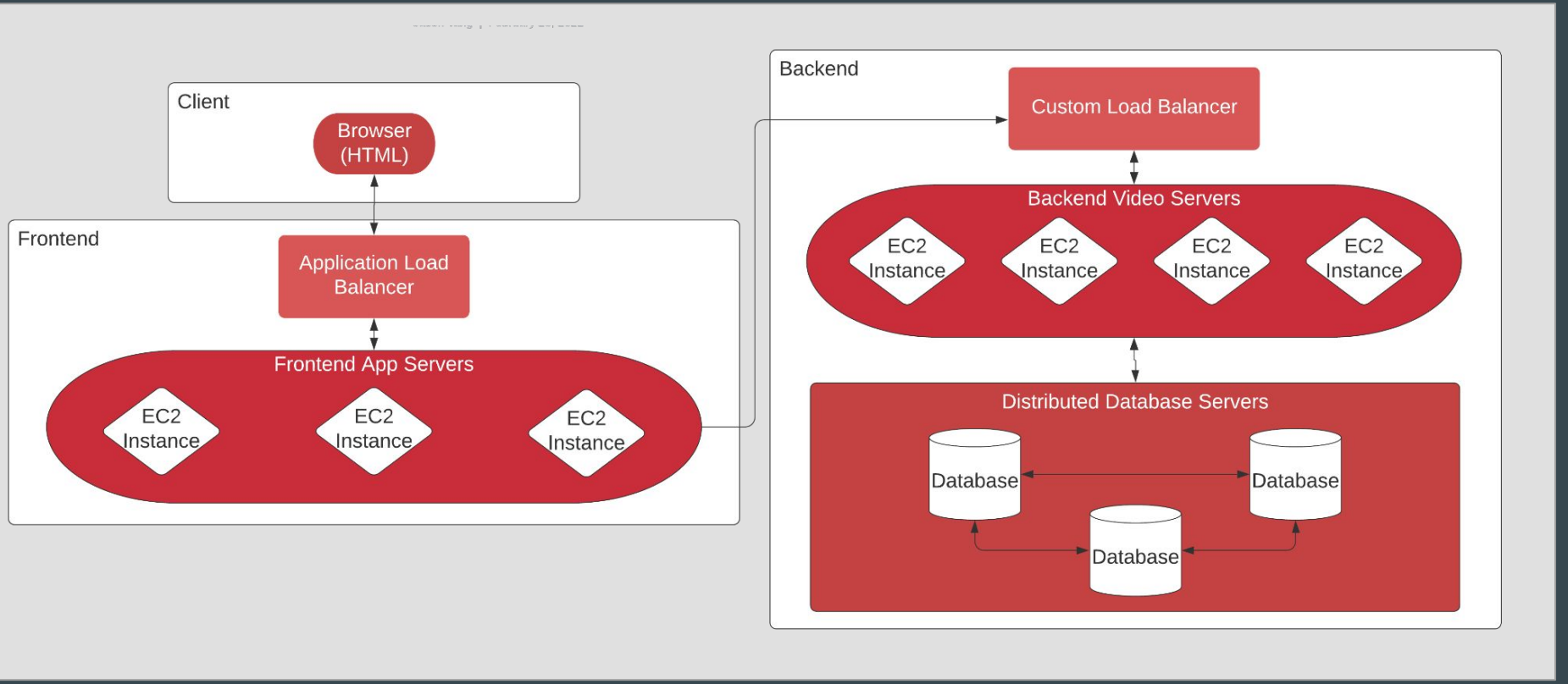
Use Case

- Load balancing is a key server architecture process
- Problem: insufficient public knowledge on effective dynamic load balancing
  - Commercial options only offer “black-box” solutions
  - Useful for building scalable and more integrated open-source-based servers

- Solution: Testing and documenting performance of LB algorithms of interest in a real world environment video streaming server
  - Comparing traditional parameter-based LBs and experimental ML models

