SD08 - Sync & Split

Ben Smith, Connor Ferris, Gannon Guess, Will Schwartz, Yi Yun Khor

Our Topic

Overview

- “Sync & Split”
- SYNC: Sync up two videos with similar time and location
- SPLIT: Detect “significant events” and create clips



Specifics

- Multi-platform (Expo)
- Multi-tier
- Lots of computer vision/AI applications
- We are setting most requirements for the project



Big Picture Challenges



- **<u>There is no existing/available technology that does all of this!</u>**
- Similar SYNC processes:
  - Adobe premiere - audio waveforms
  - YouTube Content ID - audio/visual fingerprinting (avoid dupes)
- Similar SPLIT processes:
  - None, really
  - PySceneDetect - shot boundaries & scene change detection
  - YOLO + OpenCV - region detection
    - We are using these. But it is difficult to automate them





PySceneDetect

Big Picture Challenges pt. 2

- Use case
  - Who would be using this technology - many answers
- Submitted videos could be anything
  - Different sizes, lengths, bitrates
  - Animated?
  - Unrelated? No overlap
- What qualifies as a significant event?
- We are mostly focusing on roads, cars & intersections.



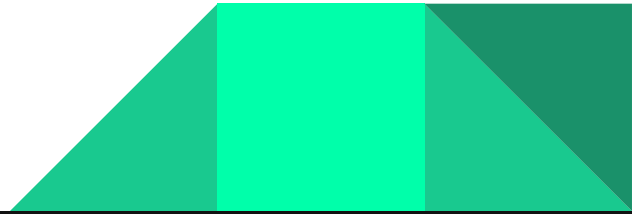
Video Storage

Video Storage

- Restriction:
  - No external cloud storage used (e.g. AWS S3, GCS)

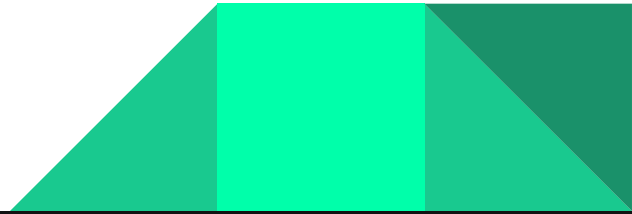
- Solution:
  - Videos stored locally on the server (in /videos/<username>/)

-



Video Storage:

- The Translator/Engine - Gunicorn:
  - Runs Django and handles requests
  - Communicates with Nginx
- The Gatekeeper/Entry Point- Nginx:
  - Forwards request to Django (through Gunicorn)
  - Serves video files directly from the server



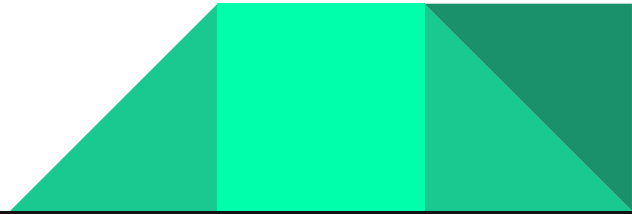
Video Uploading workflow:

[User] → Nginx → Gunicorn → Django → MySQL (metadata)

↓

Static File

/videos/<username>/<title>.mp4



Server-side (Video storage)

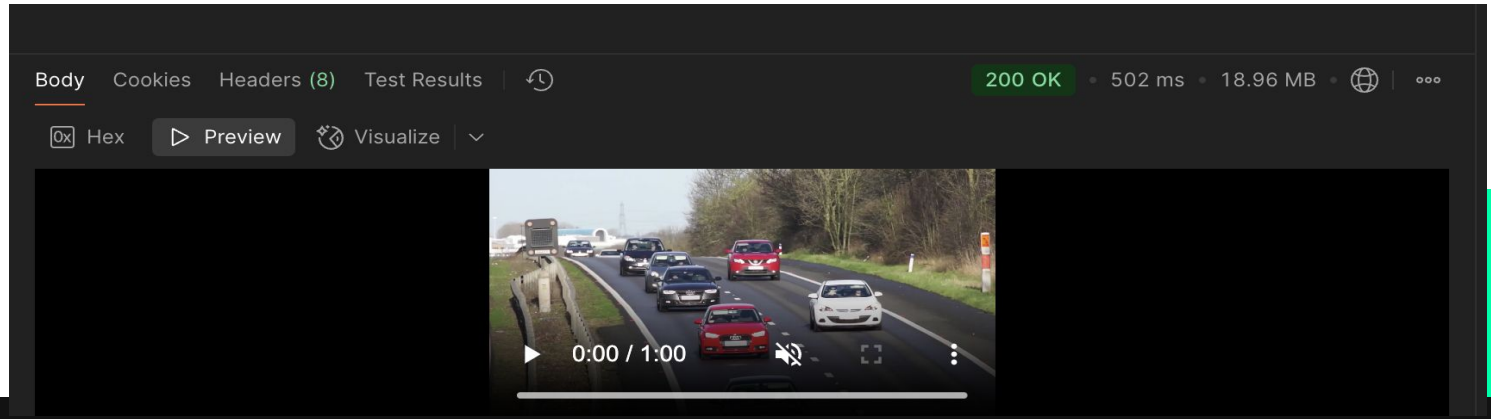
```
Last login: Thu Mar 27 22:48:36 2025 from 10.64.103.54
[belle27@coms-4020-008 ~]$ cd /home/gitlab-runner/app/videos
[belle27@coms-4020-008 videos]$ ls
roundtrip  roundtrip1234  testroundtrip  testroundtrip2  testroundtrip3  testuser
[belle27@coms-4020-008 videos]$ cd roundtrip
[belle27@coms-4020-008 roundtrip]$ ls
1900-151662242_small.mp4
```

