Youtube System Requirements.

In brief (will be flushed out below) There are 3 types of actors.

1. Standard user - search, view, and upload videos. Can provide feedback on videos in the forms of likes/comments.
2. Advertisers - bid to display an advertisement on a video.
3. Moderators - Provide restrictions on videos, (ex they may prevent a video from being viewed in a certain region), can remove videos/comments.

The system must at a minimum…

1. Provide an interface for standard users to search, view, upload, and comment on videos.
    a. System must allow users to search a video by name and filter result based on videos statistics(see 2)
    b. System must allow users to categorize the videos they upload into categories (see 3)
    c. System must allow users to see who uploaded the video and how popular the user who uploaded the video is.
    d. System must track what type of videos the user enjoys watching
    e. System must provide recommendations for videos based on 1d.
    f. System must provide a way for users to flag inappropriate comments/videos to grab the attention of moderators.
    g. System must provide a home screen for users to view their recommended videos/provide a search screen.
2. Record statistics for each video including at the minimum number of unique viewers, time watched, and most post popular region. Track viewership based on time.
3. Provide a mechanism for categorizing videos based on whether the video is related to music, news, sports.
4. Provide weighted statistics on the videos as 1. a function of (time, viewership) - how the popularity of the video increases/decreases as time moves on 2. a function of (region, viewership) - in which regions are a video more/less popular, as a function of (region, viewership, category) - which region is more particularly interested in a certain category.
5. Provide an interface for advertisers to view the statistics mentioned in step 4 (may be of interest to them).
6. Allow advertisers to bid on videos
7. Allow advertisers ,to see some statistics representing how many users “interact” with the advertisement(ie - click on it.)
8. Allow moderators to search and view videos like any user with special privileges including
    a. Ability to remove a video from a certain region or permanently delete it.
    b. Suspend a user.
    c. Remove comments.

See below for the actual diagrams.

The Class Diagram shows the classes and the relationship between each of the classes. A square is a class. An arrow indicates that one class interacts with another class in some way. In particular, each arrow is really bidirectional, I just couldn't find a double arrow on the tool I was using.A double line indicates a " has a" relationship and a dashed line represents a "is a" relationship. The class more centered is the one that "has a" and the one is more centered is the object that many things are (couldn't find arrows for these). In terms of the content of the diagram, there is a user class which only needs to keep track of its cookie. It interacts with server class. The server class doesn't maintain any state since that wouldn't scale if there are too many requests for it to serve requests frequently. Instead, it communicates with a/many database class. The database class has a userinfo class which acts as map for a userInfo class which stores the stats of the actual user. It also has a video Filterer class which provides a way for the database to search through videos very quickly, as a map and by keywords. The video filter acts as a controller to the actual videoInfo section. This allows the controller to update the state of the Video Info whenever a video is searched for. For the other actors, a moderator functions as a special type of user (is a). An advertiser is completely different since it doesn't watch video. There is an advertiser hub class which contains a reference to an Advertiser Selector class which functions like the Video Filterer class, allowing to select prospective advertisers. With this the advertising hub contacts the advertiser, which is its own class containing its own state about whether it should/could bid. The actual bidding is done in the advertising hub that way it can directly update the state through the advertising selector class.

My sequence diagram shows a very simple use case. First the user opens a connection to the Youtube server. The server iteratively asks the database to provide enough information for the server to send back a dynamic web page based on the state the database is holding about the user (ex - it's website history). The server generates a homePage and sends it back to the user. The user interacts with this HomePage and decides to search for a certain video. This generates a request to the server, who once again iteratively asks the db to provide video info. It then generates a search screen and returns in to the user. The user then clicks on a video, in essence asking for that video to be played. This request gets sent to the db which updates the video info and sends the URL back to the server. The server iteratively sends the URL and user INFO to the advertising hub. The advertising hub finds an advertiser who wants to pay to show a certain URL and sends it to the server (this would be its own sub use case). The server then sends the advertisement and video to the user.

For the activity diagram I focused mainly on what work happens when and factored out all the classes. The circle with the line shows the start state. The more oval shapes are actions done by the system, the circles with a line under them show actor actions. First the system displays a web page. The user then enters in a search term. The user generates results and displays them

on a search screen. The user clicks on a video. In response, the system generates a video screen, displays advertisements and then the video. In any of the user intervention steps I put arrows leading back to prior steps to show that the user can go back on any of the steps.

