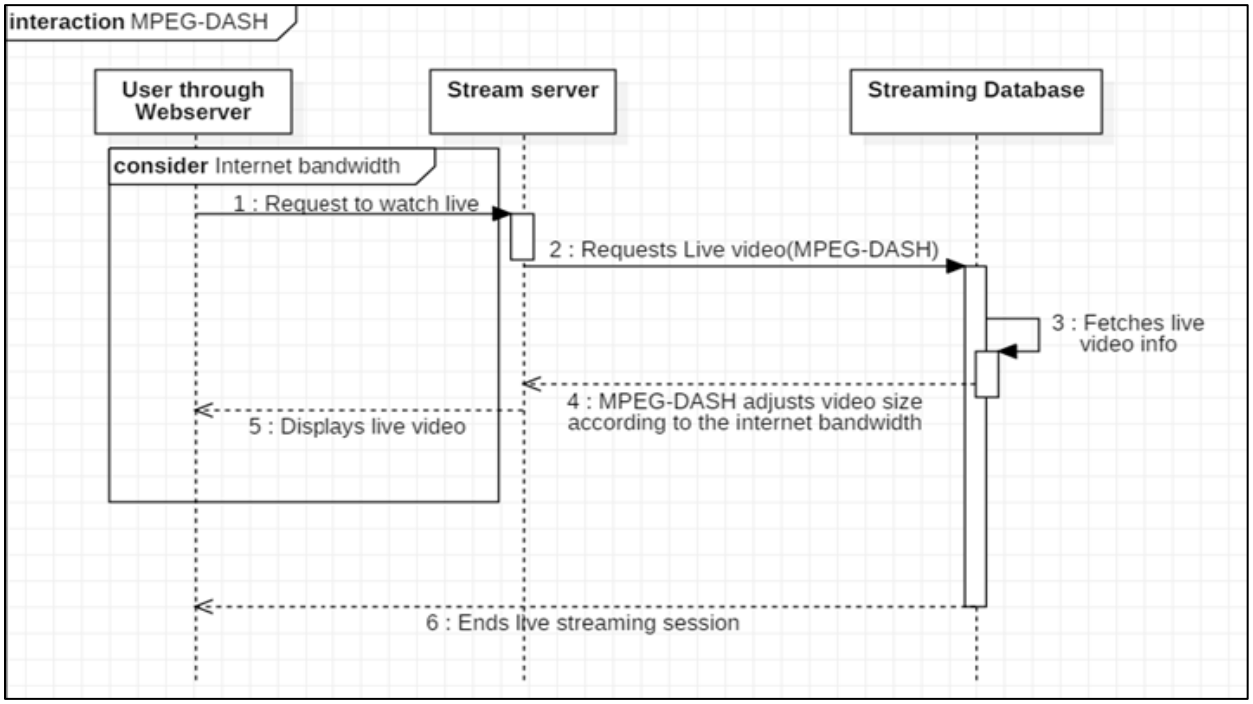
7.5 Sketch Views and Record Design Decisions



Element	Responsibility
Load Balancer	Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource.
CACHE	server side caching is the act of caching data on the server. Data can be cached anywhere and at any point on the server that makes sense. It is common to cache commonly used data from the DB to prevent hitting the DB every time the data is required.
MPEG-DASH	It supports adaptive bit-rate streaming. That means viewers will always be delivered the best video quality that their current internet connection speed can support. This can fluctuate second-to-second, and DASH can keep up.
WOWZA	The server is used for streaming of live and on-demand video, audio, and rich Internet applications over IP networks.

Sequential diagram





7.5 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
		UC-4 ~ 10	Modules across the sub-systems preliminary interfaces to support this use cases have been identified.
	CRN-2		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-3		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-4		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
		QA-2	The elements that support the associated use cases have be en identified.
QA-3			No relevant decisions have been made.

*Definitions

[1] MPEG-DASH

One of the newest protocols on the scene. While not widely used, this protocol has some big advantages. First, it supports adaptive bit-rate streaming. That means viewers will always be delivered the best video quality that their current internet connection speed can support. This can fluctuate second-to-second, and DASH can keep up.

MPEG-DASH fixes some long standing technical issues with delivery and compression. Another advantage is that MPEG-DASH is “codec agnostic”—it can be used with almost any encoding format. It also supports Encrypted Media Extensions (EME) and Media Source Extension (MSE) which are standards-based APIs for browser-based digital rights management (DRM). MPEG-DASH: one of the newest protocols on the scene. While not widely used, this protocol has some big advantages. First, it supports adaptive bit-rate streaming. That means viewers will always be delivered the best video quality that their current internet connection speed can support. This can fluctuate second-to-second, and DASH can keep up.

	HLS (HTTP Live Streaming)	**MPEG-DASH (Dynamic Adaptive Streaming over HTTP)**
Use	Video streaming	Video streaming
Segment	10sec	2-4sec
Video codec	Requires H264	Does not require any specific codec(H264,VP9,H265).
Manifest file (.mpd - media presentation description file)	This contains the meta data about the stream and has the extension .m3u8.	It has meta data also it has master manifest that links to ones encoded at different bit-rates. So that the quality of the stream can be adjusted during playback depending on the speed of the connection available.
Streaming type	Adaptive	Dynamic adaptive
Standard	No specific standard	International standard - first adaptive bit-rate HTTP format
Supports	Apple browser, Microsoft edge	Universal DRM solution supported by different browsers.
Encryption	HLS supports AES-128 encryption, along with Apple’s own DRM, Fairplay.	Content can be encrypted just once by using MPEG-CENC (MPEG Common Encryption) in conjunction with Encrypted Media Extensions (EME).

<< HLS vs. MPEG-DASH >>

[2] HTTPS

HTTPS appears in the URL when a website is secured by an SSL certificate. The details of the certificate, including the issuing authority and the corporate name of the website owner, can be viewed by clicking on the lock symbol on the browser bar.

	HyperText Transfer Protocol (HTTP)	HyperText Transfer Protocol Secure (HTTPS)
Port	Normal port no.80	Uses port 443 by default
Security	Vulnerable to attacks	More secured
Data	Does not scramble the data to be transmitted.	HTTPS scrambles the data before transmission.
Encryption	No Encryption	Uses data encryption
Domain Name Validation	HTTP website do not need SSL.	HTTPS requires SSL/TLS certificate.
Speed	Fast	Slower than HTTP

<< HTTP vs. HTTPS >>

[3] SSL

SSL stands for Secure Sockets Layer and, in short, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing hackers from reading and modifying any information transferred, including potential personal details. The two systems can be a server and a client (As our system, a shopping website and browser) or server to server.

[4] TLS

Transport Layer Security is just an updated, more secure, version of SSL. Think of an SSL/TLS certificate as a driver's license of sorts—it serves two functions. It grants permissions to use encrypted communication via Public Key Infrastructure, and also authenticates the identity of the certificate's holder.

8. Iteration 4 – Addressing Quality Attribute Scenario Driver (QA-1)

8.1 Establish Iteration Goal by Selecting Drivers

Goal of iteration 4 is satisfying QA-1. Our system has important database that contain critical information such as user account or payment information. We should protect these information from unauthorized external attackers. To protect main database from attackers, we have to modify our system architecture. In this case, we should concern about trade-off between performance and security. In other word, we should support sufficient security and performance at the same time

8.2 Choose One or More Elements of the System to Refine

In iteration 4, the first quality attribute scenario (QA-1: security) will be discussed.

- When users of the system perform any action in the system, such as personal information updating, private payment or purchase history, it should be visible only to the users and never be publicly accessible.
- Also, when the items of data that are important to users have been identified, it's important to ensure there is a standard representation format.

8.3 Choose One or More Design Concepts that Satisfy the Selected Drivers

For security, we select layer scheme to control flows of data. There are two flows; one is external flow between server and database, the other is internal flow in database.

Design decisions and location	Rationale and assumptions
Introducing the **layered architecture** to main database for security about critical data such as user account or payment information.	Our main database has various mappers (item, channel, payment, order, and user). If managers in application server communicate with data mappers directly, developer cannot control each flow of data between manager and mapper. This structure increases system complexity and vulnerability. To solve this problem, we choose **layered architecture** to main database. We divide main database to 2 layer; communication layer and data layer.
Introducing the **secure protocol** to encrypt communication channels.	We should encrypt communication between application server and main database. Because there are threaten about packet wiretap despite of we use layered database.