Postman-side (Video Uploading)


```
    "data": {
        "id": 22,
        "title": "videotesting",
        "video": "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/videotesting.mp4",
        "created_at": "2025-04-03T23:39:28.016227Z"
    },
```

Postman Side (Video Retrieval)


```
videos": [
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/1900-151662242_small.mp4",
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/1900-151662242_small_NoLmtef.
        mp4",
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/left-video.mp4",
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/left-video_MKv3nqG.mp4",
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/right-video.mp4",
    "http://coms-4020-008.class.las.iastate.edu/videos/testroundtrip3/1900-151662242_small_Bra9MSI.
```

Challenges & What We Learned in Video Storing

- Nginx Was Not Set Up To Serve Video Files
  - Added a new rules to the Nginx configuration

```
location /videos/ {
    alias /home/gitlab-runner/app/videos/;
    add_header Access-Control-Allow-Origin *;
}
```

```
location ~ ^/videos/(upload|delete|videos)/ {
    proxy_pass http://127.0.0.1:8000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}
```

- File permission errors (403 Forbidden from Nginx)
  - Solved by setting correct read/execute permissions for video folders and files

```
sudo chmod o+rx /home
sudo chmod o+rx /home/gitlab-runner
sudo chmod o+rx /home/gitlab-runner/app
sudo chmod o+rx /home/gitlab-runner/app/videos
sudo chmod o+rx /home/gitlab-runner/app/videos/testroundtrip3
```

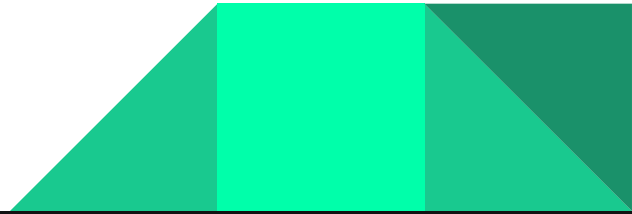
Challenges & What We Learned in Video Storing

- VM/ server is down
  - The entire video storage system becomes unavailable — no uploads, no playback
- Speed of retrieving the video from vm
  - The server has limited bandwidth
  - Multiple users request large video files at once

Syncing Process - In Depth

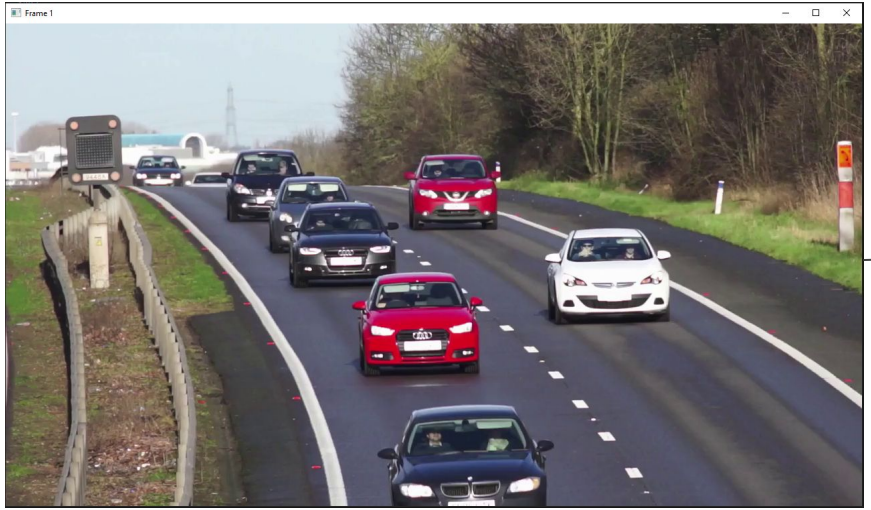
Syncing process overview

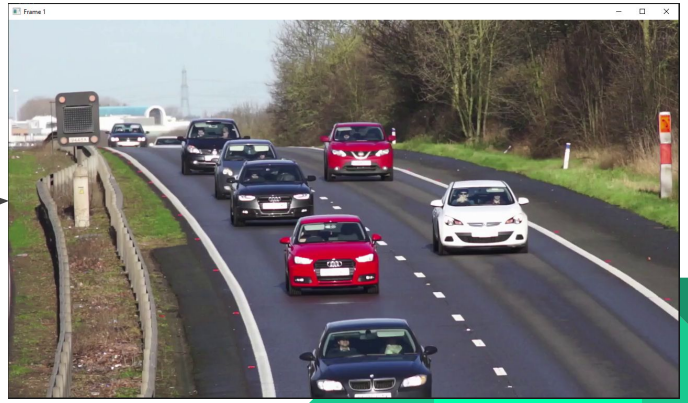
- Multi-step process of fitting two videos together
- Written in Python
- User selects two videos
- Extracts data from the videos
- Reformats the videos
- Uses the data to sync the videos side by side
- Outputs synced video



