# System Design Online Photo Editor

FR : 1) The client side of application is used to open, view, apply processing/editing commands on the image.

2) Option to upload/save to/from local disk or cloud storage.

3) User can simultaneously edit image from multiple devices.

optional - 1) Share with others, 2) Undo/Redo actions

NFR : 1) All the services are High available, low latency i.e almost realtime.

Assumptions or clarifications needed - 1) The image processing commands will be execute serially and results are shown to user ASAP so that they can proceed to next action.

2) The image processing happens at server side. The client application is lightweight.

# APIs

1) String login( String userName, String password)
2) void loadImage(String userToken, File image)
2) File performAction( String userToken,FileID imageID, Integer actionID, List<String> parameters)

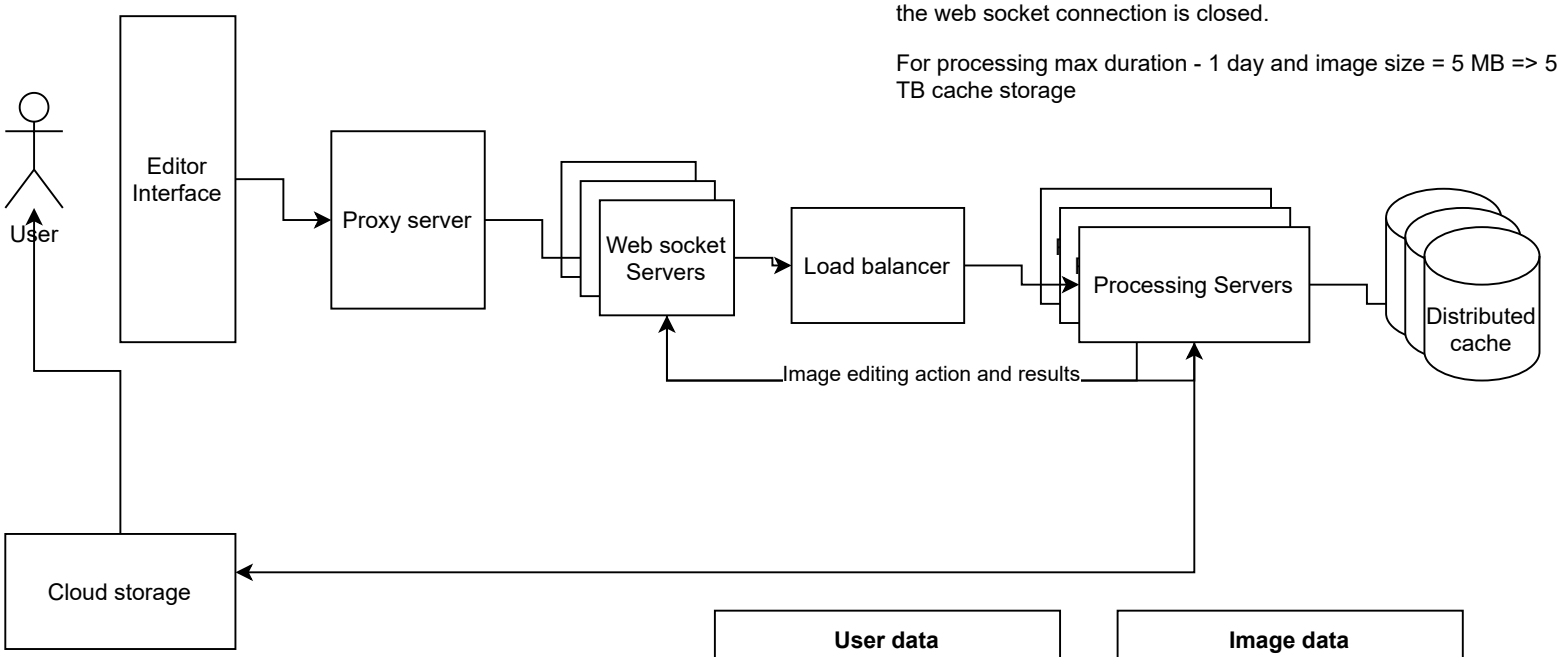
# Capacity estimation

Storage - Assume images are stored temporarily for processing by the next action inline. The final image storage is done at third party cloud service.

Assume 1 M image edits per day.

To save B/W. Once the image is initially loaded, the subsequent data can transferred to client in low quality. High quality image is stored by processing server to cloud storage automatically once the web socket connection is closed.

For processing max duration - 1 day and image size = 5 MB => 5 TB cache storage




| User data | |
|---|---|
| **PK** | **User ID** |
| | UserName |
| | Last edit Image ID |

| Image data | |
|---|---|
| **PK** | **Image ID** |
| | Image URL |

# Control flow -

1) Open image- user opens the image from a cloud server or local storage.

2) Once the client opens the image, a web socket connection is established so that user edits and the resulting image can be transferred seamlessly without opening/closing session everytime.

3) The load balancer is responsible to assign processing server to the current user and image. The processing server handles the image editing tasks. Preferably sticky session is used so that latency is reduced.

4) The processing server uses distributed cache to store intermediate results so that if one goes down, other can pick up from where left.

5) When the user goes offline or switches to another device, the result is stored to cloud storage and cache is cleared. a new web socket connection is established and user is promoted if they want to start from where left. If they choose yes then the cloud image is loaded on the client and the server.