Server Architecture



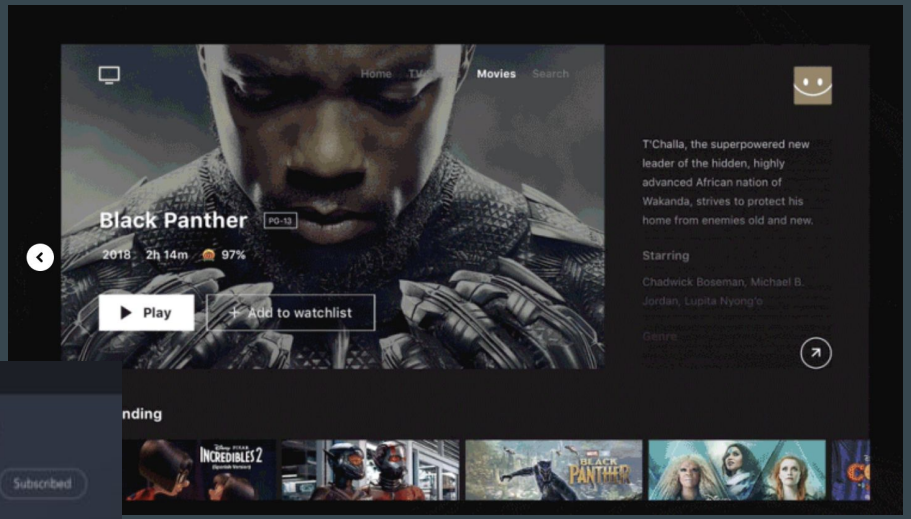
Picking the right VM

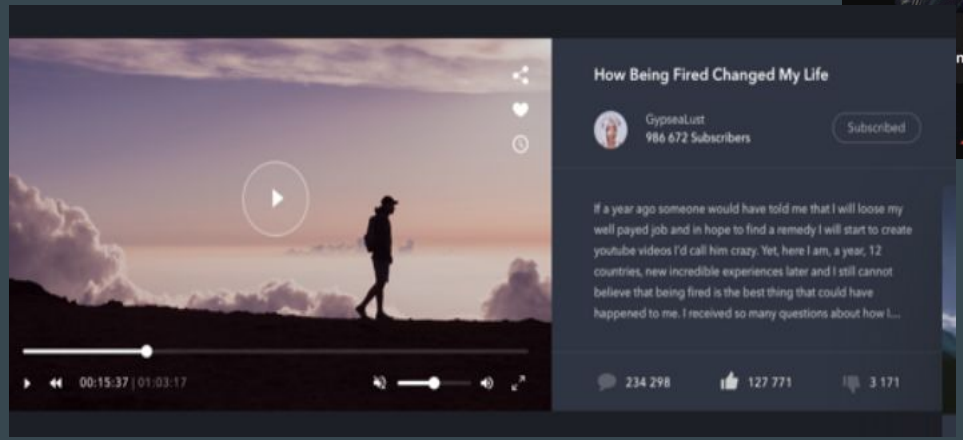
- Between Azure and AWS we chose to run our instances on AWS
  - We have experience deploying web apps on EC2 instances

- Several types of EC2 instances defined by the technical specifications of the VM
  - General purpose, storage optimized

- We will be leveraging different general purpose EC2 instances with the same specifications (e.g. T4g for backend, T3 for frontend, H1 for database)