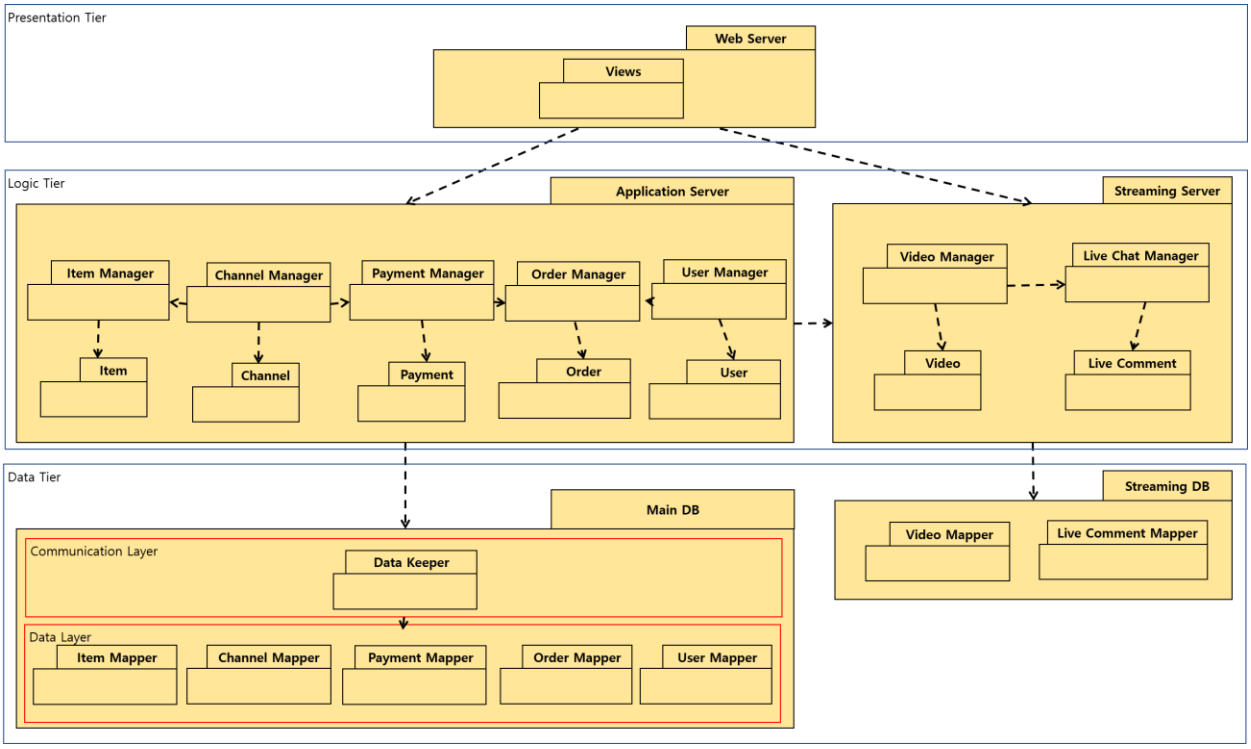
8.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design decisions and location	Rationale
Apply **DBMS** for creating and managing databases.	***Definition:*** A database management system (DBMS) is a system software for providing a central store of data that can be accessed by multiple users in a controlled manner. ***Usage:*** DBMS makes it possible for end users to create, read, update and delete data in a database. It serves as an interface between the database and end users, ensuring that data is consistently organized and remains easily accessible. ***Key points:*** The DBMS manages three important things: 1) the data; 2) the database engine that allows data to be accessed, locked and modified; 3) the database schema, which defines the database's logical structure. These three fundamental elements help provide concurrency, security, data integrity and uniform administration procedures.
The **Data keeper** is used for managing and protecting private database.	***Definition:*** The data keeper is a manager of the data in the database, rather than a user. ***Usage:*** Data keeper has responsibility for the development, implementation, operation, maintenance and security of the database and the applications that use it. ***Key points:*** 1) requires the development of the database structure and data dictionary (a catalogue of the data in the database); 2) the provision of security measures to permit authorized access and prevent unauthorized access to data; 3) and to guard against failures in hardware or software in order to offer reliability.
Firewalls used for Network security and **access control** within AliExpress Live.	***Definition:*** Firewall is network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. ***Usage:*** Built-in firewalls allow users to create private networks within AliExpress, and also control network access to user's instances and subnets by verifying the identity of users who wants to access data, resources, or applications. ***Key points:*** Identity and access management capabilities enable users to define individual users accounts with permissions across AliExpress resources. Here, identity and access management as well as the security aspects of amazon EC2, virtual private cloud (VPC), elastic load balancing (ELB), and cloudTrail are covered.
AliExpress provides efficient **Encryption** features.	***Definition:*** Encryption is the process of encoding a message or information in such a way that only authorized parties can access it. ***Usage:*** Flexible key management options allow users to choose whether to have AliExpress mange the encryption keys or to keep complete control over the keys by themselves. Here, it covers Key Management Service, S3 access controls, and database platform

	security features. ***Key points:*** There are two types for encryption in widespread use today: symmetric and asymmetric encryption. The names derive from whether or not the same key is used for encryption and decryption. • **Symmetric encryption:** In symmetric encryption the same key is used for encryption and decryption. It is therefore critical th at a secure method is considered to transfer the key between sender and recipient. • **Asymmetric encryption:** Asymmetric encryption uses the noti on of a key pair: a different key is used for the encryption and decryption process. One of the keys is typically known as the *private key* and the other is known as the *public key*. The priva te key is kept secret by the owner and the public key is either shared amongst authorized recipients or made available to th e public at large.
Database Backup is also important for enhancing the security of AliExpress.	Database backup is the process of backing up the operational state architecture and stored data of database software. It enables the creation of a duplicate instance or copy of a database in case the primary database crashes, is corrupted or is lost.

8.5 Sketch Views and Record Design Decisions

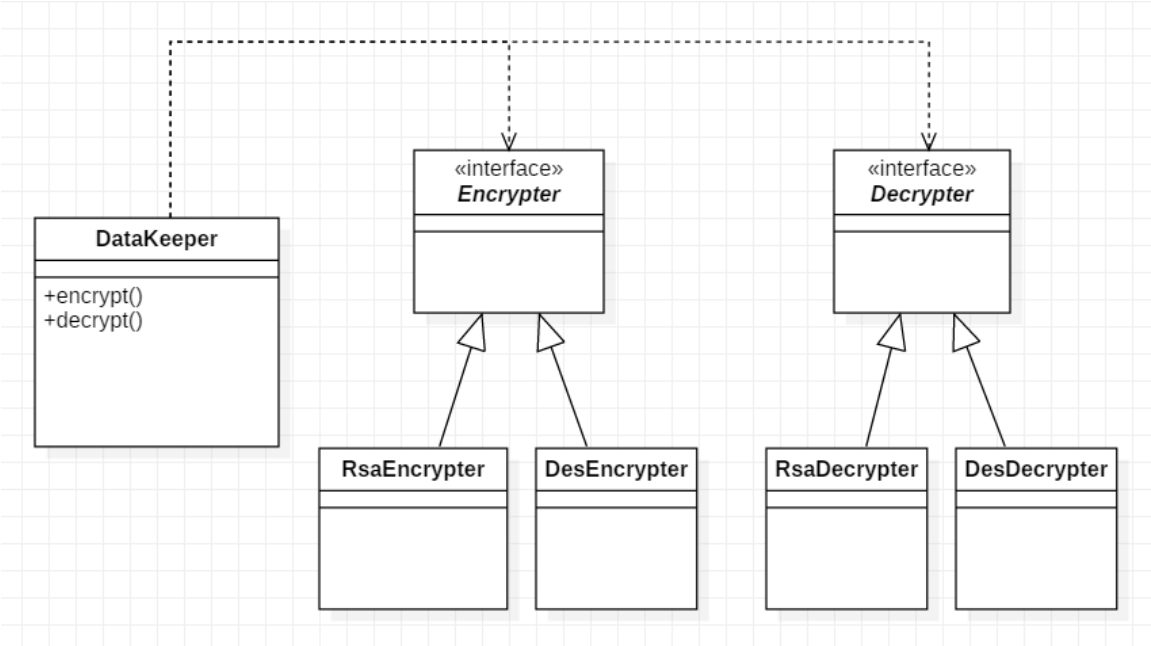
Package Diagram



Layered architecture is newly applied to Main DB sub-system. The data can be accessed only through 'Communication Layer' and 'Data Keeper' in this layer will encrypt data to store and decrypt data to deliver for trusted requests.