Preparing to sync

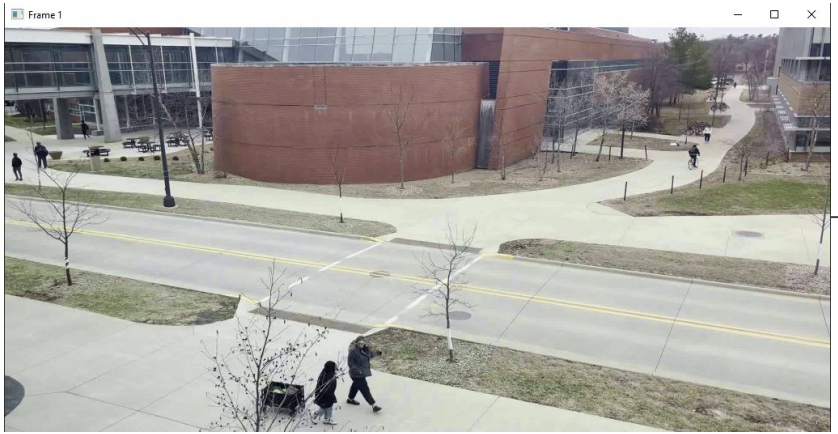
- Resizing the videos
- Selecting landmarks
- Resampling video

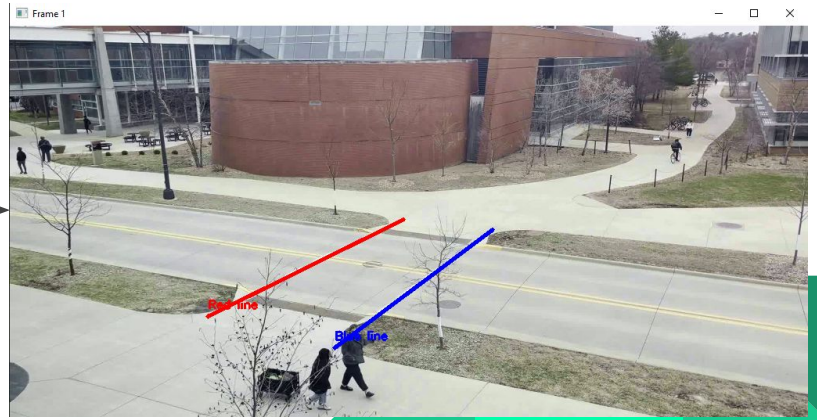




Preparing to sync

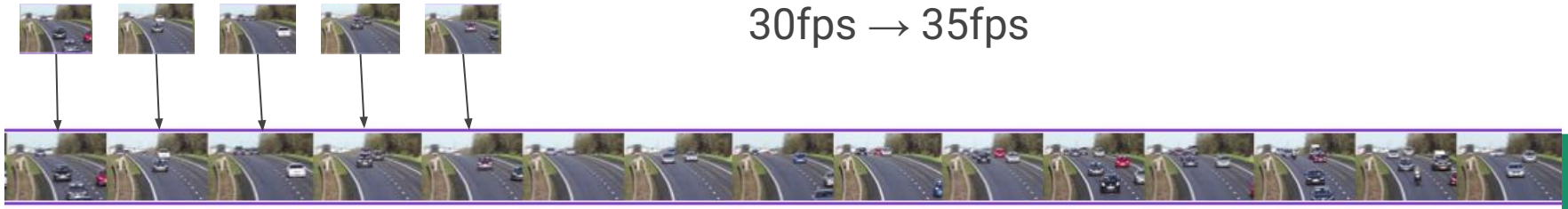
- Resizing the videos
- Selecting landmarks
- Resampling video
- Extracting the car data





Preparing to sync

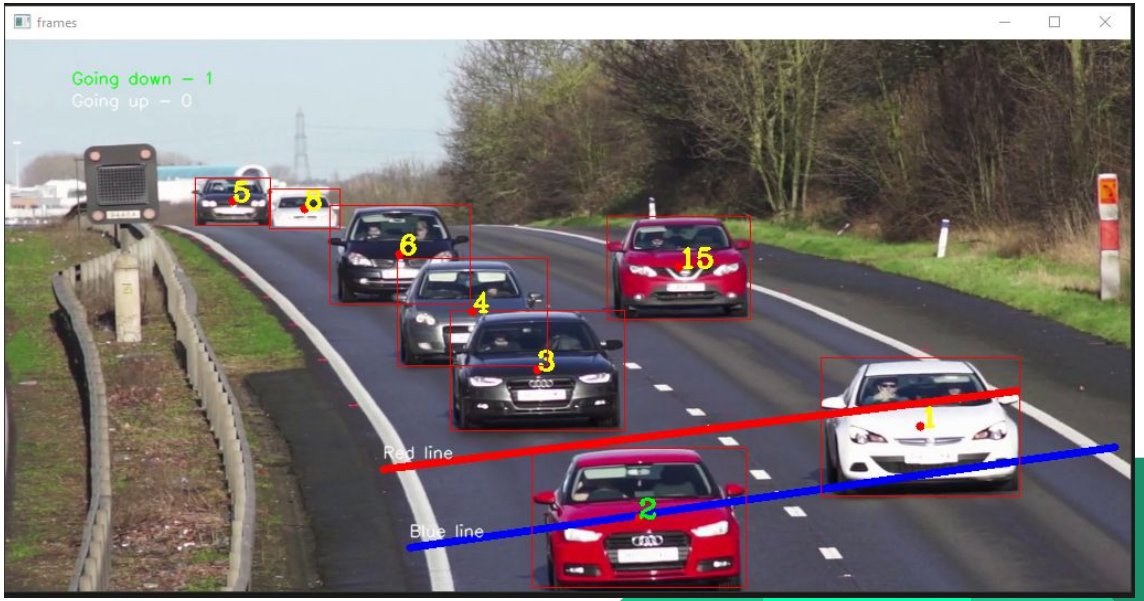
- Resizing the videos
- Selecting landmarks
- Resampling video
- Extracting the car data



