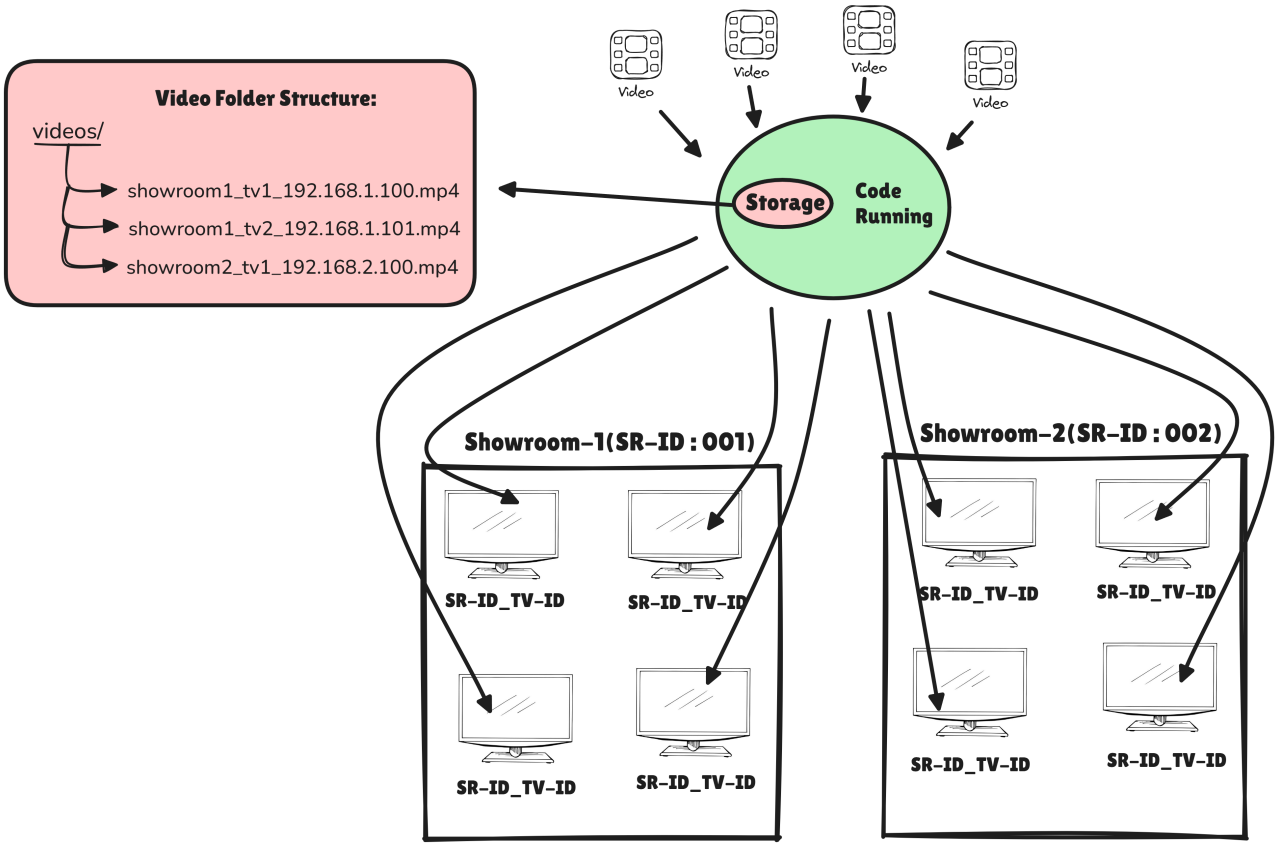
The Problem to Solve :

we are trying to make a central hub which decides what to be displayed on the TV's in the showrooms across the locations.