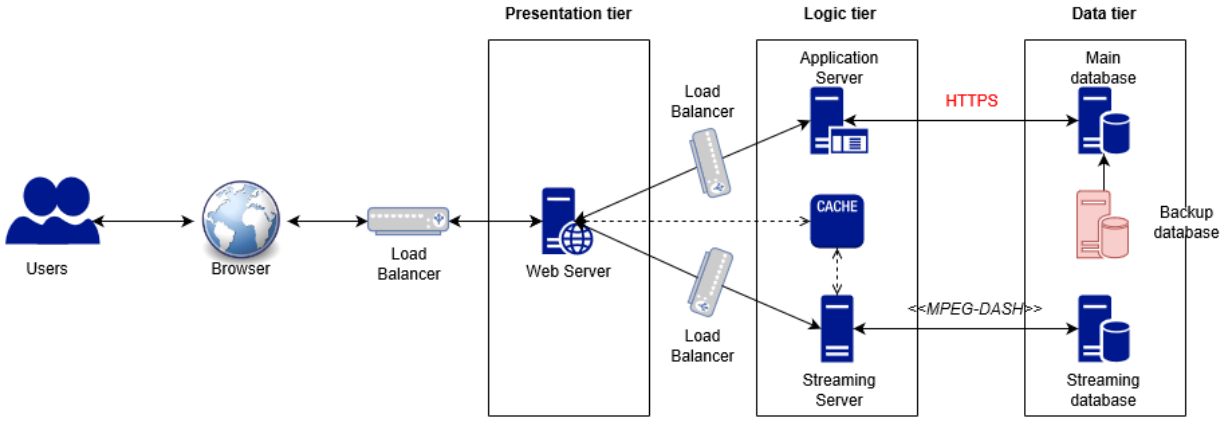
Data Keeper	A gate keeper of sensitive information. Every connection must ask to Data Keeper to access the data. Data Keeper encrypts data when it needs to be stored and decrypts data when it need to be delivered to outside.

Class Diagram for Data Keeper



Data Keeper delegates its encryption/decryption algorithm to dedicated classes. The concrete implementation can be replaced by adding new Encrypter/Decrypter classes.

Deployment Diagram



Communication between application server and main DB will be encrypted by HTTPS. As communication through network is vulnerable and it is possible to sniffed by hackers, we need to protect sensitive assets to be delivered safely. HTTPS is secure protocol so we can keep communication to be protected from outside.

Additionally a backup database is added to periodically replicate main database. It keeps data of recent 30 days so it can be restored when some error occurs on main database.

8.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
		UC-4 ~10	Modules across the sub-systems preliminary interfaces to support this use cases have been identified.
	CRN-2		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-3		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-4		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
		QA-2	The elements that support the associated use cases have be en identified.
QA-3			No relevant decisions have been made.
		QA-1	The elements that support the associated use cases have been identified.
		CON-3	The backup server is newly introduced to replicate main database periodically

9. Iteration 5 – Addressing Quality Attribute Scenario Driver (QA-3)

9.1 Establish Iteration Goal by Selecting Drivers

Our system is based on e-commerce system, so payment system for selling merchandise to customers is very important in our system. Therefore, safety transaction without failure during whole sequence is also important in payment process and architectural design decision is needed to satisfy it. In this iteration, we will discuss about payment safety attributes. (QA-3)

9.2 Choose One or More Elements of the System to Refine

In iteration 5, we will address quality attribute scenario 3 (QA-3: safety).

1. When a user makes a payment for their purchase, navigation between AliExpress and payment sites should be safe.

2. The whole process should be done in one transaction in case of an error situation in the purchase procedure.

9.3 Choose One or More Design Concepts that Satisfy the Selected Drivers

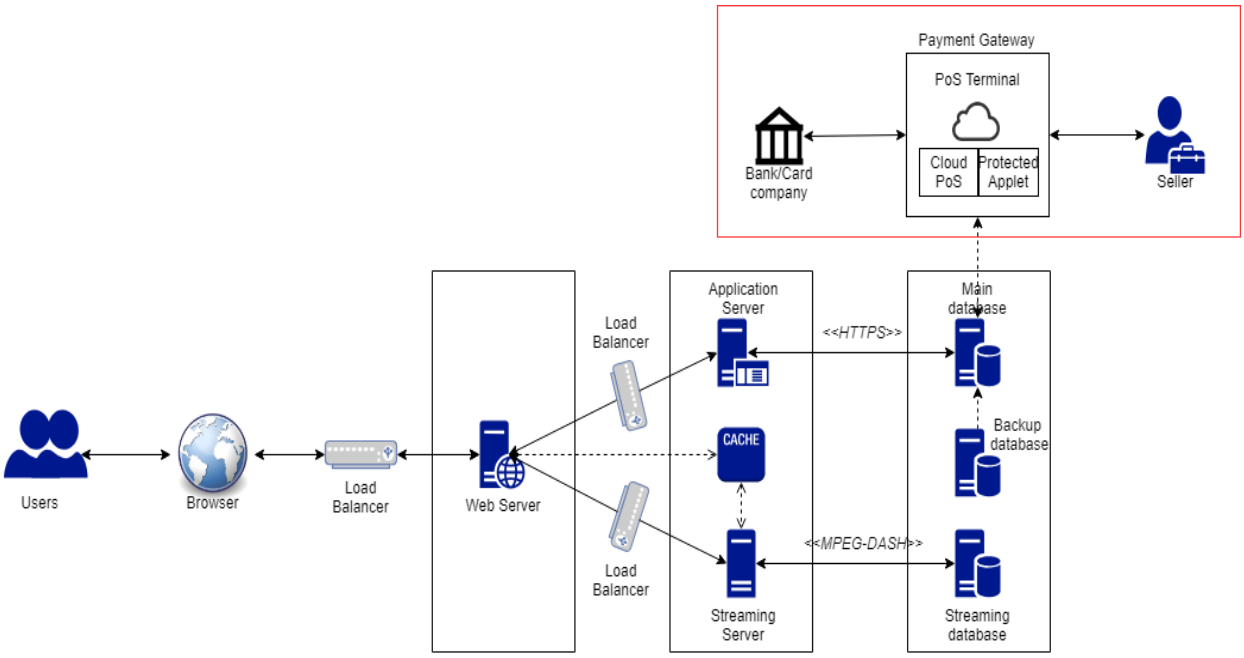
Design decisions and location	Rationale and assumptions
Introducing “Payment gateway”	A merchant service provided by an e-commerce application service provider that authorizes credit card or direct payments processing for e-businesses, online retailers, bricks and clicks, or traditional brick and mortar. The payment gateway may be provided by a bank to its customers, but can be provided by a specialised financial service provider as a separate service, such as a payment service provider. A payment gateway facilitates a payment transaction by the transfer of information between a payment portal (such as a website, mobile phone or interactive voice response service) and the front end processor or acquiring bank.

9.4 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

We select "Cloud base POS (Cloud POS)" technology to provide merchants with a low cost, feature rich, flexible, and secure payment transaction system. And we also select "Secure Elements" to ensure transactions during payment process.

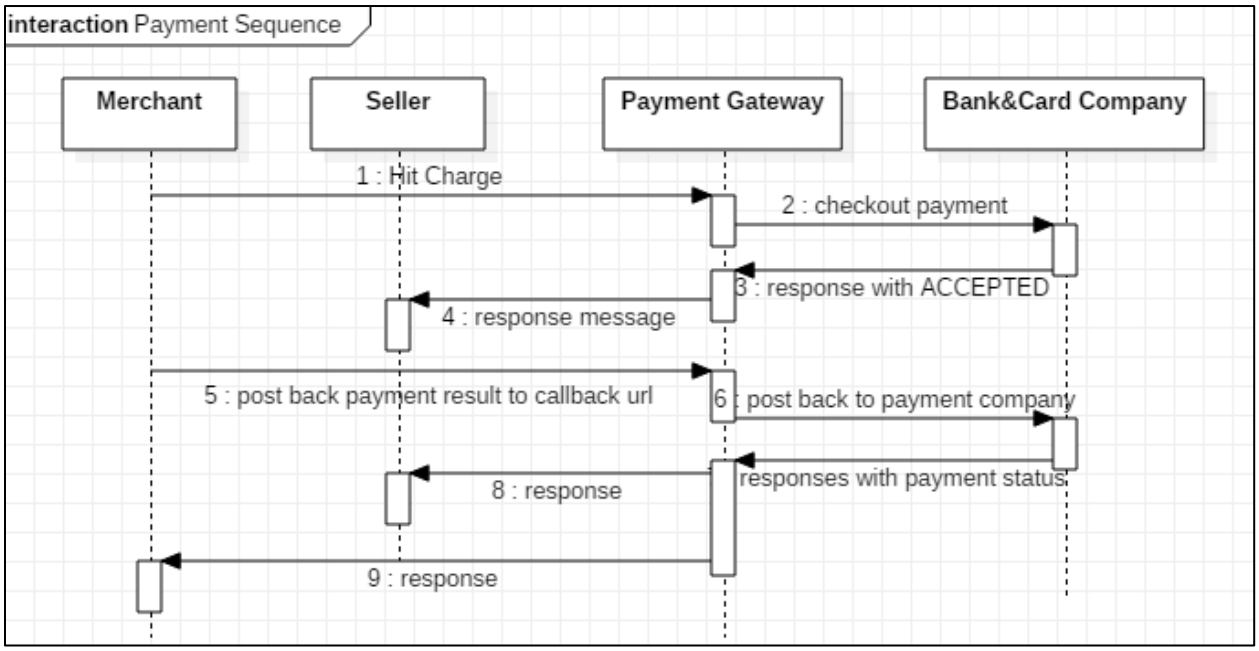
Design decisions and location	**Rationale**
Introducing "**Cloud base POS"** architecture for secure payment transaction system.	With Software-as-a-Service (SaaS) becoming popular, there is an increasing shift towards cloud based PoS systems. - Cloud PoS systems do **secure transmission** and storage of sensitive credit card data. - Data from multiple devices and locations are centrally stored and processed in the cloud. This simplifies data management and provides **secure access** to that data from any device, any place. - Cloud PoS systems have an offline mode in the event of a network failure. The payment application will **batch transactions** and send them once the network connection is re-established.
Apply "**Secure Elements"** to ensure transactions without failure.	**"**Cloud based POS" architectural solution uses a secure element to process credit card transactions**.** The credit card information is read by the PoS terminal and sent directly to a secure element, which Intel calls Protected Applet (PA), bypassing the PoS software. **The PA manages all transaction processing requests with the banks and can be configured to share certain data with the PoS software.** Moving all credit card processing actions to the secure element and completely bypassing the RAM ensures the sensitive data cannot be stolen by RAM Scraper malware. The secure element is designed to be tamper resistant and cannot be infected by malware.