30fps → 35fps

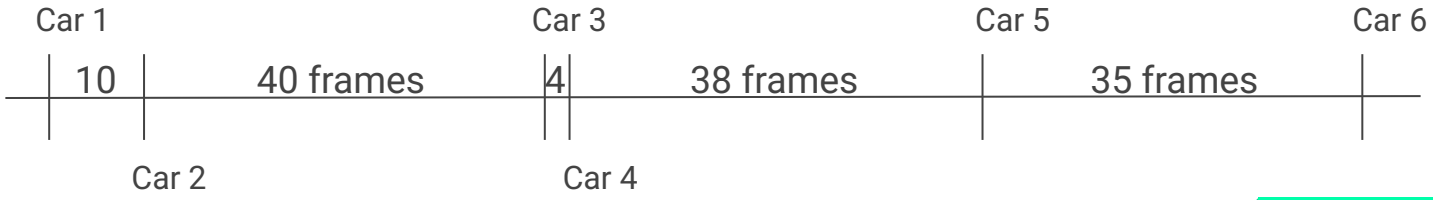
Preparing to sync

- Resizing the videos
- Selecting landmarks
- Resampling video
- Extracting the car data



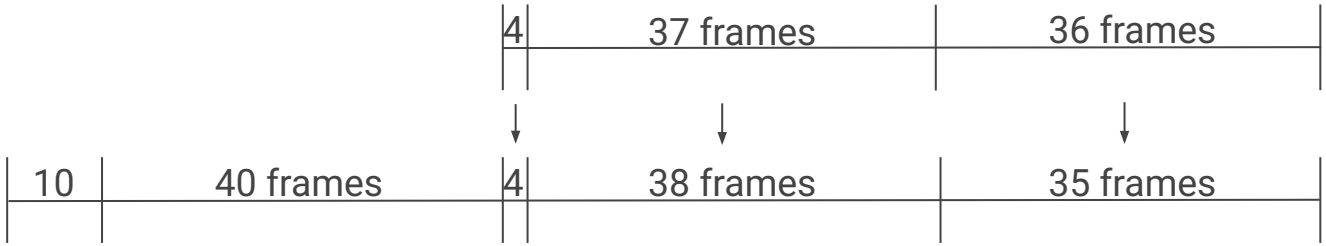
Syncing the videos

- Creating time interval list
- Best fit algorithm
- Creating synced video



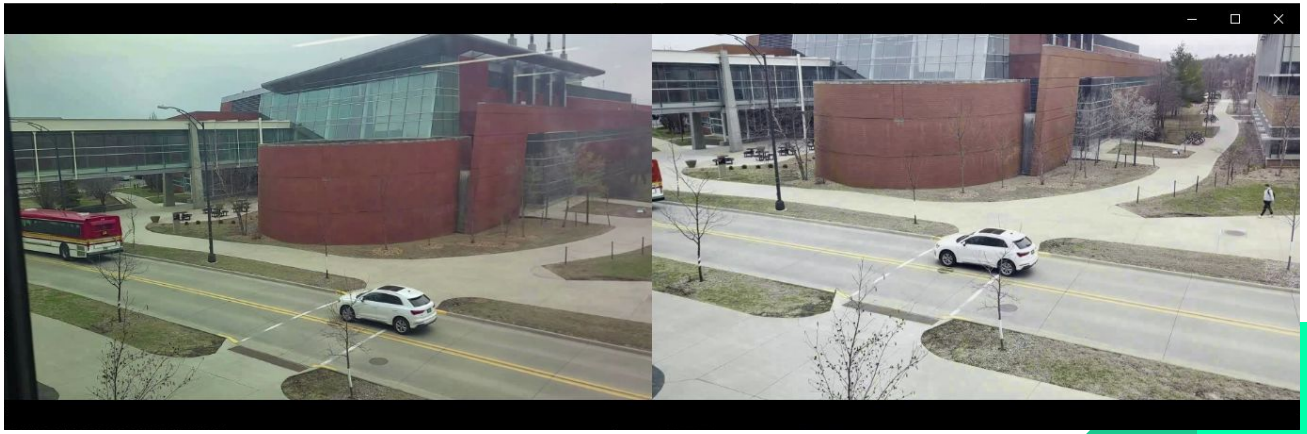
Syncing the videos

- Creating time increment list
- Best fit algorithm
- Creating synced video



Syncing the videos

- Creating time increment list
- Best fit algorithm
- Creating synced video



Split - Detecting Important Events

- You Only Look Once - Object detection
- OpenCV - display object detection to user
- Inside of sync videos process
- Track frames and split from videos based on the frames provided
    - If one video detects the event but the other doesn’t both will be included in the split

Current Important Events

- Abnormal speed
  - Save time and space by using data from video syncing
  - Doesn’t need constant speed calculations that would eat into video process time
  - Store frames that the events(cars) cross the event lines
  - Use statistics formula to find outliers



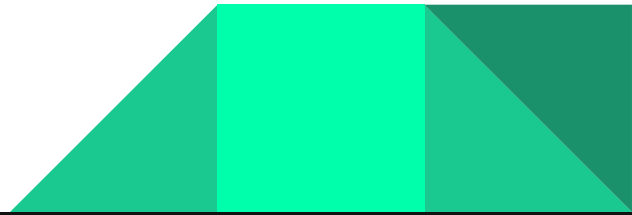
Additional Important Events

- Some events need OpenCV or YOLO to be identified
  - Car crashes
  - Driving on curb
  - Running of stop sign/red light



