

## OBJECTIVE

Build a flexible, intelligent advert injection engine for OBS that:

- Responds to real-time gameplay cues (score overlays)
  - Rotates ads dynamically based on timing goals
  - Prioritizes certain sponsors via weighted rotation
  - Tracks and logs every ad playback for post-event reporting
  - Offers full control through the web admin panel (no code edits)
- 

## FUNCTIONAL OVERVIEW

### A. GAMEPLAY MODES & TRIGGERS

#### 1. TEAM MODE

Trigger	Condition	Action
Halfway Ad	After 5 frames complete on <b>both overlays</b>	Play ~30s of ads
Lane Change	After 10 frames complete on both → <b>wait 30 sec</b>	Play ~180s of ads

#### 2. CUP MODE

Trigger	Condition	Action
Halfway Ad	After 5 frames complete on both overlays	Play ~30s of ads
Game Change	“Game X of Y” changes → Game 1 → 2, etc.	Play ~30s of ads
Final Lane Change	“Game Y of Y” + 10 frames complete → <b>wait 15 sec</b>	Play ~180s of ads

Frame count and game state are parsed from the hidden but accessible content inside scoring overlays (see `overlay_embed.html` and `overlays.py`).

---

### B. AD ROTATION LOGIC

#### New: Weighted Ad Rotation

Each ad will have a **priority weight** (e.g., 1–10 scale).

Priority	Effect
1 (low)	Rare rotation

Priority	Effect
10 (high)	Shown more frequently

Equal weights Uniform rotation

Internally:

- Ad queues per stream are sorted using a **weighted reservoir shuffle**
- Higher-priority ads are more likely to appear early in the queue
- Back-to-back repeats are always avoided

### Fuzzy-Fit Duration

Ads will be selected to:

- Meet or exceed a **target time** (30s, 180s, etc.)
- **Never cut off mid-play**
- Prefer combos that come **closest to target** (greedy or combination-based)

## C. PLAYBACK LOGGING

Each ad shown will be logged with:

Field	Description
-------	-------------

Timestamp	UTC time of playback
-----------	----------------------

Stream Name	e.g. "Pair 1&2"
-------------	-----------------

Ad ID	UUID of the ad
-------	----------------

Ad Name	Human-readable title
---------	----------------------

Duration	How long it played
----------	--------------------

Trigger Type	e.g. halfway, lane_change, game_change
--------------	--

Log file: /home/cornerpins/portal/logs/ad\_playback\_log.jsonl (JSON Lines)

## D. WEB INTERFACE CHANGES — advertising.html

### 1. Mode & Timing Configuration Panel

Dropdown for:

- TEAM MODE

- CUP MODE

Input fields (per mode):

- Halfway Ad Duration
- Game Change Ad Duration (CUP only)
- Lane Change Delay
- Lane Change Ad Duration

Values are saved to /home/cornerpins/portal/ads\_config.json.

---

## 2. Ad Priority Setting (on Upload Form)

When uploading an ad, add:

- **New Input:** “Priority Weight” (1–10)
    - Default: 5
    - Tooltip: “Higher value = more frequent playback”
- 

## 3. Playback History Report

New section in the Advertising Panel:

- Table of:
    - Time
    - Stream
    - Ad Name
    - Duration
    - Trigger
  - **Download CSV** button
    - /download\_ad\_log route
    - CSV contains all playback events
- 

## E. BACKEND STRUCTURE & FILES

### 1. New Config File — ads\_config.json

json

CopyEdit

```
{
```

```
"mode": "TEAM",

"team": {

  "halfway_duration": 30,

  "lane_change_delay": 30,

  "lane_change_duration": 180

},

"cup": {

  "halfway_duration": 30,

  "game_change_duration": 30,

  "final_game_delay": 15,

  "final_game_duration": 180

}

}
```

## 2. Playback Log File

- JSON Lines format: logs/ad\_playback\_log.jsonl
- Sample entry:

json

CopyEdit

```
{

  "timestamp": "2025-07-05T09:34:22Z",

  "stream": "Pair 3&4",

  "ad_id": "abc123",

  "ad_name": "KFC Logo Bumper",

  "duration": 10,

  "trigger": "halfway"

}
```

---

## 3. Modified Files

File	Changes
advertising.html	Add mode dropdown, duration fields, priority input, playback table

File	Changes
app.py	Handle config save/load, serve CSV download
watchdog_pair.py	Implement new trigger logic, ad rotation, fuzzy-fit, playback logging
ads_metadata.json	Now includes "priority": 1–10 per ad
ads_config.json	Created for ad logic settings
ad_playback_log.jsonl	Created and appended to at runtime

---

## OBS SCENE INTERACTION

Ads **do not** need to be added to OBS scenes in advance.










They will be:

- **Dynamically injected** into active scenes via WebSocket when triggered
- Then **removed** after playback
- **Scene layout stays clean** (maintained by setup\_12\_streams.py)

No changes to setup\_12\_streams.py are required.

---

## DELIVERABLES CHECKLIST

Feature	Status
Mode detection logic (TEAM/CUP)	
Trigger timers and durations	
Weighted ad rotation	
Ad fuzzy-fit selector	
Playback logger	
CSV download endpoint	
Priority setting UI	
Config save/load backend	
OBS injection preserved	
Per-stream queue isolation	