Quick requirements summary :
1. what need to be displayed on TV’s in showrooms must be controled by central hub.
2. in the process we have to make sure that whenever a TV get disconnected to central hub or internet, video must continue to play that means we have to store a video or two atleast in local storage of TV.
3. deploying a tracking mechanism in the TV to get live status of the TV weather video is playing or not, what’s the storage status of TV.
4. make a dashboard we shows the live status.

Our resources :
1. videos to be displayed on TV’s(central Hub will begetting that)
2. central Hub is basically a server(that we have)
3. TV’s static ID and credentials.
4. raspberi pi mudule is their with TV(i think not sure. Mrityunjay sir told it is present.)



1. Central hub/server will be recieving videos in specific folders or with specific naming(SR-ID_TV-ID)

2. establishing the connection between central Hub and TV

3. we will be sending those videos to their specified TV(static IP) via established connection.

4. play recieved video on TV.

1. first task is to build central hub system(relatively easier task)

below function is an excerpt from code at central hub. this function send video to TV give TV_IP and a port opened at TV .

```python
def send_video_to_tv(self, video_path, tv_ip):
    """Send video file to TV via HTTP POST"""
    try:
        url = f"http://{tv_ip}:8080/upload"

        logger.info(f"Sending {video_path} to TV at {tv_ip}")

        with open(video_path, 'rb') as video_file:
            files = {'video': video_file}
            data = {'filename': os.path.basename(video_path)}

            response = requests.post(
                url,
                files=files,
                data=data,
                timeout=300  # 5 minutes timeout for large files
            )

            if response.status_code == 200:
                logger.info(f"✓ Successfully sent video to {tv_ip}")
                return True
            else:
                logger.error(f"× Failed to send video to {tv_ip}: {response.status_code}")
                return False
```

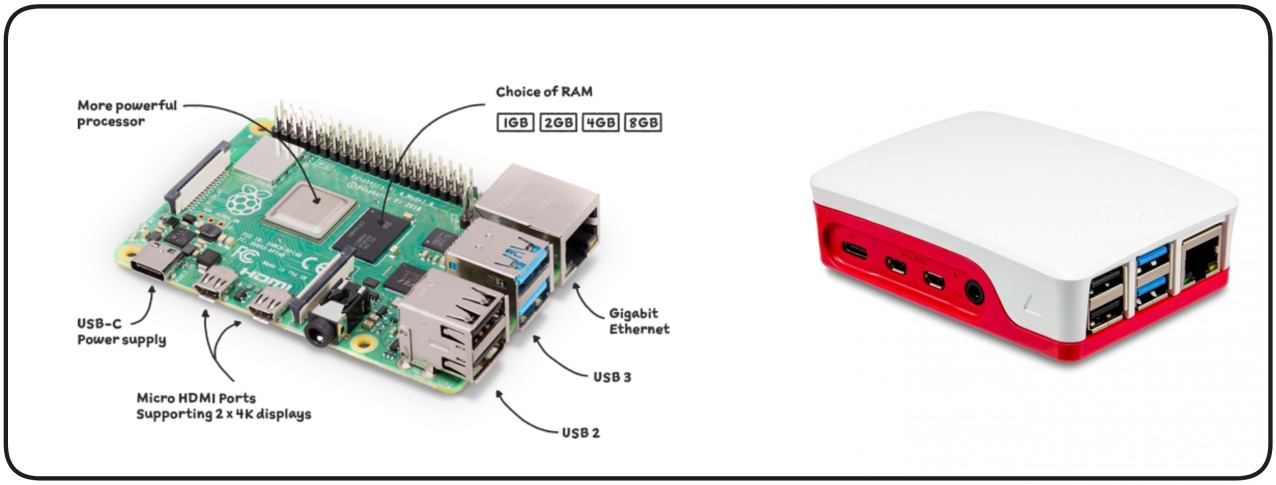
Python's watchdog library :
Watchdog is a Python library and set of shell utilities designed to monitor file system events. It provides a cross-platform API to detect changes in files and directories, such as creation, modification, deletion, and movement.

we can employ this library for detecting the changes in the folder for example whenever a new videos get uploaded to folder in the central hub it will detect these changes and run the functions in the script responsible for sending the video forward to TV.

