Creator Payout Web App Plan

1) What the app does

The app calculates **how much to pay each creator** for a selected period using post performance data from Shortimize.

It will:

1. Pull TikTok + Instagram video data from Shortimize API
2. Match the same video across both platforms (same creator, same date, same length)
3. Use the **higher view count** between the two platforms
4. Apply the payout tier rules
5. Return a google sheet/ excel:
    - total pay per creator (tab 1)
    - a detailed audit breakdown per video (tab 2)
    - a list of unmatched/ambiguous rows for review (tab 3)

---

2) Inputs

A) Creator mapping file (internal file) google sheet the tab needed is published to the web

This file is maintained internally and is the source of truth for creator identity.

Each row contains:

- `Creator_name (column b)`

- `Tiktok_handle (column P)`
- `Instagram_handle (column Q)`

This file is used to link a creator's TikTok and Instagram accounts together.

B) Shortimize API data (live)

The app pulls video data directly from Shortimize for a selected date range.

Required fields from Shortimize (per video)

Only use these fields:

- `username`
- `platform`
- `ad_link`
- `uploaded_at`
- `created_at`
- `video_length`
- `latest_views`
- `latest_updated_at`

Extra data needed for (reliability/debugging)

- `linked_account_id`
- `ad_id`
- `title`

- `private`
- `removed`

Do not include unrelated metrics or metadata in the payout logic.

---

3) User flow in the web app

1. User selects a payout period:
    - `start_date`
    - `end_date`
2. User runs payout calculation
3. App:
    - loads creator mapping file
    - pulls Shortimize videos for the selected period
    - processes matching and payout logic
    - returns results

---

4) Data ingestion and preparation logic

Step 1 — Pull videos from Shortimize API

Call the Shortimize videos endpoint with:

- selected date range (`uploaded_at_start`, `uploaded_at_end`)
- sorting by `created_at`
- metrics included (views must be present)

The purpose is to retrieve **all videos posted during the selected period**.

Step 2 — Keep only the needed fields

From each API row, keep only the required/recommended fields listed above.

Everything else is ignored.

Step 3 — Standardize the pulled video rows

For each video row:

- normalize `platform` labels (must be exactly `tiktok` or `instagram`)
- ensure `latest_views` is numeric
- ensure `video_length` is numeric
- parse:
    - `uploaded_at` as date
    - `created_at` as datetime
    - `latest_updated_at` as datetime

Step 5 — Map each video row to a creator

Use the mapping file to assign each video to a creator:

- If `platform = tiktok`, match `username` to `tiktok_handle`
- If `platform = instagram`, match `username` to `instagram_handle`

Add `creator_name` from the mapping file to each row.

If no match is found:

Put the row into an **Exceptions list** in tab 3 (unmapped handle) and exclude it from payout until reviewed. And add a comment “not in creator list” in the comment column.

Step 6 — Remove duplicate video rows

Deduplicate using:

- `ad_link` (primary)
- if needed, fallback to `ad_id`

If duplicates exist, keep the row with the most recent `latest_updated_at`.

This avoids double-paying the same post.

---

5) Matching logic (TikTok ↔ Instagram same video)

Goal

For each creator, identify which TikTok and Instagram videos are the same piece of content so the app can pay based on the **higher-performing platform**.

Step 7 — Group by creator only (not by date)

For each `creator_name`, split videos into:

- TikTok list
- Instagram list

Step 8 — Sort each platform list by `created_at`

Within each creator:

- sort TikTok videos by `created_at` ascending
- sort Instagram videos by `created_at` ascending

This creates a stable posting sequence across the entire pay period.

Why this works:

- `created_at` is the most precise timestamp you have
- your creators usually post in the same order on both platforms
- this avoids timezone date-boundary issues from `uploaded_at`

Step 9 — Primary match rule: sequence match + length confirmation

Try to pair:

- TikTok #1 with Instagram #1
- TikTok #2 with Instagram #2

- etc.

Then confirm the pair using `video_length`:

- exact match first
- allow ±1 second tolerance if needed

If lengths match → accept the pair.

