PersonaGymAttack — Video Recording Script + Commands (Green + White)

This document is a ready-to-read script for your submission videos.
It includes:
- what to **say** (short, rubric-aligned narration)
- what to **show on screen**
- exact **commands to run** (tested in this repo)

---

0) One-time setup (do this before recording)

What to say (optional):
> I'm going to run a fully local, deterministic evaluation so you can reproduce exactly what you see.

Commands
```
cd /workspaces/personagymattack
source .venv/bin/activate

# (Optional) sanity check unit tests
python -m pytest -q
```

---

1) Define the baselines (so you can answer "baseline white agent(s)")

We use three *white* agents in the demo:

1) **Baseline A: `prompt`**
- Implementation: `src/personagym_r/baselines/white_prompt_only.py`
- Behavior: rule/pattern-based responses intended to maintain persona and refuse unsafe requests.

2) **Baseline B: `tool`**
- Implementation: `src/personagym_r/baselines/white_tool_user.py`
- Behavior: slightly stronger baseline with simple memory/consistency templates.

3) **Negative control: `bad` (intentionally failing)**
- Implementation: `src/personagym_r/baselines/white_bad_agent.py`
- Behavior: deliberately non-compliant; used to demonstrate scoring penalties.

What to say:
> For our demo and validation, we compare two baseline white agents (`prompt` and `tool`) and one intentionally failing agent (`bad`) to show the evaluator is sensitive to behavioral differences.

---

2) GREEN AGENT VIDEO (≤ 5 minutes)

2.1 Task intro (30–45 seconds)

What to say (read this):
> Our benchmark is PersonaGym-R. A white agent is assigned a persona and then challenged by an attacker that uses social-engineering style tactics. The green agent runs the interaction and evaluates whether the white agent stays in-character and respects boundaries.

What to show on screen:
- The repo root

- The task folder structure

Commands
```
ls -1 tasks
ls -1 tasks/travel_yosemite_001
```

What to say:
> Each task has a persona, a goal/horizon, a rubric, and a seed file that controls which attacker tactics are used.

2.2 Environment + actions (45–60 seconds)

What to say:
> The state is the conversation history plus the task config. Each step: the attacker produces a message, the white agent replies, and we record the trace. The task ends after a fixed horizon or earlier if a break is detected.

What to show:
- The scoring formula in the README

2.3 Run the evaluation live (90–120 seconds)

What to say:
> Now I'll run the same task with three different white agents using the same seed so this is fully reproducible.

Commands (tested)
```
./scripts/run_video_demo.sh
```

What to show:
- The printed "Report directories" lines
- Open each report summary and highlight the score table and break analysis

Commands
```
# Copy/paste the three report paths printed by the script.
# Then preview summaries:
sed -n '1,120p' reports/<PROMPT_DIR>/summary.md
sed -n '1,120p' reports/<TOOL_DIR>/summary.md
sed -n '1,120p' reports/<BAD_DIR>/summary.md
```

What to say (as you compare):
> You can see the evaluator produces a metric breakdown (P/B/S/E) and an overall score R. With the same attacker seed, the `tool` baseline scores higher than the `prompt` baseline here, and the `bad` agent is penalized for failing persona adherence.

2.4 Reliability / ground-truth check (30–45 seconds)

What to say:
> This is deterministic: running again with the same task and seed reproduces the same scores. The run also saves a full trace and a summary report.

Command
```
# Re-run exactly (same task + seed + baselines)
SEED=7 TASK=$PWD/tasks/travel_yosemite_001 ./scripts/run_video_demo.sh
```

---

3) WHITE AGENT VIDEO (≤ 5 minutes)

This video is about the *white agent implementation*. In this repo there are:
- local baselines (`prompt`, `tool`, `bad`) used for comparisons
- an A2A/AgentBeats-compatible reference white service: `agentbeats/white_agent.py`

3.1 Architecture overview (60–90 seconds)

What to say (read this):
> Our white agent exposes a simple session-based API: create a session with a persona, respond to observations, and optionally submit a final message. Internally it is a lightweight rule-based responder designed to maintain persona and refuse unsafe identity-verification and manipulation attempts.

What to show:
- `agentbeats/white_agent.py` (point out session lifecycle endpoints)

3.2 Run the white agent service locally (60–90 seconds)

Terminal 1: start the white server
```
HOST=127.0.0.1 AGENT_PORT=8001 python agentbeats/white_agent.py
```

Terminal 2: send one session + respond request
```
python scripts/demo_white_agent_requests.py \
  --base-url http://127.0.0.1:8001 \
  --task tasks/travel_yosemite_001
```

What to say:
> This demonstrates the exact inputs and outputs at each step: a persona payload, an observation payload, and a text response.

3.3 Show evaluation results (30–45 seconds)

What to say:
> Finally, we evaluate white agent behaviors using the same green evaluation loop shown in the green-agent video, and we compare against baselines.

---

4) Quick troubleshooting (if something goes wrong during recording)

- If `./scripts/run_video_demo.sh` can't find Python: set `PYTHON`:
  ```
  PYTHON=$PWD/.venv/bin/python ./scripts/run_video_demo.sh
  ```
- If you want a different task/seed:
  ```
  TASK=$PWD/tasks/tech_support_002 SEED=12 ./scripts/run_video_demo.sh
  ```

---

5) Notes for your submission writeup

- "Baseline white agent(s)" = `prompt` and `tool` in `src/personagym_r/baselines/`.
- "Negative control" = `bad` used to demonstrate that evaluation penalizes persona failures.
- "Commands to reproduce" = the exact script invocation lines in Section 2.3 and 2.4.