Webpage Layout/Functionality

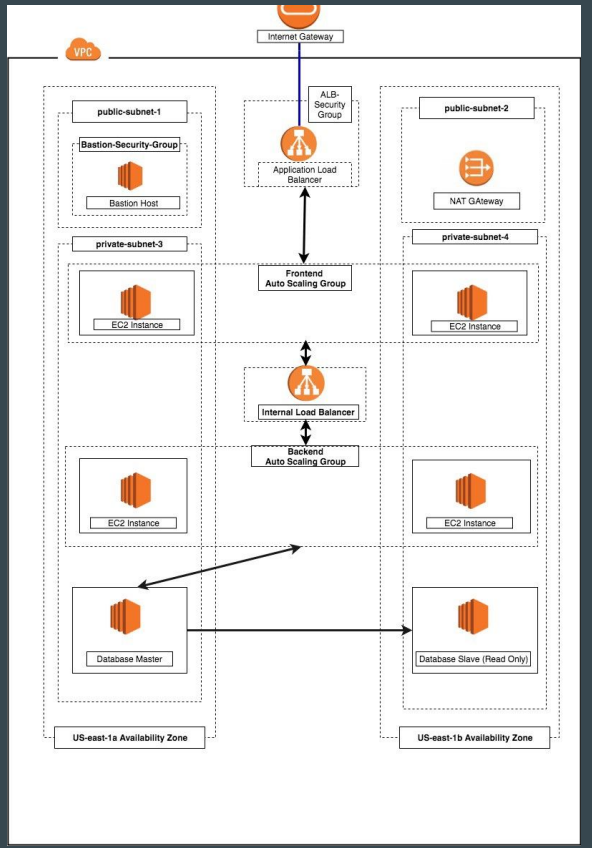




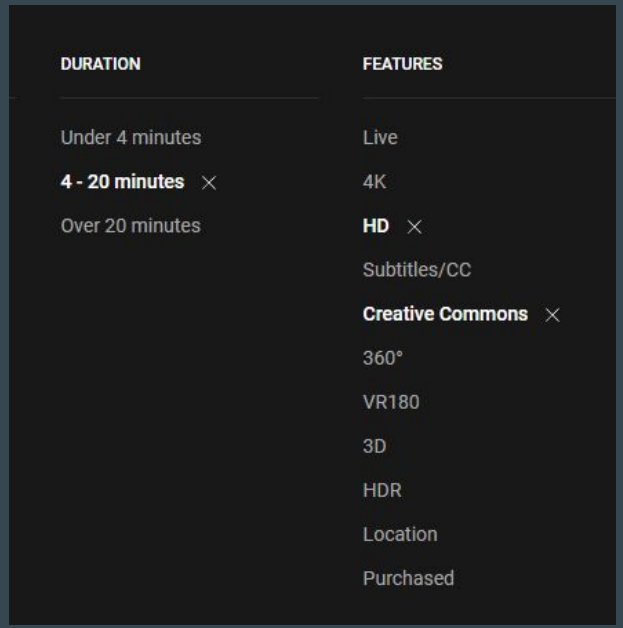
Load Balancer Simulation

- One load balancer b/w front-end and back-end, another b/w back-end and database

1. Receive request from tier either above or below
2. Request+receive parameters from target tier
3. Apply parameters to model -> determine target node
4. Send request to target node
5. (optional) Receive feedback from target node for updating model



Video Specs/Source



- Stream 1080p videos at 60 fps that are ~10 minutes (8-12)
- Upgrade to 4k videos if more load on LB is needed
- Use a catalogue of Creative Commons videos from Youtube
- CC license allows republishing without copyright issues
- Dynamic video that changes a lot to put sufficient load
- Gaming videos, stock videos

Initial LB Algorithms (Mitul)

Name	Parameters	Brief Description
Round Robin	N/A	Chooses nodes in specific sequence (e.g. 1, 2, 3, 1, …)
CPU Decider	VM specs, CPU usages	Chooses node with lowest current CPU utilization
Response-based UCB1	VM response times	ML model tracks recent response speeds and chooses fastest or random based on log chance
TBD	VM specs, Request sizes	Another reinforcement-learning-based algorithm

Quantitative Specifications

Metric	Benchmark	Description
Bit Rate	3.5 MB/s	How many bits is transmitted over a specified time
Lag Ratio	2.5%	Waiting time over watching time of the video
Buffer Fill	< 2s	Time it takes for video to load at the start
User Latency	6s	Time elapsed between the video request from the user and receiving video data from the server
LB Latency	500ms	Time elapsed after the request leaves the LB until a response from the target is received
CPU Usage	80%	Percentage of capacity of VM’s in use by the system