Step 10 — Fallback if the sequence pair fails length check

If the lengths do **not** match for a candidate pair:

Fallback search (within same creator)

Look for a better match on the other platform using:

1. **same `video_length`** (or ±1 sec)
2. **closest `created_at (+- 24 hours)`**
3. optionally check `uploaded_at` date as a secondary hint (not primary)

This is exactly what you described:

- use sequence as default
- use length as validator
- if mismatch, search nearby for the correct same-length post

This keeps the matching resilient when:

- one platform post is delayed
- extra post exists on one platform

- timezone shifts affect `uploaded_at`

Step 11 — Mark confidence levels

This is useful and easy to explain:

- **High confidence**
    - matched by sequence
    - length confirmed
- **Medium confidence**
    - sequence mismatch, but fallback found same-length nearby
- **Low confidence / exception**
    - no good same-length match found
    - leave as unpaired and flag for review

Step 10 — Handle unmatched videos

If a video exists on only one platform (or no valid match is found):

- treat it as a valid single-platform payout row
- mark it as `unpaired (as a column) and add to tab 3`

This ensures creators are still paid for videos that weren't cross-posted.

---

6) Views selection logic

For each matched pair:

- `chosen_views = max(tiktok latest_views, instagram latest_views)`

For each unpaired row:

- `chosen_views = latest_views` from the available platform

Also store which platform had the higher views (for audit transparency).

---

7) Payout logic (exact business rules)

Step 11 — Qualification threshold

If `chosen_views < 1,000`:

- payout = $0
- mark as not qualified

Step 12 — Apply the 10M cap

If `chosen_views > 10,000,000`:

- use `10,000,000` for payout calculations
- keep original views for audit
- mark as capped

Step 13 — Apply payout tiers

Use the capped value (`effective_views`) to determine payout.

Tier logic

- 1,000–9,999 → $35
- 10,000–49,999 → $50
- 50,000–99,999 → $100
- 100,000–249,999 → $150
- 250,000–499,999 → $300
- 500,000–999,999 → $500
- 1,000,000–1,999,999 → $700
- 2,000,000–2,999,999 → $900
- 3,000,000–3,999,999 → $1,100
- 4,000,000–4,999,999 → $1,300
- 5,000,000–5,999,999 → $1,500

6M–10M logic (floor only)

For 6M and above:

- calculate whole millions using floor only:

6M–10M → **$1,500 + $150 × (floor_millions − 5)**

- e.g., 6.7M → floor_millions=6 → $1,500 +150 = $1650
- 9.2M → floor_millions=9 → $1,500 + $150×4 = $2100
- 12M → capped to 10M → floor_millions=10 → $1,500 + $150×5 = $2,250

---

8) Output logic, name the file “Polymarket Payout Summary yyyy-mm-dd to yyyy-mm-dd”

Output A, tab 1 — Creator payout summary (main result)

One row per creator with:

- `creator_name`
- `Qualified_video_count (should be videos from both platforms)`
- `Total_payout (sum of all the payout for that creator in this period)`
- `Paired_video_count (should be videos from both platforms, so 1 pair is count as 2)`
- `Unpaired_video_count (should be videos from both platforms)`
- `exception_count`

This is the main payroll output.

Output B — Video-level audit breakdown

One row per payout unit (matched pair or unpaired video):

Include:

- `creator_name`
- `uploaded_at`

- `video_length`
- TikTok:
    - `tiktok_link`
    - `tiktok_views`
- Instagram:
    - `instagram_link`
    - `instagram_views`
- `chosen_views`
- `effective_views` (after cap)
- `payout_amount`
- `paired/unpaired`
- `match_notes` (e.g., matched by length, closest pair)
- `latest_updated_at` (for freshness reference)

This makes payouts explainable and reviewable.

Output C — Exceptions / manual review list

Rows that could not be confidently processed, such as:

- handle not found in mapping file
- missing required fields (`video_length`, `uploaded_at`, `latest_views`, etc.)
- duplicate rows with conflicting values

- ambiguous matches (same day + same length + inconsistent counts)

This allows the app to stay automated while isolating edge cases for review.