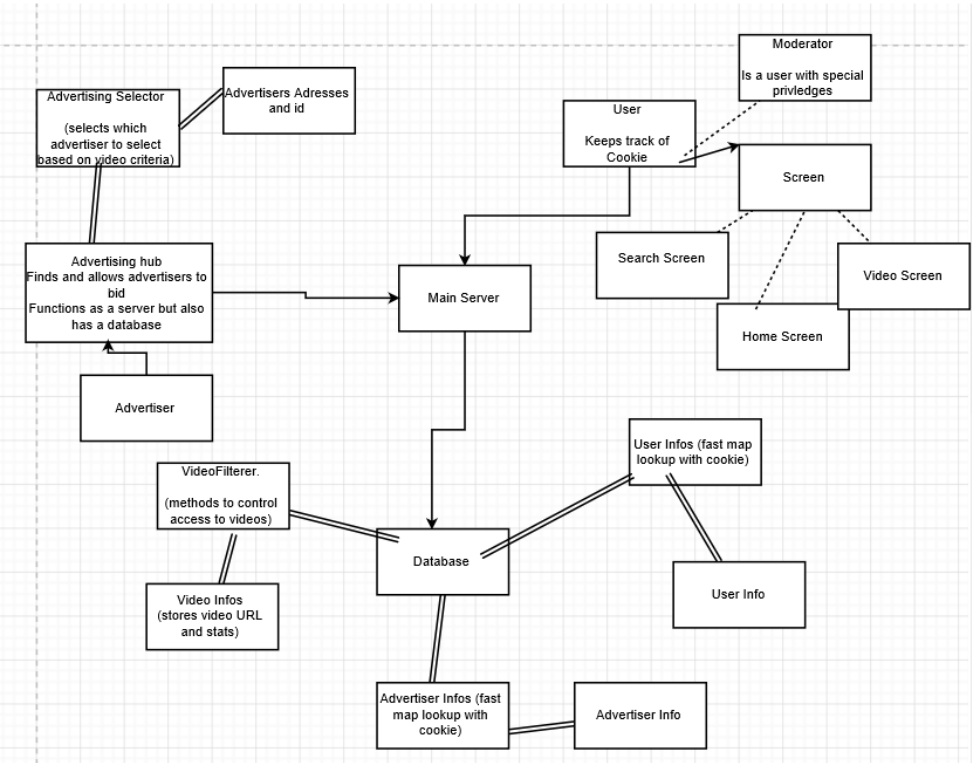
There is also naturally no end state since this process ideally would loop forever and if the user does anything to try to leave the vacuum he would break the system.

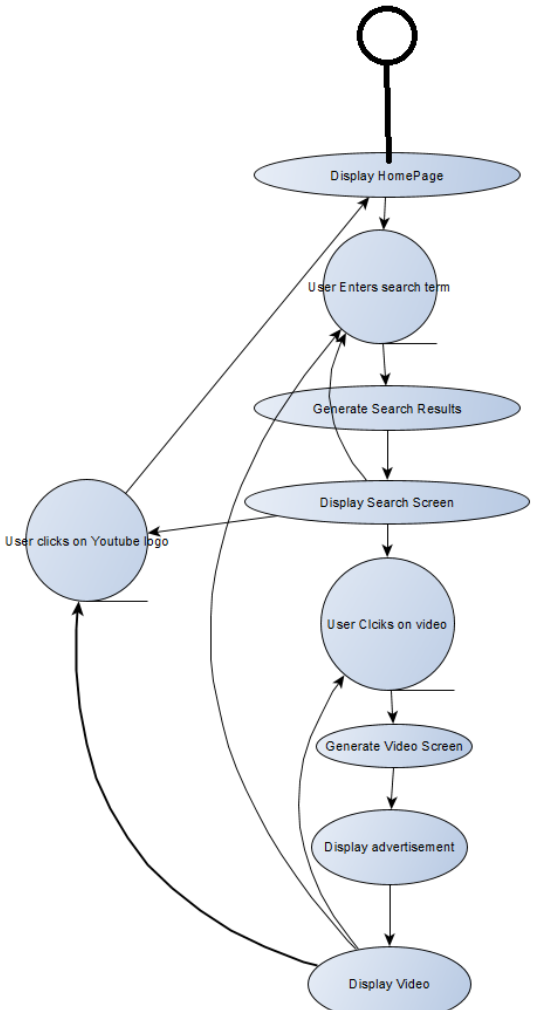
For my state diagrams, I made some of the states a bit more generic. The database for example, has a state where it forever loops until it gets a request, processes the request, and sends it back to the server. In essence, during there are different types of request and the way it processes them differ but I didn't think this type of diagram needed to show that. The other diagrams are pretty self explanatory given the sequence diagram. The user at any point in time is either generating a request to send to the server, or viewing a specific screen (home,search,video). The screen is mainly static but it has handlers to deal with user interaction and is what actually talks to the server since the user isn't going to do that directly. It's state is therefore showing the screen or responding to user interaction. The server is like the database in that it has a state waiting for a request, but unlike it, it also has to send a request to the database at each step so the step being done could be recorded. Also sometimes this request to the database may provide needed information. Therefore a state to wait for a reply from DB was necessary. I was debating with some of the states whether wait for an ack should be included in the "send info" and was somewhat inconsistent with that.