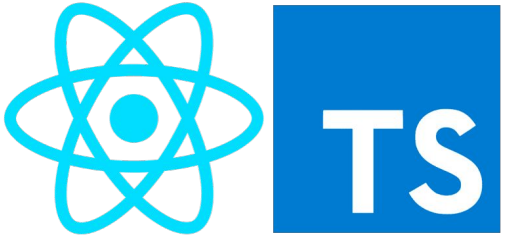
Challenges with Split and Event Detection

- Balancing features with runtime
- Client’s goal is to have a quick video process time
  - Ensures user has a good experience using the app
  - Continue use of our app



Frontend Design

Frontend Overview



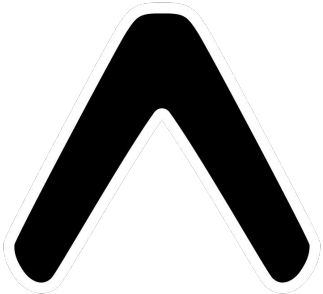
React Native

- Web and mobile application building

With Typescript

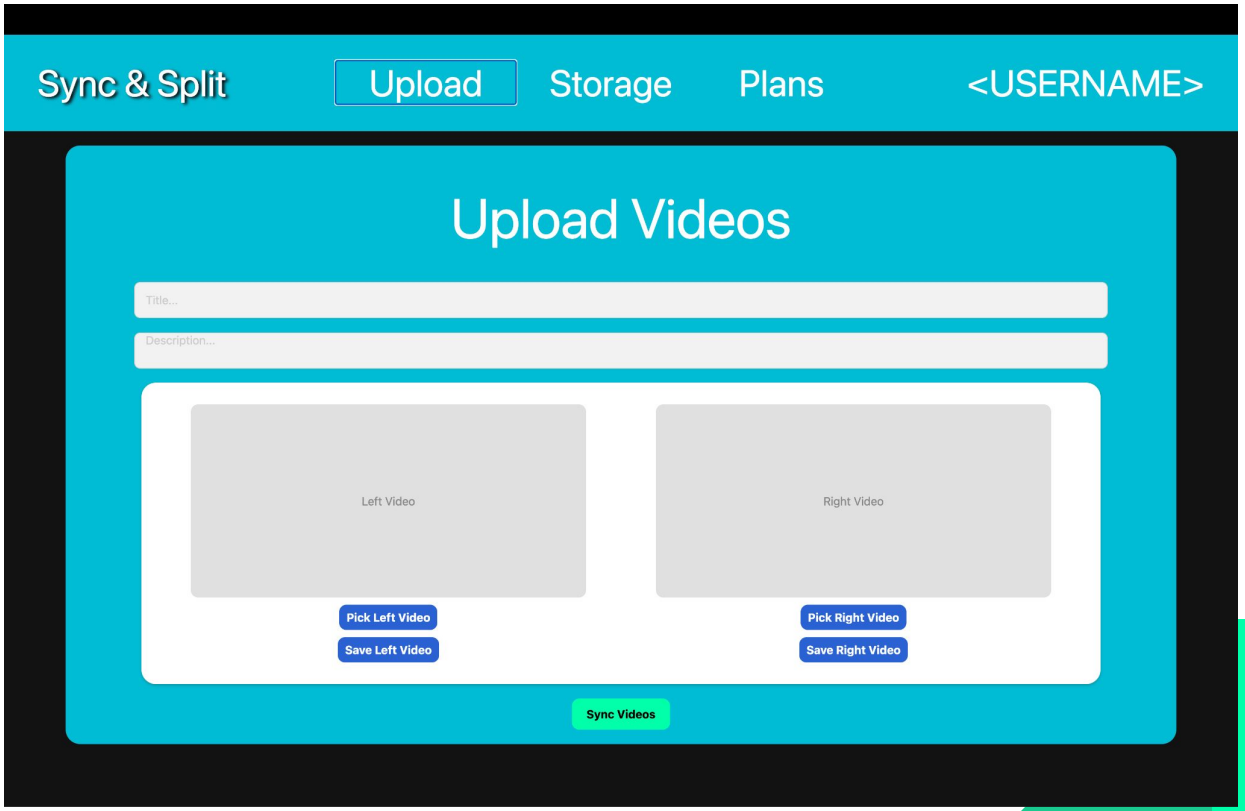
- Scalability, better error checking over JS