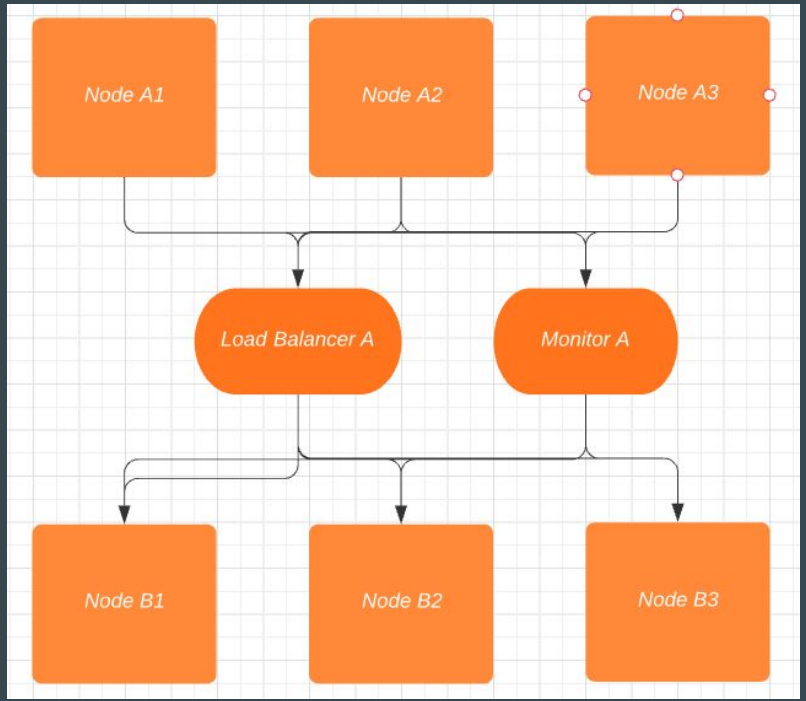
Testing Plan - User Simulation

- Run python scripts on 30 devices/VMs to simulate users
- Metrics averaged per test and compared with benchmark for each algorithm
- Users will pick random videos from our application to send requests for
- Ensures requests that put varying loads on servers

User Behavior	# of users	requests/min	Description
Full-Length	10	~1 / 10 min	Watches videos to its full length
Half-Length	10	~1 / 5 min	Watches videos to half length
Quick-Swap	10	~5 / 1 min	Switches between different videos in quick succession

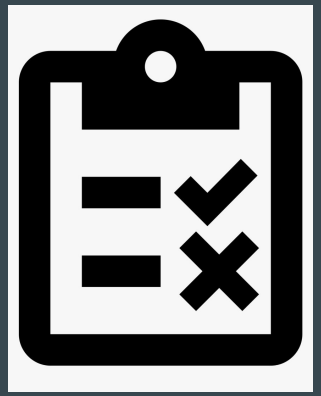
Monitor Node

- Each LB will have an accompanying monitor node situated in an adjacent position
- They will collect data on the LB performance via response from adjacent tiers
  - Local response time
  - CPU utilization variance