9.5 Sketch Views and Record Design Decisions



Element	Responsibility
Salesforce Commerce Cloud (Cloud POS)	Salesforce Commerce Cloud provides a host of helpful POS capabilities such as digital commerce, mobile-first POS and store operations, predictive analytics, and order management. Online sellers can use this platform to rationalize their businesses for enhanced efficiencies. This POS platform delivers functionalities in intelligence, operations, and experience categories. Intelligence functionalities provide you with customer customization capabilities and valuable business insights. Operations tools enable you to enhance your key functions and integrate processes like back-office and order fulfillment. Experience features allow you to streamline pricing, promotions, products, and content to achieve higher customer engagement. As one, these different functionalities enhance your eCommerce business over diverse transaction channels like mobile, web, kiosk, store, and call center.
Protected Applet	The Protected Applet manages all transaction processing requests with the banks and can be configured to share certain data with the PoS software. Moving all credit card processing actions to the secure element and completely bypassing the RAM ensures the sensitive data cannot be stolen by RAM Scraper malware. The secure element is designed to be tamper resistant and cannot be infected by malware.

Below is payment sequence diagram. The payment gateway performs control and monitoring of the payment process as a whole and responds to the occurrence of an error situation during the transaction.



9.6 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
		UC-15	Modules across the sub-systems preliminary interfaces to support this use cases have been identified.
	CRN-2		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-3		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-4		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
		QA-3	The elements that support the associated use cases have been identified.
QA-4			No relevant decisions have been made.

10.Iteration 6 – Addressing Quality Attribute Scenario Driver (QA-4)

10.2 Establish Iteration Goal by Selecting Drivers

Business goal of our system is stimulate users' desire to purchase using live streaming. If user can access to our system using various devices (PC and mobile), we can get more chance to make profit. In other word, we should support PC website and mobile application (or website) both. At the same time, we should not change original architecture too much while add some modules that support portability. Therefore, our goal in iteration 6 is modifying the architecture to support portability (QA-4) but maintaining base structure of system (CRN-4).

10.3 Choose One or More Elements of the System to Refine

In iteration 6, quality attribute scenario 4 (QA-4: portability) will be discussed.

1. AliExpress Live should be produced for different computing platforms, such as Mobile APP, Mobile web browser and Desktop web browser.
2. Also, AliExpress Live should be able to move across different environment, not just across platforms. For example, modify software and make it adaptable to work on Linux, Windows, Mac OS, Unix, etc.

Hence, the system interfaces and application logic will be analyzed respectively.

10.4 Choose One or More Design Concepts that Satisfy the Selected Drivers

Design decisions and location	Rationale and assumptions
Using “Cross-platform” development tools	Our team do not afford time and member to development various platforms separately. Cross-platform development helps us to support various platform with one source. If our team members are good at a certain cross-platform development tool, we can use that tools.
Introducing “Factory patterns”	We choose factory pattern to avoid modifications to the existing core functionality. Factory pattern describe how to solve recurring design problems to design flexible and reusable object-oriented software, that is, objects that are easier to implement, change, test, and reuse.

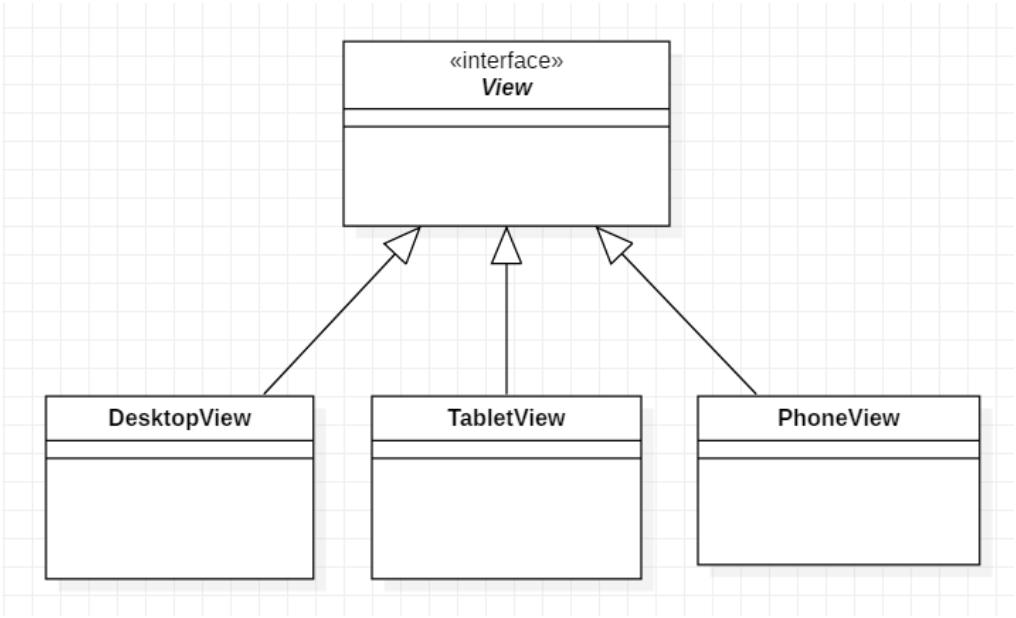
10.5 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design decisions and location	Rationale
'Three in One' - **Weex** (Vue Native) is tailored to AliExpress Live in a novel way.	***Definition:*** Weex is a framework developed by Alibaba industry, for building Mobile cross-platform UIs. *(Mobile APP, Desktop web browser)* ***Usage:*** Weex enables developers to use modern web development experience to build Android, iOS, and web applications with a single code-base. ***Advantages:*** Weex is different from web app, HTML5 app, or hybrid app and can be viewed as an improvement of React Native (Facebook). Also, Weex tries to keep up with modern development technologies and platform capabilities both for web and native.
In AliExpress Live's homepage, **Tangram** has been applied for dynamic layout.	***Definition:*** Tangram is a UI Framework for building a fast and dynamic Scroll View. *(Homepage of all applications)* ***Usage:*** Tangram provides a series of basic units, just like building blocks, you can build a page or adjust the structure of the page by quick assembly. ***Advantages:*** Compare to system standard controls (such as UI Collection View, Grid View), the advantages of Tangram are: (1) Easily control 'layout' selected for elements (cells); (2) provide default parser, quick parse JSON to View; (3) Provide several kinds of layout.
Progressive Web Application**(PWA**) combines the flexibility of the web app and naive app	***Definition:*** PWA is a web application loads like regular web pages or websites but can offer the user functionality. *(Mobile web browser)* ***Usage:*** AliExpress looked for a way to provide all of their web users with the benefits of their app, such as performance and the ability to work offline and re-engage users. They built a cross-browser PWA to combine the best of their app with the broad reach of the web. ***Advantages:*** (1) *Reliable:* load instantly and never show the downasaur, even in uncertain network conditions. (2) *Fast:* Respond quickly to user interactions with silky smooth animations and no janky scrolling. (3) *Engaging:* Fell like a natural app on the device, with an immersive user experience.
Design Java **factory patterns** for class-based programming.	***Definition:*** The factory patterns in Java uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created. * For easier understanding, we assume AliExpress Live contains

	multiple live streaming videos, and each video seller sells products, then the relationship between the three is abstract factory (AliExpress Live), simple factory (video seller), and product.