---

9) Key assumptions (must be documented in the app/spec)

1. Creator identity is determined by the internal handle mapping file.
2. Cross-platform matching is based on:
    - same creator
    - same `uploaded_at` (date only!)
    - same `video_length` (with optional ±1 sec tolerance)
    - `created_at` used only as tie-breaker for ordering
3. Payout uses the **higher view count** across TikTok and Instagram for the same video.
4. Videos under 1,000 views do not qualify.
5. Payout calculation caps views at 10,000,000.
6. For million-based tiers, views are **floored**, never rounded up.

API docs for get all videos:

Shortimize API

Video Management

Endpoint: `https://api.shortimize.com`

API information

Shortimize API provides endpoints for managing and tracking social media accounts and videos.

Rate Limits

All API endpoints are rate limited per IP address. Rate limits vary by endpoint:

- **Standard endpoints**: 30 requests per minute
- **Tracking & History endpoints**: 120 requests per minute

When rate limited, you'll receive a `429 Too Many Requests` response. Check the `RateLimit-*` headers for limit status.

Manage and track video content

Get All Videos

`GET https://api.shortimize.com/videos`

Premium Operation - upgrade to Premium API in the app.

Retrieves a list of all videos tracked by your organization in Shortimize. This endpoint provides detailed information about each video, including engagement metrics, content details, and historical performance data.

Use this to get a comprehensive view of your video advertisement portfolio and their performance across different platforms.

Rate Limit: 30 requests per minute

Query Parameters

`page` integer · min: 1

Page number for pagination

Default: `1`

`limit` integer · min: 1 · max: 20000

Number of items to return per page

Default: `20000`

GET /videos — Test

```python
import requests

url = "https://api.shortimize.com/videos"

headers = {"Authorization": "<string>"}

response = requests.get(url, headers=headers)

print(response.text)
```

Show example in Python

Example Responses

200 401 429

```json
{
  "data": [
    {
      "organisation_id": "00000000-0000-0000-0000-000000000000",
      "ad_id": "00000000-0000-0000-0000-000000000000",
      "username": "username",
      "platform": "tiktok",
      "ad_link": "https://www.example.com/path/to/resource",
      "created_at": "2024-08-25T15:00:00Z",
```

`order_by` string · enum

Field to order results by

Enum values: `created_at` `uploaded_at` `latest_views` `latest_likes` `latest_comments` `latest_bookmarks` `latest_shares` `latest_engagement` show 1 more

Default: `latest_updated_at`

`order_direction` string · enum

Direction of ordering (ascending or descending)

Enum values: `asc` `desc`

Default: `desc`

`username` string

Filter videos by specific username

`linked_account_id` string · uuid

Filter videos by specific linked account ID

`uploaded_at_start` string · date

Filter videos uploaded on or after this date (inclusive)

`uploaded_at_end` string · date

Filter videos uploaded on or before this date (inclusive)

`latest_updated_at_start` string · date-time

Filter videos which have been last updated on or after this date and time (inclusive)

`latest_updated_at_end` string · date-time

Filter videos which have been last updated on or before this date and time (inclusive)

`ad_info_shop` boolean

Filter TikTok shop ads (true/false)

`ad_product_id_shop` boolean

Filter TikTok shop ads that have a product id. (true)

`has_metrics` boolean

Only get videos which have been retrieved atleast once (metrics are not null).

`collections` string

Optional comma-separated list of collections to filter accounts (e.g. collection1,collection2,collection3)

Make sure to URL encode them before sending.

