Hafnia Hackathon 2025: Reimagine the Future with Vision Language Models

Instructions

1. Overview 2
2. Important dates & prizes 2
3. Eligibility & rules (short) 2
4. Access: registration, Slack, API token 2
5. The VLM service (Cosmos-Reason1-7B) — capabilities & limits 2
6. What you can send 3
7. Prompting guidelines & templates 4
8. API overview & auth (+ OpenAPI) 5
9. Integration patterns examples 6
10. Code snippets 7
11. Submission package & judging criteria 9
12. Responsible AI, privacy & data handling 9
13. Support & resources 9
14. Troubleshooting & FAQs 9

1. Overview

Build an integration that uses our cloud VLM-as-a-Service (powered by NVIDIA Cosmos-Reason1-7B) to deliver real value for end customers—e.g., validate real-time analytics alerts with short video clips, or generate automated situation reports for city areas. Shortlisted teams demo live at the Milestone Developer Summit in Copenhagen.

2. Important dates & prizes

- Hackathon window: Oct 13 – Nov 11, 2025
- Webinars/Q&A: Oct 23, 2025
- Submissions open: Oct 24, 2025
- Final submission deadline: Nov 3, 2025 at 12:00 (noon) CET
- Presentations & winners: Nov 11, 2025 (Developer Summit)
- Prizes: €5,000 (1st), €3,000 (2nd), €2,000 (3rd) + Jetson Thor for the top two + promo opportunities.

3. Eligibility & rules (short)

- Teams or individuals.
- Build original projects during the hackathon window.
- Participants retain IP; organizer may showcase results with credit (see Terms).

4. Access: registration, Slack, API token

Register on the hackathon page; accepted teams (limited seats) receive:

- API docs & key (sent by the Hafnia team via email).
- Invite to the Hafnia × NVIDIA Slack support channel.

5. The VLM service (Cosmos-Reason1-7B) — capabilities & limits

- **Model:** NVIDIA **Cosmos-Reason1-7B**, optimized for physical common sense and spatio-temporal reasoning in video.
- **Evaluation defaults:** The model will use **2 FPS** sampling; long contexts are supported at the model level; please make sure the videos are shorter than **30 s**.
- **Output:** The model will output maximum of **4096 tokens**.

6. What you can send

Video to the service:

- Containers: **.mp4** and **.mkv**
- Max file size: **100 MB**
- Duration: **<30 s** mandatory (≈ 15 s optimal)
- Resolution: No hard cap; effective processing up to **~1920 px** width
- Audio: **Ignored**
- Server-side processing: Service auto-transcodes/downsamples if needed.
- Storage & deletion: Assets/results are temporarily stored for processing and purged after the hackathon. You can delete assets yourself at any time (see DELETE endpoint below). (All organizer-provided)
- Prompts (content & size)
- Encoding: UTF-8 strings.
- System prompts: **Supported** (recommended for output format & safety).
- Recommended prompt size (hackathon): keep **≤ ~500 tokens** total input for reliability (most use-cases need far less). (Guidance based on Reason1's long-context support)
- Structured output: You can request strict JSON; outputs are not guaranteed perfectly deterministic—validate before parsing.

7. Prompting guidelines & templates

General tips

- Be specific about the time span (e.g., “Between 00:12–00:27...”). Please note that the time spans will work only if provided in the system prompt.
- Provide context (camera semantics, thresholds/policies).
- Ask for structured output (JSON).
- Encourage a reasoning section (e.g., <think>...</think>) and a separate <answer>/JSON.

Templates

Alert validation (binary + why)

```
[SYSTEM] Follow exactly this JSON schema; if uncertain use "uncertain".
<format>
{"verdict": "true|false|uncertain", "reason": "...",
 "time_spans": [{"start_s": float, "end_s": float}]}
</format>

[USER] Validate the {ALERT_TYPE} alert in this clip between {T0}-{T1} seconds.
Explain briefly. Cite frame/time spans.
```

Status report

```
[SYSTEM] Produce concise bullet points with time-codes, then a final JSON.
JSON keys: traffic_flow, incidents, obstructions, vulnerable_users, notable_behaviors.

[USER] Create a situation report for {AREA} using the video.
```

Comparative check

```
[USER] Compare {ZONE_A} vs {ZONE_B}. Who waits longer at crossing?
Return {"zone_with_longer_wait": "...","evidence":[{"start_s":...,"end_s":...}]}.
```

8. API overview & auth (+ OpenAPI)

- **Base URL:** *https://api.mdi.milestonesys.com* (single environment; no separate sandbox).
- **Auth header format:** Authorization: ApiKey <KEY>:<SECRET>
- **OpenAPI (participants-only):** https://api.mdi.milestonesys.com/api/v1/vlm-docs

Core flow (current)

1. **Upload a video asset:**
   POST /api/v1/assets (multipart form with file=@yourclip.mp4|.mkv) → returns asset_id.