10.6 Sketch Views and Record Design Decisions

Structure of view classes

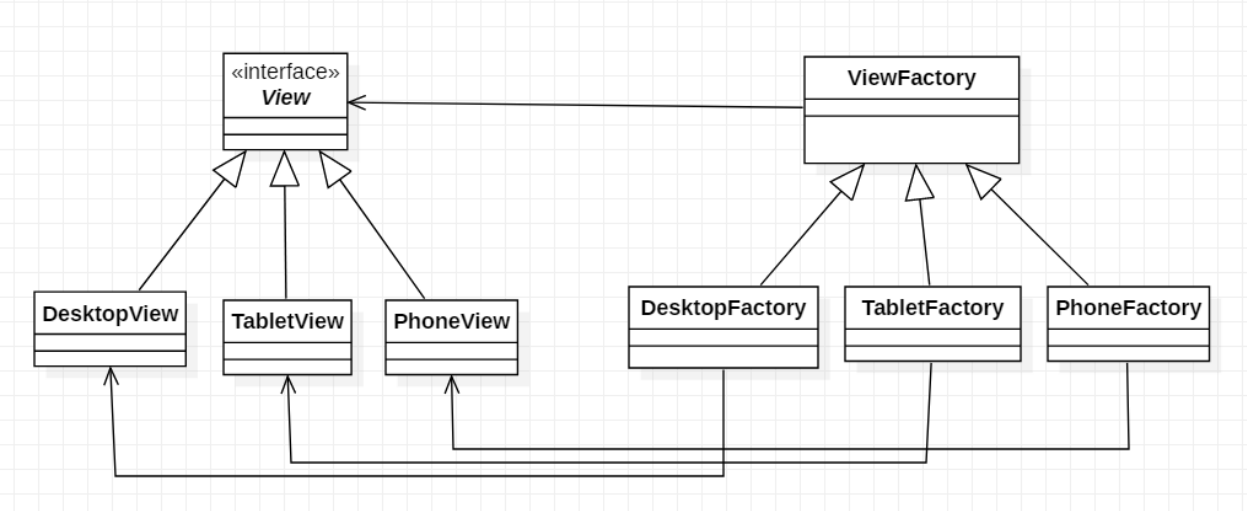


View classes composes the screen that users will face. As our system will be accessed through various types of devices such as desktop, tablet or phone, the view classes should provide different types of screen which is convenient to see with regard of device's screen size.

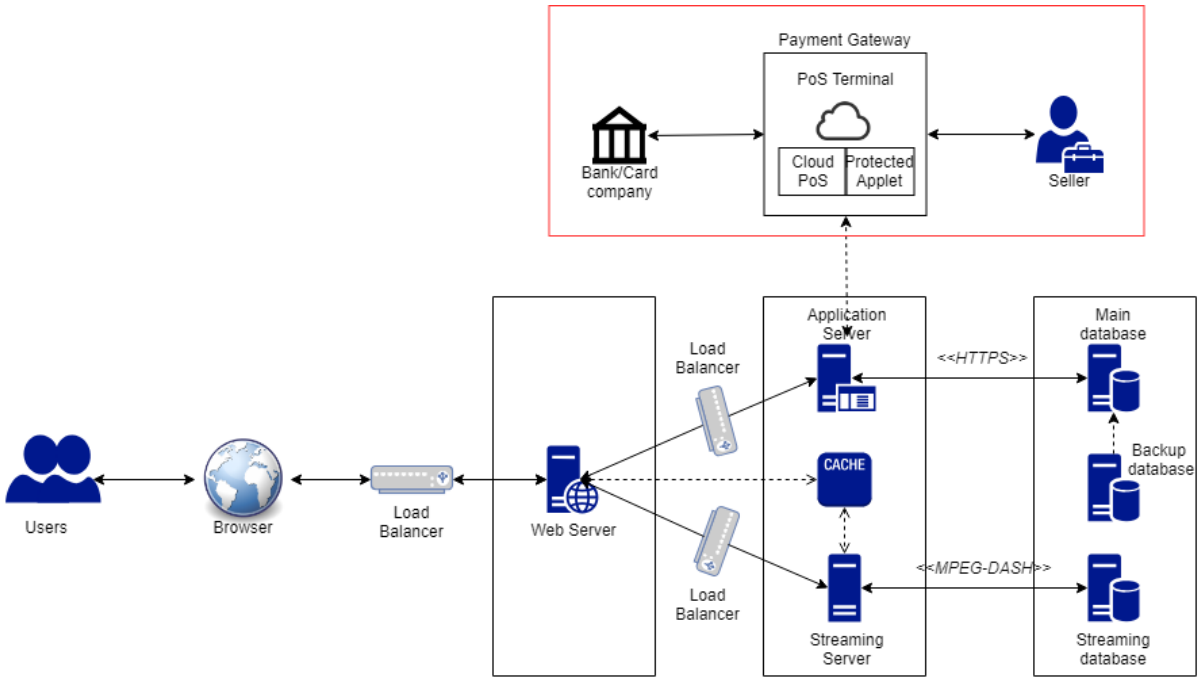
For this purpose, various types of views will be provided. And they will follow same interface named 'View' so we can add another view easily.

<<interface>> View	The interface of views
DesktopView, TabletView, PhoneView	Concrete views for various type of devices. Desktop has largest screen so the GUI components will be placed broadly and other mobile devices will place their GUI component densely.

Creation of Views



Factory pattern is applied to create views. We have several view factories which are dedicated for relative views. It will help creating views to be easier.



In this iteration we supply various types of view which are dedicated for specific device.

10.7 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
	UC-12		Modules across the sub-systems preliminary interfaces to support this use cases have been identified.
	CRN-2		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-3		This new architectural concern is introduced in this iteration: no relevant decisions have been made.
	CRN-4		This new architectural concern is introduced in this iteration: We choose “factory pattern” to avoid modifications to the existing core functionality.
		QA-4	The elements that support the associated use cases have been identified.

11.Iteration 7 – Addressing Extensibility and Testability

11.2 Establish Iteration Goal by Selecting Drivers

The goal of this iteration is twofold: the first goal is the Scalability of the system. When adding a new feature to the system, you must be able to add the existing feature without modification (except when you have to remove the existing feature). The second goal is to ensure reliability. To ensure the reliability of the functions before the system is deployed, the developer or the tester can measure the priority of the main functions and then generate the TCs according to the priority, and pass the certain TCs or higher to distribute the system.

11.3 Choose One or More Elements of the System to Refine

In iteration 7, we will address quality attribute scenario 5 (QA-5: Scalability)

1. Developer should be able to add new features to AliExpress Live system without modifications. In other word, the system must be scalable.
2. Also, developer should be able to test AliExpress Live system.

So, we will refine Application server side to add scalability and apply testing concept.

11.4 Choose One or More Design Concepts that Satisfy the Selected Drivers

Design decisions and location	Rationale and assumptions
Apply **'Plugin architecture'** for ext ending our application's function ality.	***Definition:*** A plugin architecture is an architecture that will call e xternal code at certain points without knowing all the details of th at code in advance. ***Advantages:*** (1) Extensibility: the application can be dynamically extended to include new features; (2) Parallel development: featu res can be developed in parallel by different teams; (3) Clear dev elopment direction: plugin framework provides a well-defined int erface and documentation for plugin writers, developers have a cl ear road-map for development; (4) Simplicity: a plugin typically h as one function, developers have only one single focus.
The usage of **'Unit testing'** reduc es the difficulties of discovering e rrors in our application.	***Definition:*** Unit testing is a software testing method by which indi vidual units of source codes are tested to determine whether the y are fit for use. ***Usage***: segregate each part of the program and test that the indiv idual parts are working correctly. ***Advantages:*** (1)Unit testing makes the coding process more agile; (2) It improves the quality of the code and find issues at an early stage; (3)It allows the programmer to re-factor code or upgrade s ystem libraries at a later date; (4) It also provides documentatio n, helps simplify the debugging process, so it can help reduce the

	cost of bug fixes.
'Mockup framework' is applied to isolate a portion of a system to test by imitating behavior of dependencies.	***Definition:*** Mock objects can be used to test a portion of the system, which can replace the dependencies. ***Usage:*** A mock object implements at least part of the same interfaces as the production object, but often in a simpler, faster, more predictable, or more controllable way. ***Advantages:*** (1) Mock behavior is closer to the test code because it is in the test code. This makes tests easier to understand without having to look at others; (2) We don't need to create other mock class because we only want slightly different behavior;