2. second task to handle is connecting central hub with TV(challenging).
Currently their are few ways.

1. getting remote access of TV using static IP and credentials.
And then we can run a executable file on the TV remotely which will install media player and tracking mechanisms.

2. using raspberi pi.
We can connect raspberry pi to the TV and We can run a python script on the module which starts the port which will recieve the videos from central hub and play the videos, and send the tracking data. We can also add storage device to module if TV does not have enough storage for holding video. Main idea is that we interact with module and module executes actions in the TV.



3. third task is to run script on TV(challenging)
third task is to run executable or python script on tv for playing video and tracking the status of video and storage.

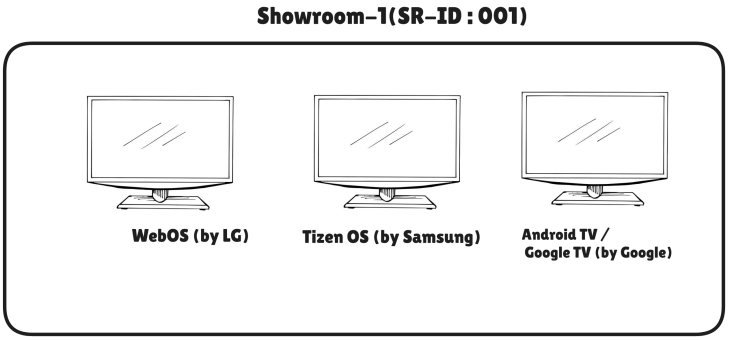
This task specifically challanging because i am not sure how executable or python script would run on TV.

My question : can i run my custom python based player on those TV's, or can i run a python script on those TV's

answer :
No, generally can’t run a Python script directly on consumer Smart TV OSes (webOS, Tizen, Roku OS, VIDAA, Panasonic “My Home Screen”). These platforms don’t ship a Python runtime or let us install arbitrary interpreters.

If TV can run a program then other challange come variation of OS used TV brands.

For example:



below is an table which shows OS used by different brands and on what they are based.

Brand	OS	Based on	App Store
LG	**webOS**	Linux	LG Content Store
Samsung	**Tizen**	Linux	Smart Hub
Sony / TCL / Xiaomi	**Android TV / Google TV**	Android	Play Store
Amazon / Onida	**Fire TV OS**	Android fork	Amazon Appstore
TCL / Hisense	**Roku OS**	Linux	Roku Store
Hisense / Toshiba	**VIDAA OS**	Linux	VIDAA Store
Panasonic	**My Home Screen**	Firefox OS	Panasonic Market

most of TV OS are based on Linux Kernal except few which are based on android or firefox or else.

They each have their own app model:

- **LG webOS:** HTML5/JavaScript apps (Enact), webOS JS services.
- **Samsung Tizen:** Web apps (JS) or .NET; media via AVPlay APIs.
- **Roku OS:** BrightScript / SceneGraph only.
- **VIDAA / Panasonic:** HTML5/JS-style apps with restricted APIs.
- **Amazon Fire TV OS & Android TV / Google TV:** Android apps (Java/Kotlin; C++ via NDK). You can *sometimes* sideload things like Termux/Python on Android TV, but it’s hacky, fragile, and not suitable for production.

Solutions :
1. either we make applications using their specified app model and put that on respective stores.

2. External player box (full Python control)

- Plug a **Raspberry Pi / Intel NUC / Android box** into HDMI on each TV.
- Run **Python player** (e.g., `python-vlc`, `mpv` via `subprocess`, or PySide/Qt GUI).
- our agent **watches a folder or polls a server**, downloads videos to disk, auto-plays, and survives reboots.

4. making dashboard(easier)

below are a rough design how a dashboard might look like(obviously it will change as we progress in project building stage)