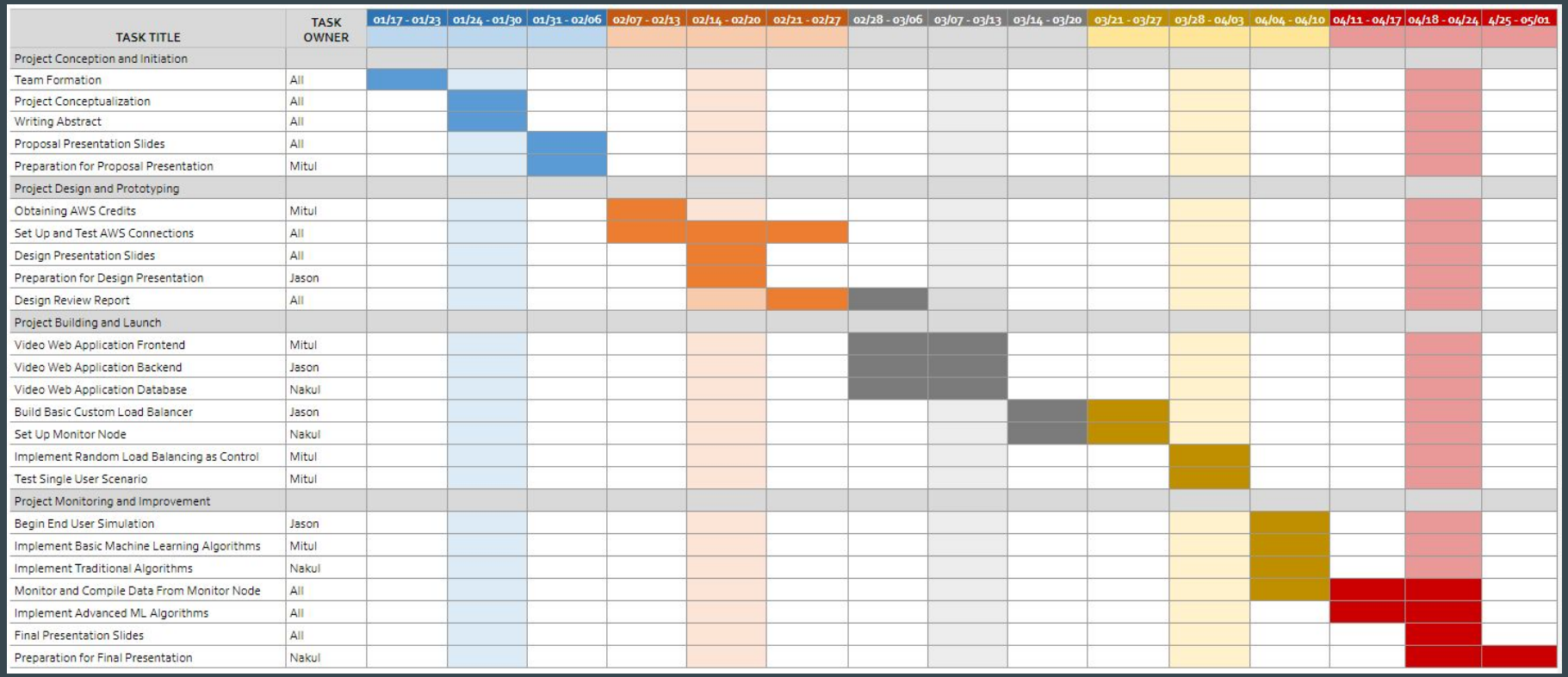


Test Evaluation

- Metrics of our video streaming application benchmarked against load balancer that randomly picks servers (RandomLB)
- Metrics also have to meet our specifications
- Potential sources of failure:
  - Not generating enough load for making LB consequential
  - Not having significant improvement over randomLB
- Can be solved by reducing testing load/using better VM's



Task Management



TASK TITLE	TASK OWNER	01/17 - 01/23	01/24 - 01/30	01/31 - 02/06	02/07 - 02/13	02/14 - 02/20	02/21 - 02/27	02/28 - 03/06	03/07 - 03/13	03/14 - 03/20	03/21 - 03/27	03/28 - 04/03	04/04 - 04/10	04/11 - 04/17	04/18 - 04/24	4/25 - 05/01
Project Conception and Initiation																
Team Formation	All															
Project Conceptualization	All															
Writing Abstract	All															
Proposal Presentation Slides	All															
Preparation for Proposal Presentation	Mitul															
Project Design and Prototyping																
Obtaining AWS Credits	Mitul															
Set Up and Test AWS Connections	All															
Design Presentation Slides	All															
Preparation for Design Presentation	Jason															
Design Review Report	All															
Project Building and Launch																
Video Web Application Frontend	Mitul															
Video Web Application Backend	Jason															
Video Web Application Database	Nakul															
Build Basic Custom Load Balancer	Jason															
Set Up Monitor Node	Nakul															
Implement Random Load Balancing as Control	Mitul															
Test Single User Scenario	Mitul															
Project Monitoring and Improvement																
Begin End User Simulation	Jason															
Implement Basic Machine Learning Algorithms	Mitul															
Implement Traditional Algorithms	Nakul															
Monitor and Compile Data From Monitor Node	All															
Implement Advanced ML Algorithms	All															
Final Presentation Slides	All															
Preparation for Final Presentation	Nakul															