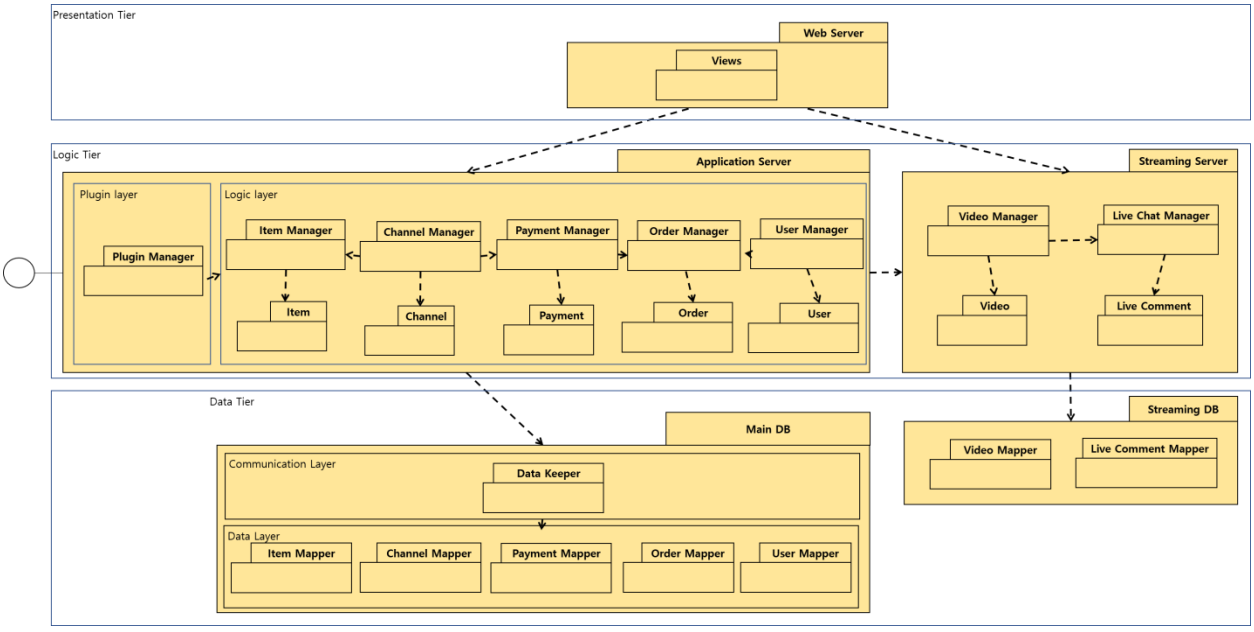
11.5 Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Design decisions and location	Rationale
Plugin Architecture **"Equinox"** has been used for extending our application's functionality without affecting the architecture.	***Definition:*** Equinox is an implementation of the OSGi core framework specification, a set of bundles that implement various optional OSGi services and other infrastructure for running OSGi-based systems. ***Usage:*** The Equinox OSGi core framework implementation is used as the reference implementation and as such it implements all the required features of the latest OSGi core framework specification. ***Advantages:*** The most powerful advantage would be that it could actually run code from the main plug-in in the actions, without reorganizing the code into separate plug-ins. Java classes required are instrumented on-the-fly, classes that are not loaded don't need to be processed.
For Unit testing, **"JUnit"** has been used.	***Definition:*** JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development ***Usage:*** JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. ***Advantages:*** (1) JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results. (2) JUnit tests can be organized into test suites containing test cases and even other test suites. (3) JUnit shows test progress in a bar that is green in the test is running smoothly, and it turns red when a test fails.

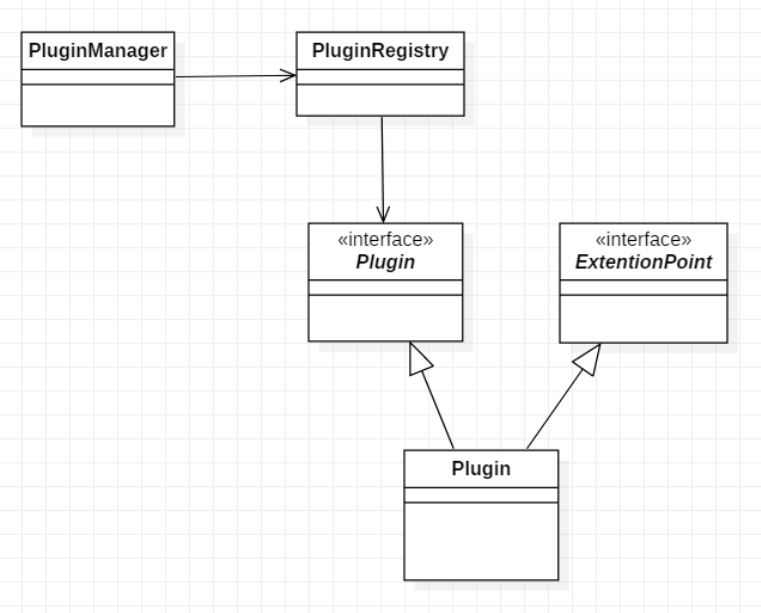
Mockup framework used in our application is **"mockito"**	***Definition:*** Mockito is an open source testing framework for Java released under the MIT License. The framework allows the creation of test double objects in automated unit tests for the purpose of test-driven development or behavior-driven development. ***Usage:*** Mockito is a popular mock framework which can be used in conjunction with JUnit. Mockito allows you to create and configure mock objects. Using Mockito simplifies the development of tests for classes with external dependencies significantly. ***Advantages:*** (1) Mock away external dependencies and insert the mocks into the code under test (2) Execute the code under test (3) Validate that the code executed correctly

11.6 Sketch Views and Record Design Decisions

Extending system with plugins



Application server got divided into two layers which are logic layer and plugin layer. Logic layer does core tasks for application server and plugin layer extends functionalities. Detailed structure of plugin layer is described below.



The plugin implements Plugin and ExtensionPoint interfaces. When a plugin is registered to PluginRegistry, PluginManager starts to track the plugin. The plugin which got started by PluginManager will add some functionalities to original service.

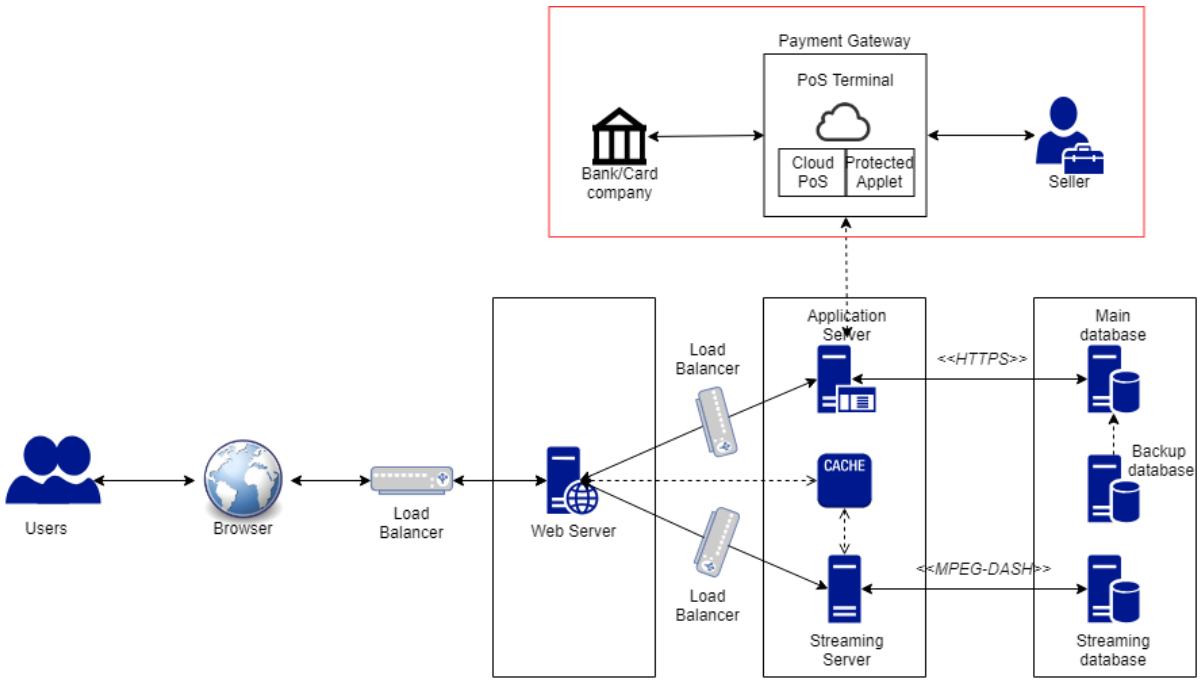
PluginManager	Manages cycle of plugins
PluginRegistry	Plugins get registered to PluginRegistry then the plugin can be tracked by PluginManager.
<<interface>>Plugin	Interface for plugins
ExtensionPoint	ExtensionPoint indicates what functionality has been extended from the plugin.