2. **Request a completion (chat)**
   POST /api/v1/chat/completions with an OpenAI-style messages array; a user message can include text and an asset reference:

```
{
 "messages": [
  {
   "role": "system",
   "content": [
    {
     "type": "text",
     "text": "Return strict JSON with time-codes."
    }
   ]
  },
  {
   "role": "user",
   "content": [
    {
     "type": "text",
     "text": "Write a summary of the provided video."
    },
    {
     "type": "asset_id",
     "asset_id": "<YOUR_ASSET_ID>"
    }
   ]
  }
 ]
}
```

3. **Synchronous responses; no webhooks**
   Wait for the response before sending the next request. There are no fixed per-team quotas right now, but dynamic throttling may apply under heavy load (HTTP 429/5xx → backoff/retry).

4. **Delete assets when done**
   DELETE /api/v1/assets/{asset_id}. All assets will be purged after the hackathon, but you can delete earlier.

Optional downstream integration: If you're tying into XProtect, use the Events REST API to trigger/consume events and the Events & State WebSocket API for realtime subscriptions.

9. Integration patterns examples

- **Real-time alert validation (VMS + Analytics + VLM)**
  - Analytics raises an alert → export a 10–20 s clip → Upload + completion with a validation prompt → use the verdict to escalate/close via your system’s alarms/events API (e.g., XProtect Events).
- **Automated city status reports**
  - Sample short clips on a schedule → batch Upload + completion → store JSON → dashboards.
- **Assisted incident triage**
  - Operator bookmarks a segment → one-click “Explain” → VLM returns a 5-bullet summary + time-codes.

10. Code snippets

curl

```
# 0) Auth
export API_PLUGIN='ApiKey <KEY>:<SECRET>'

# 1) Upload an asset (.mp4 or .mkv)
curl -H "Authorization: $API_PLUGIN" \
    -F file=@sample.mp4 \
    https://api.mdi.milestonesys.com/api/v1/assets
# -> { "asset_id": "41aefe6a-45b9-46c9-aef8-a3c21f451a9c", ... }

# 2) Run a completion (chat)
cat > req.json << 'JSON'
{
    "messages": [
        {"role":"system","content":[{"type":"text","text":"Return a
JSON object and include time-codes."}]},
        {"role":"user","content":[
            {"type":"text","text":"Validate whether the bus blocks the
bike lane between 12s and 27s. Explain briefly."},
            {"type":"asset_id","asset_id":"41aefe6a-45b9-46c9-aef8-
a3c21f451a9c"}
        ]}
    ]
}
JSON

curl -H "Authorization: $API_PLUGIN" \
    -H "Content-Type: application/json" \
    -d @req.json \
    https://api.mdi.milestonesys.com/api/v1/chat/completions

# 3) Delete the asset (optional)
curl -X DELETE -H "Authorization: $API_PLUGIN" \
    https://api.mdi.milestonesys.com/api/v1/assets/41aefe6a-45b9-46c9-
aef8-a3c21f451a9c
```

Python

```python
import os, json, requests

BASE = "https://api.mdi.milestonesys.com/api/v1"
AUTH = {"Authorization": "ApiKey <KEY>:<SECRET>"}

# 1) Upload asset
with open("clip.mp4","rb") as f:
    up = requests.post(f"{BASE}/assets", headers=AUTH, files={"file":
f}, timeout=120)
asset_id = up.json()["asset_id"]

# 2) Chat completion
payload = {
    "messages": [
        {"role":"system","content":[{"type":"text","text":"Return
strict JSON with time-codes."}]},
        {"role":"user","content":[
            {"type":"text","text":"Create a short situation report."},
            {"type":"asset_id","asset_id":asset_id}
        ]}
    ]
}
resp = requests.post(f"{BASE}/chat/completions",
    headers={**AUTH,"Content-Type":"application/json"},
    data=json.dumps(payload), timeout=180)
print(resp.json())

# 3) (Optional) Delete asset
requests.delete(f"{BASE}/assets/{asset_id}", headers=AUTH, timeout=60)
```

11. Submission package & judging criteria

Submit:

- Application (working integration using the VLM API)
- Text description (features, workflow, how VLM is used)
- Demo video (≤ 3 minutes; public YouTube/Vimeo link)
- Optional: Source code repo, live demo link, extra docs.

Judging:

- Potential value/impact
- Creativity
- Technological execution & architecture
- Functionality & scalability
- Demo quality / wow factor.

12. Responsible AI, privacy & data handling

- **No data provided by the organizer**; participants must ensure rights & compliance for any footage used.
- **Audio ignored by the service**; content filters (e.g., faces/plates) are your responsibility.
- **Retention:** Temporary storage only for processing; all data removed after the hackathon. You can delete assets via the API.
- Processing region: **US servers**.

13. Support & resources

Cosmos-Reason1 Evaluation Guide (evaluation flow).

- Cosmos-Reason1 model card note on output tokens & timestamps.
- Cosmos-Reason1 GitHub (overview).
- XProtect Events REST & Events/State WebSocket.

14. Troubleshooting & FAQs

- “Response is truncated.” Try modifying your system prompt.
- “Temporal references are off.” Trim to the relevant window; reference explicit time ranges in your prompt.
- “Upload fails.” Keep files ≤100 MB; <30 s, ≤~1920 px width, .mp4/.mkv.
- “Async/polling?” Completions are synchronous; no webhooks. Send the next request after the prior one returns; back off on 429/5xx.
- “Strict JSON?” You can request it; not perfectly deterministic—validate before parsing.