And Expo

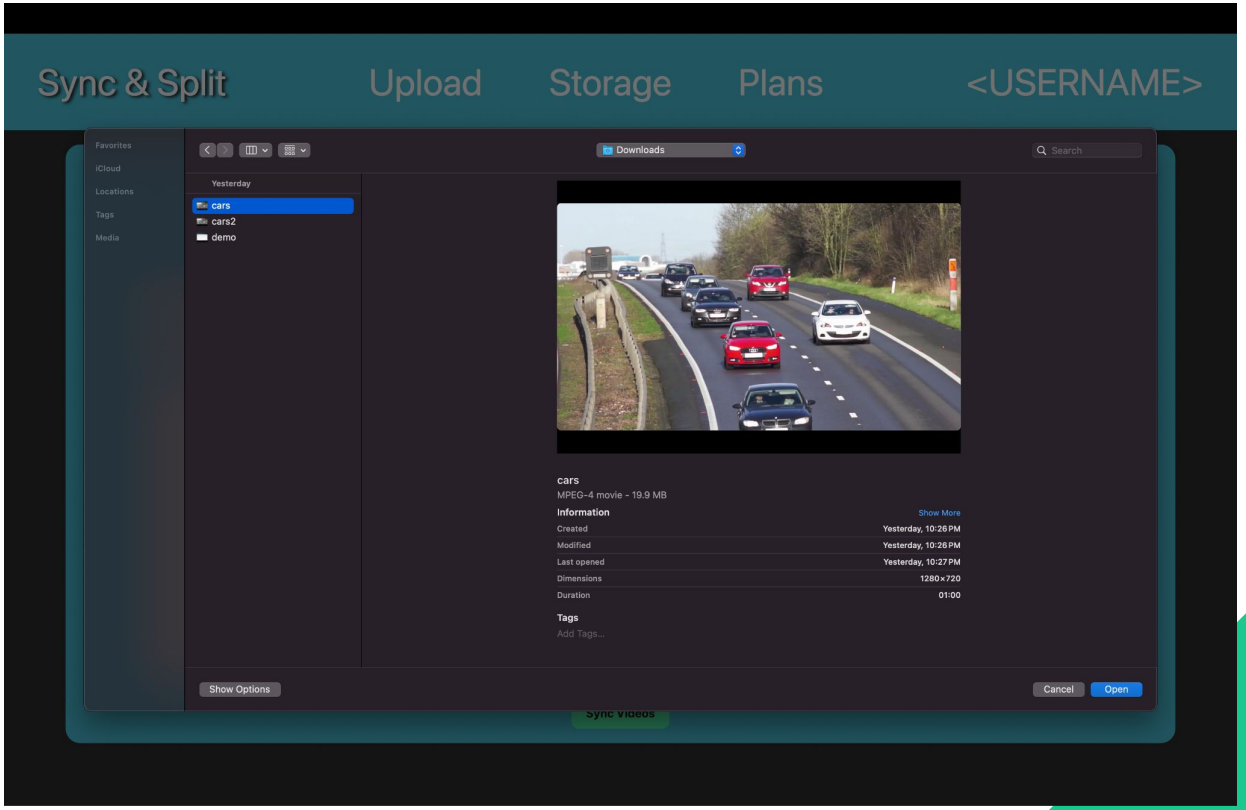


- Media and File System APIs

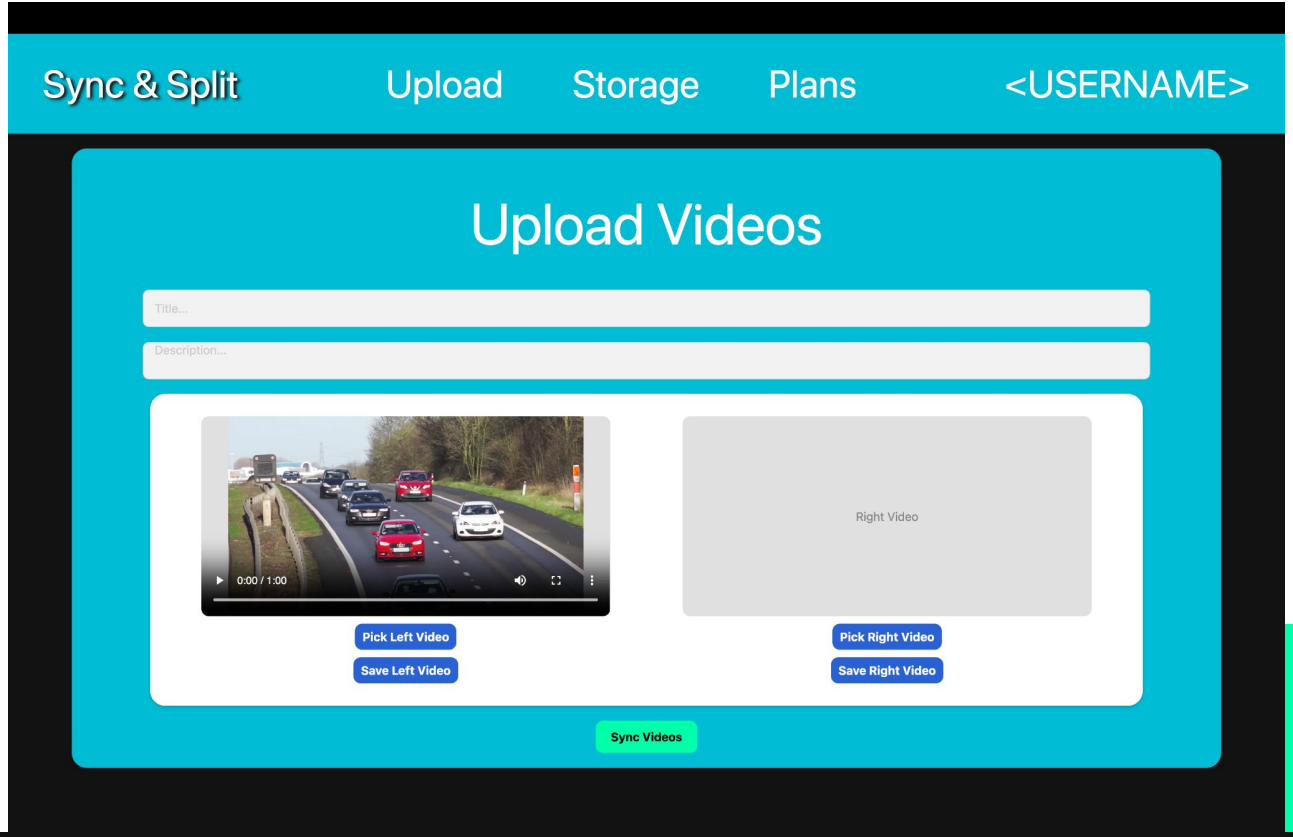
Video Uploads



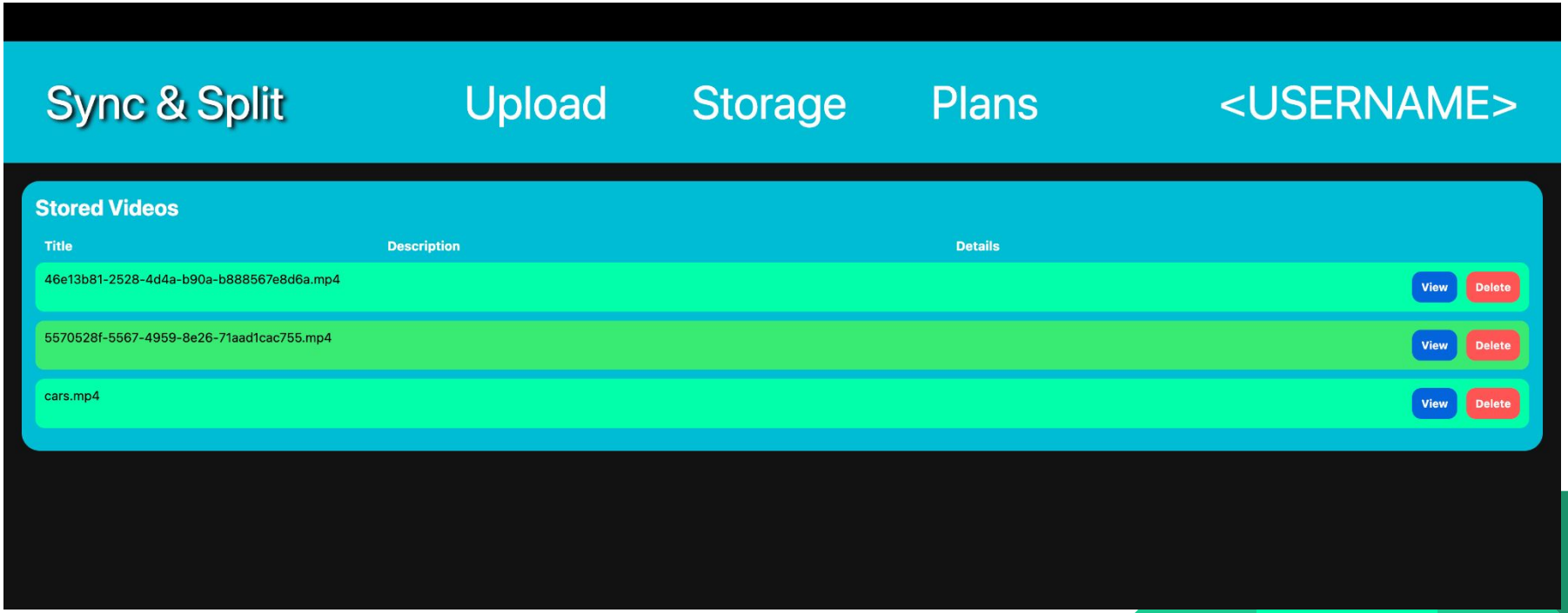
Expo Video Picker



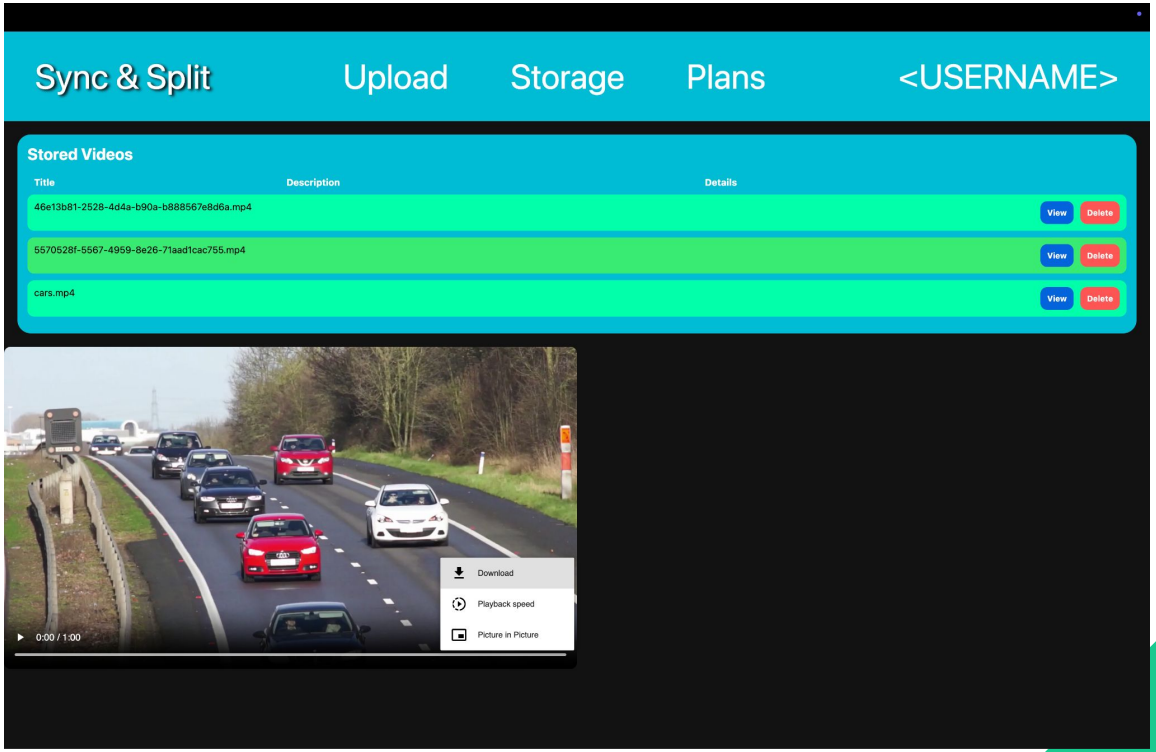
Video Preview



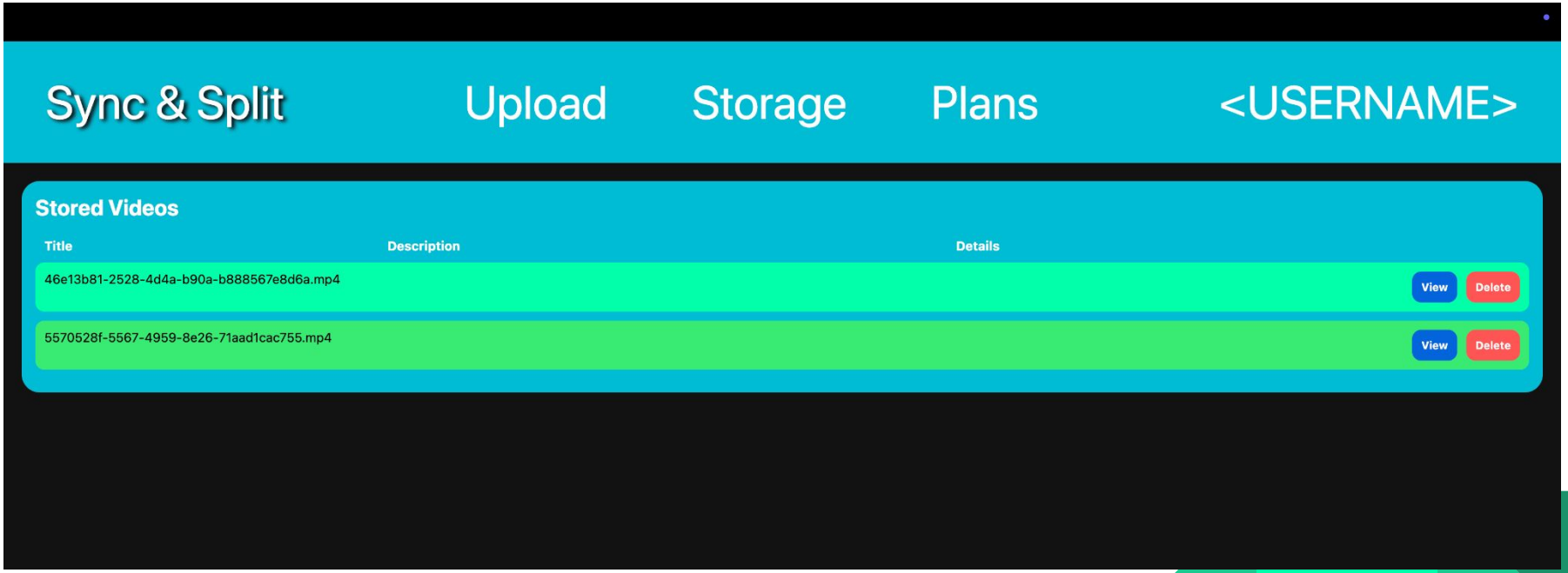
Storage Page