11.7 Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

Not addressed	Partially addressed	Completed addressed	Design decisions Made During the Iteration
		UC-1~16	Modules across the sub-systems preliminary interfaces to support this use cases have been identified. All use cases are functionally address in our design architecture. (such as "Manage channel, Watch video, Purchase products…")
		CRN-1~7	All architectural concern is addressed during iterations.
		CON-1~4	The constraints have been addressed.
		QA-1~5	The elements that support the associated use cases have been identified. (Security/Performance/Safety/Portability/Scalability)

12. Final Architecture

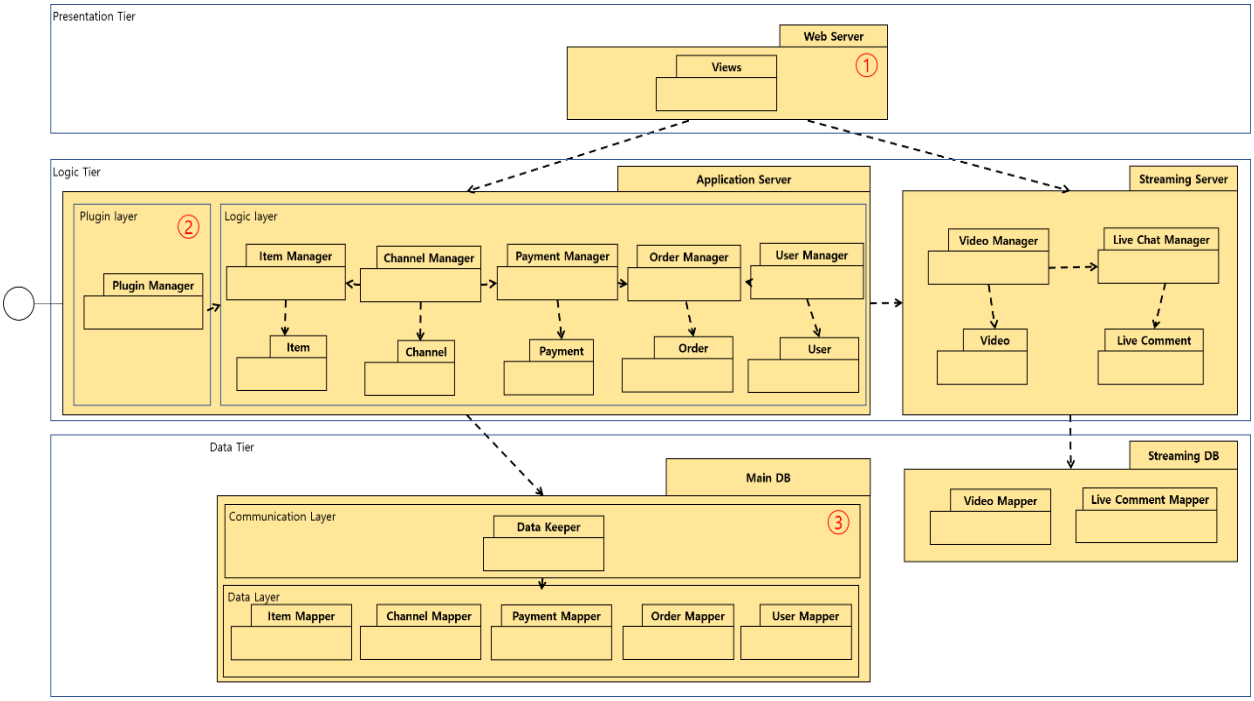
12.1. Physical View



Element	Responsibility
Browser	It is responsible for the visual interface between the customer and the web server. (such as internet explorer, chrome, firefox…)
Web Server	It is publicly accessible and are used to present information such as web pages, forms, advertisements, merchandise, and shopping cart contents to the consumer's web browser.
Application Server	It performs a variety of processing functions and should never be publicly accessible.
SQL database	It should store information about product items, payment and personal information.
Stream server	It performs live video streaming services.
Stream database	It should store media data(video) for video streaming service.
Load Balancer	Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource.
Cache	server side caching is the act of caching data on the server. Data can be cached anywhere and at any point on the server that makes sense. It is common to cache commonly used data from the DB to prevent hitting the DB every time the data is required.

12.2. Implementation View

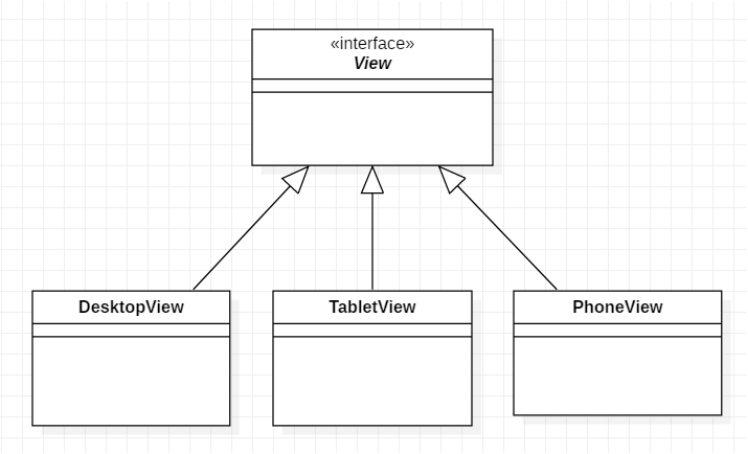
High level architecture



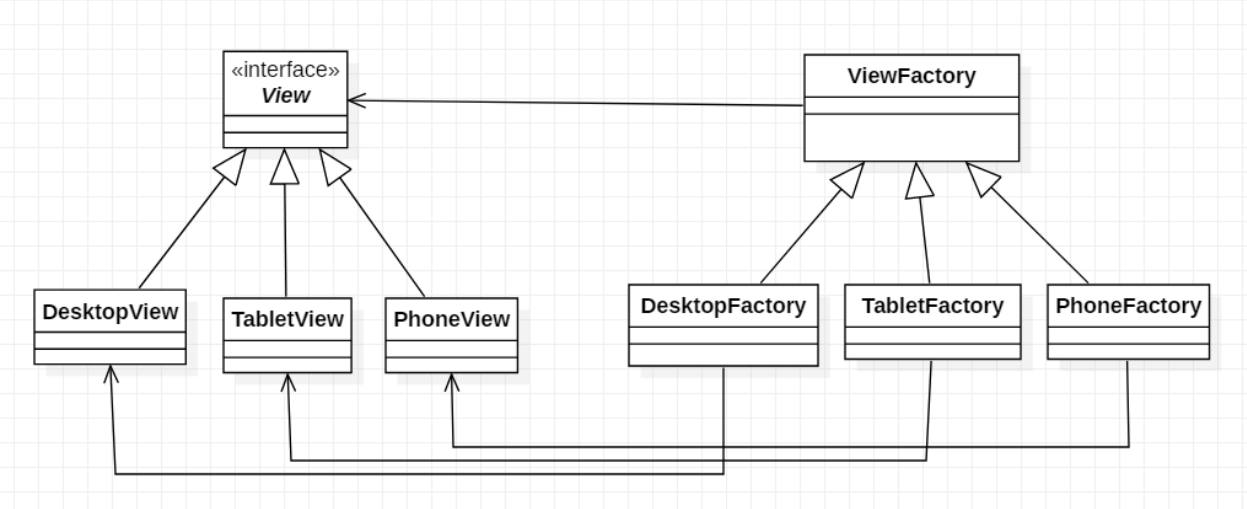
Presentation Tier	This tier contains modules that control user interaction and use case control flow.
Logic Tier	This tier contains modules that perform business logic operations that can be executed locally
Data Tier	This tier contains modules that are responsible for communication with the server.

12.3. Logical View

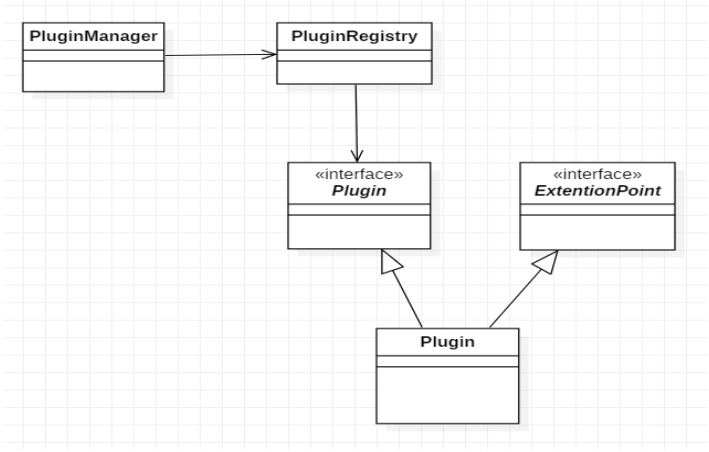
Break down of Views(1)



<<interface>> View	The interface of views
DesktopView, TabletView, PhoneView	Concrete views for various type of devices. Desktop has largest screen so the GUI components will be placed broadly and other mobile devices will place their GUI component densely.