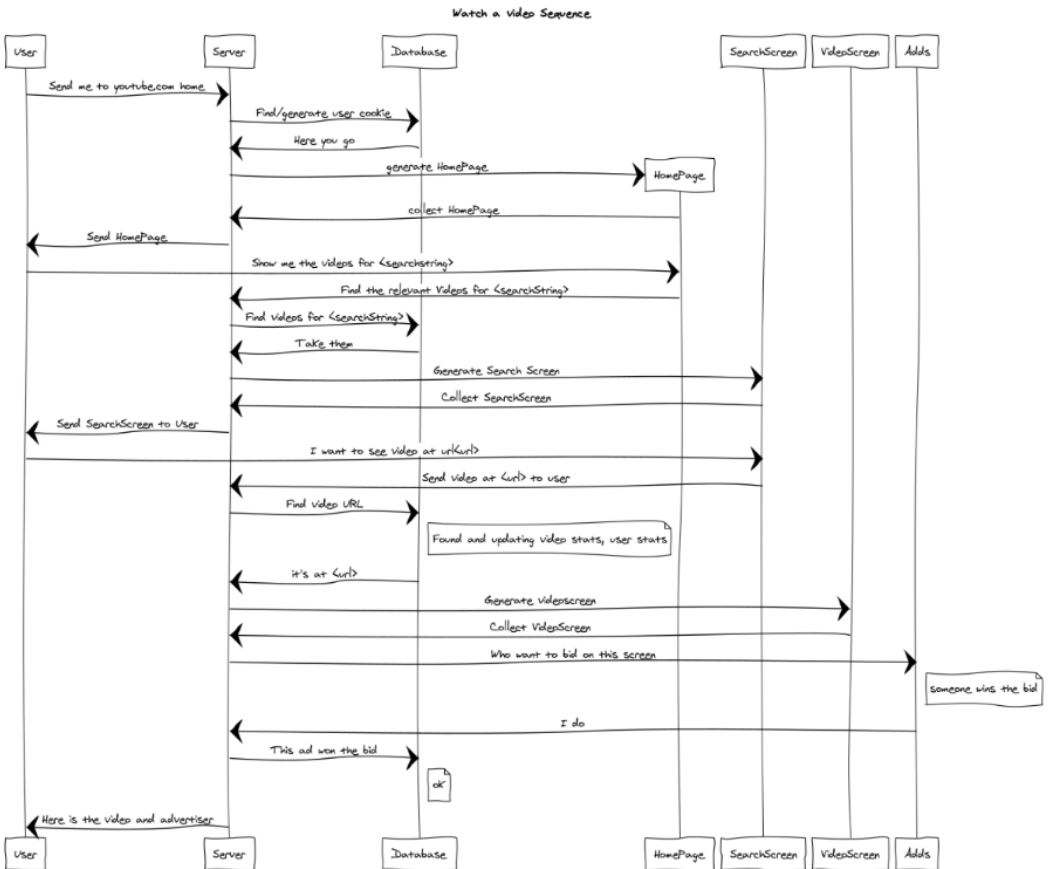
For my diagrams, I am going to use the following use case. This is the timeline in words of what will happen:

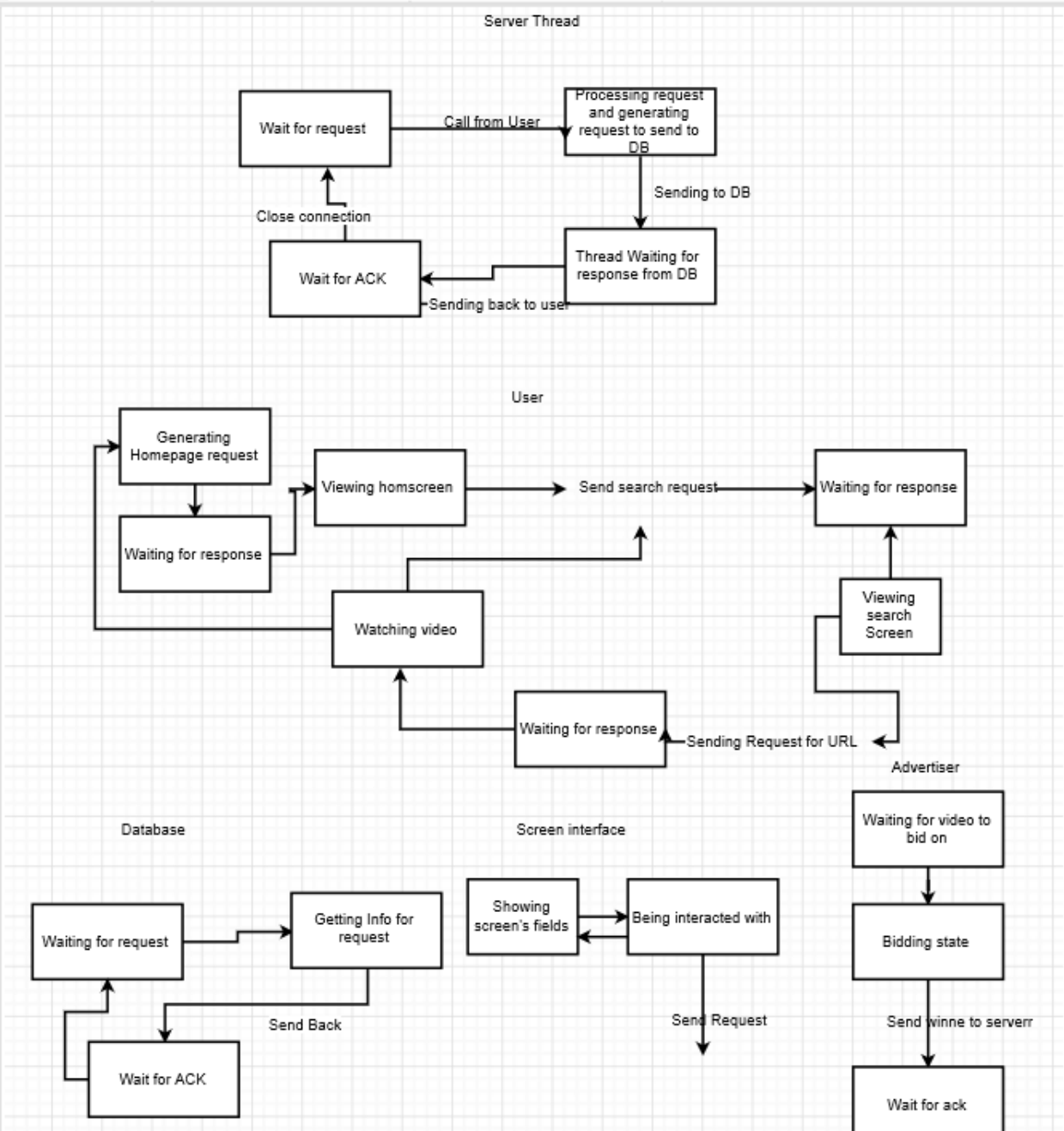
A user sends an http request to some youtube server. The server sends back a home page, consisting of a search bar, and recommended videos. The user searches with the search bar for a video. A list of videos pop up along with an option to filter. The user clicks on the first video. A video screen pops up, the video is pushed to the cdn if it isn't there, and is sent to the user and displayed in the video screen for the user to watch.

Class diagram - This shows the components and the relationship between these components.


Moderator
Is a user with special privledges
Advertising Selector
(selects which advertiser to select based on video criteria)
Advertisers Adresses and id
User
Keeps track of Cookie
Screen
Advertising hub
Finds and allows advertisers to bid
Functions as a server but also has a database
Main Server
Search Screen
Video Screen
Home Screen
Advertiser
User Infos (fast map lookup with cookie)
VideoFilterer.
(methods to control access to videos)
Database
User Info
Video Infos
(stores video URL and stats)
Advertiser Infos (fast map lookup with cookie)
Advertiser Info


Display HomePage
User Enters search term
Generate Search Results
Display Search Screen
User clicks on Youtube logo
User Clciks on video
Generate Video Screen
Display advertisement
Display Video


Watch a video Sequence
User
Server
Database
SearchScreen
VideoScreen
Adds
Send me to youtube.com home
Find/generate user cookie
Here you go
generate HomePage
HomePage
collect HomePage
Send HomePage
Show me the videos for <searchstring>
Find the relevant Videos for <searchString>
Find videos for <searchString>
Take them
Generate Search Screen
Collect SearchScreen
Send SearchScreen to User
I want to see video at url<url>
Send video at <url> to user
Find video URL
Found and updating video stats, user stats
it's at <url>
Generate VideoScreen
Collect VideoScreen
Who want to bid on this screen
someone wins the bid
I do
This ad won the bid
ok
Here is the video and advertiser
User
Server
Database
HomePage
SearchScreen
VideoScreen
Adds
www.websequencediagrams.com


Server Thread
Wait for request
Call from User
Processing request and generating request to send to DB
Sending to DB
Close connection
Thread Waiting for response from DB
Wait for ACK
Sending back to user
User
Generating Homepage request
Viewing homscreen
Send search request
Waiting for response
Waiting for response
Viewing search Screen
Watching video
Waiting for response
Sending Request for URL
Advertiser
Database
Screen interface
Waiting for video to bid on
Waiting for request
Getting Info for request
Showing screen's fields
Being interacted with
Bidding state
Send Back
Send Request
Send winne to serverr
Wait for ACK
Wait for ack