Headers

`Authorization` string · required

Bearer token for authentication

Responses

200 401 429

Successful response with a list of videos and their detailed information

`data` object[]

– Hide properties

`organisation_id` string · uuid

Unique identifier for your organization

`ad_id` string · uuid

Unique identifier for the video

`username` string

Username of the account that posted the video

`platform` string · enum

Social media platform where the video is posted

Enum values: `tiktok` `instagram` `youtube`

`ad_link` string · uri

URL of the video

`created_at` string · date-time

Timestamp when the video was added to Shortimize

`removed` boolean

Indicates if the video has been removed from tracking

`linked_account_id` string · uuid

Unique identifier of the linked account

`uploaded_at` string | null · date

Date when the video was uploaded to the platform

`song_name` string | null

Name of the song used in the video

`song_link` string | null

Link to the song used in the video

`video_length` string | null

Length of the video in seconds

`title` string

Title or caption of the video

`not_safe` boolean

Indicates if the video is flagged as not safe for work

`private` boolean

Indicates if the video is set to private

`hidden_stats` boolean

Indicates if the video's statistics are hidden

`latest_views` integer

Most recent count of views

`latest_likes` integer

Most recent count of likes

`latest_comments` integer

Most recent count of comments

`latest_bookmarks` integer

Most recent count of bookmarks

`latest_shares` integer

Most recent count of shares

`latest_engagement` integer | null

Most recent count of total engagement

`latest_updated_at` string · date-time

Timestamp of the most recent update to the video's statistics

`outlier_multiplier` number

Multiplier indicating how much this video outperforms the account's median

`increase_1d` number

Percentage increase in views over the last day

`increase_7d` number

Percentage increase in views over the last 7 days

`increase_14d` number

Percentage increase in views over the last 14 days

`increase_30d` number

Percentage increase in views over the last 30 days

`ad_info_shop` boolean

Whether the ad is a tiktok shop ad or not. Only valid for Tiktok videos.

`ad_is_ad` boolean

Whether the ad is an actual ad. Only valid for Tiktok videos.

`ad_product_id_shop` string

The product shop id. Only valid for Tiktok videos.

`label_ids` array | null

Array of collection/label IDs associated with the video

`label_names` array | null

Array of collection/label names associated with the video

`pagination` object

– Hide properties

`total` integer

Total number of records

`page` integer

Current page number

`limit` integer

Number of items per page

`total_pages` integer

Total number of pages

`order_by` string

Field used for ordering

`order_direction` string

Direction of ordering

`filters` object

— Hide properties

`username` string

`linked_account_id` string

`uploaded_at_range` object

— Hide properties

`start` string · date

`end` string · date

`ad_info_shop` boolean

`has_metrics` boolean

`collections` string[]

Extras:
This is a sample data returned from shortimize API:


```
{
  "data": [
    {
      "organisation_id": "4d5b2d7b-660f-41e4-8506-021b7e7aa110",
      "organisation_created_at": "2026-02-22T00:31:06.331573",
      "ad_id": "2cd88fa1-46f9-447f-9901-ef9520779347",
      "username": "flow_bruce",
      "platform": "tiktok",
      "ad_link": "https://www.tiktok.com/@flow_bruce/video/7609364728032365845",
      "created_at": "2026-02-22T00:31:05.461255+00:00",
      "removed": false,
      "linked_account_id": "f4367436-a135-45c3-a337-9883ffbb9d05",
      "uploaded_at": "2026-02-21",
      "song_name": "original sound - flow_bruce",
      "song_link":
"https://sf16-ies-music-sg.tiktokcdn.com/obj/tiktok-obj/7609364769379797776.mp3",
      "video_length": 13,
      "ad_platform_id": "7609364728032365845",
      "title": "Are aliens real?? #skit #alien #fyp ",
      "not_safe": false,
      "private": false,
      "hidden_stats": false,
      "ad_info_shop": false,
      "ad_product_id_shop": null,
      "ad_is_ad": false,
      "latest_views": 367,
      "latest_likes": 14,
      "latest_comments": 1,
      "latest_bookmarks": 0,
      "latest_shares": 1,
      "latest_engagement": 16,
      "latest_updated_at": "2026-02-22T00:31:06.859191+00:00",
      "outlier_multiplier": 0,
      "increase_1d": 0,
      "increase_7d": 0,
      "increase_14d": 0,
      "increase_30d": 0,
      "label_ids": [
        "79689376-d475-4e7e-8c96-7993b18d6e55",
        "8dd8b959-6f08-4a84-89d6-6610ac5a7c40"
      ]}
```