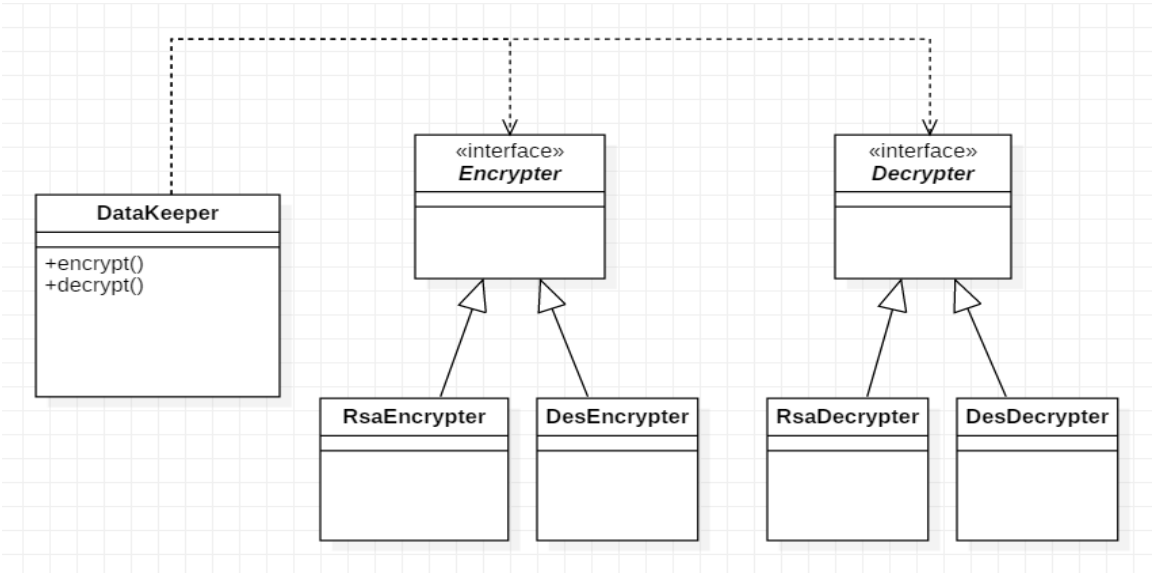


Break down of Plugin Manager(2)



PluginManager	Manages cycle of plugins
PluginRegistry	Plugins get registered to PluginRegistry then the plugin can be tracked by PluginManager.
<<interface>>Plugin	Interface for plugins
ExtensionPoint	ExtensionPoint indicates what functionality has been extended from the plugin.

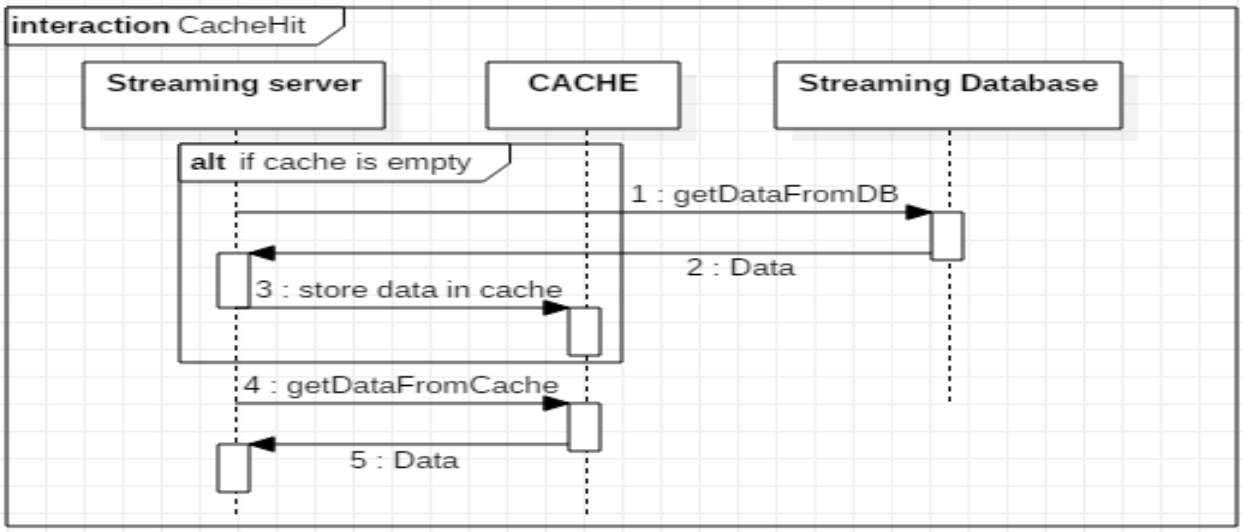
Break down of Data Keeper(3)



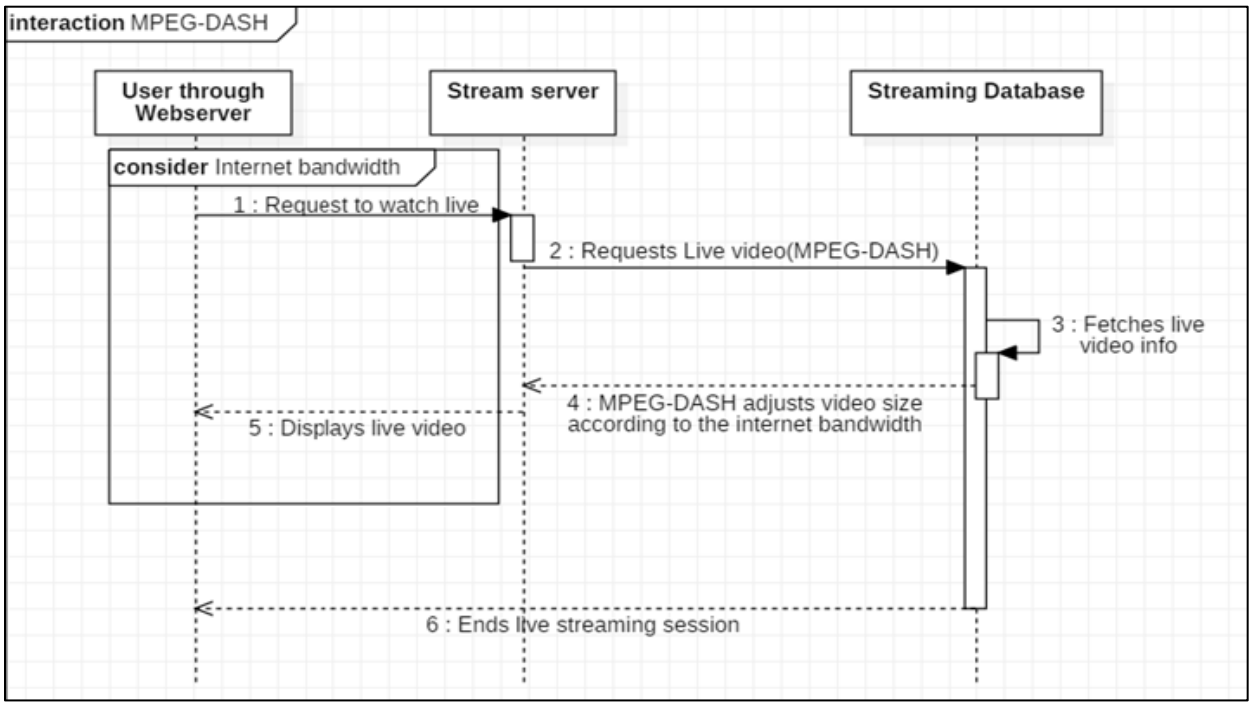
Data Keeper	A gate keeper of sensitive information. Every connection must ask to Data Keeper to access the data. Data Keeper encrypts data when it needs to be stored and decrypts data when it need to be delivered to outside.

12.4. Process View

Cache work flow



Video resolution adjustment with MPEG-DASH



13. Conclusion and Future work

The results presented in this report highlights important architectural design considerations based on ADD method for AliLive. First, with the help of reverse engineering we have examined AliLive's requirements, design, architecture and work flow. Second, from our understanding, we have came up with the requirements document with important use cases, quality attributes, constraints and architectural concerns. Finally, ADD architecture has been designed incrementally through iterations. After comparing the well-known approaches, the best and suitable architecture drivers has been chosen and applied to our system. Also, discussion on drivers has been provided.

Many different adaptations, tests, and experiments have been left for the future due to lack of time(i.e. the experiments with real data are usually very time consuming, requiring even days to finish a single run). Our future work concerns deeper analysis of the architecture and build an application based on our analysis.

14. References

[1] Zheng Qin, Han Yi, Li Shundong, Dong Jinchun, Yan Lixiang, Qin Jun **"E-commerce Architecture and System Design"** Introduction to E-commerce pp 271-303

[2] N-tier architecture - https://stackify.com/n-tier-architecture/

[3] Video Streaming - https://www.dacast.com/blog/video-streaming-protocol/

[4] Payment Architecture - https://www.trendmicro.com/vinfo/us/security/news/security-technology/next-gen-payment-processing-architectures

[5] Ian Sommerville, **"software engineering"** - 10th edition

[6] Humberto Cervantes, Rick Kazman, **"Designing Software Architectures - A Practical Approach"**