Viewing Videos



Deleting Videos



Remaining Work Timeline

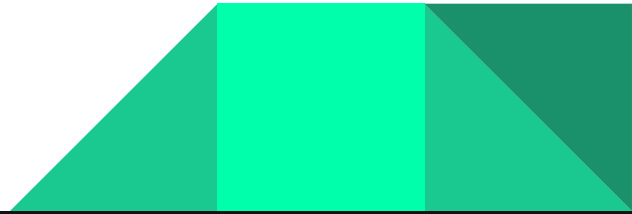
Currently In Dev

Backend:

- Fine-tuning sync
- Setting flags for splitting

Frontend:

- Separate page for viewing clips
- Pairing product videos with component videos
- Event success/failure alerts
- Plans page



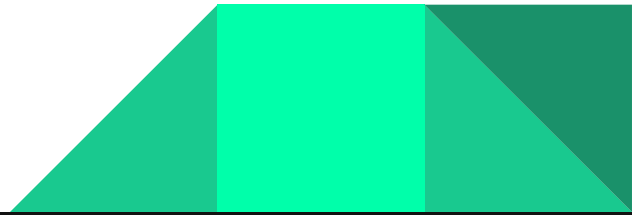
What Else?

High Priority:

- Connect “Sync & Split” button to Sync & Split method
- Automate the Sync & Split process
- User tiers and permissions
- Display video details and description

Low Priority:

- Tutorial/Demo page
- Mobile view/css (prioritizing features)



Demo