ANIMAtiZE App Summary (One Page)

What it is

ANIMAtiZE is a Python framework for turning static images into cinematic video-generation prompts using computer vision analysis and model-specific prompt compilation. It also includes provider routing, reliability controls, and continuity tooling for multi-shot workflows.

Who it's for

Primary persona: content creators, filmmakers, and creative technical teams producing AI video from still images.

What it does

- Analyzes scene structure (objects, depth, composition, aesthetics) with OpenCV and PIL in `src/analyzers/scene_analyzer.py`.
- Predicts justified character, camera, and environmental movement from a single frame in `src/analyzers/movement_predictor.py`.
- Compiles director intent into model-ready prompts with temporal controls, versioning, and deterministic seeds in `src/generators/video_prompt_generator.py`.
- Executes unified video requests with retries, fallback chains, cache, and metrics in `src/core/video_pipeline.py`.
- Supports multiple generation providers through adapters (`flux`, `veo`, `runway`, `sora`, `pika`) under `src/adapters/`.
- Maintains cross-shot continuity with character/style/world references and validators in `src/wedge_features/consistency_engine.py` and `src/wedge_features/consistency_integration.py`.
- Includes regression and benchmarking utilities in `src/evaluation/` plus scripts such as `scripts/validate_golden_reference.py`.

How it works (repo evidence)

- Input flow: static image (and optional director intent) enters analyzers and prompt compiler.
- Analysis layer: `SceneAnalyzer` and `MovementPredictor` extract composition, depth, object cues, and justified motion hypotheses.
- Prompt layer: `VideoPromptAnalyzer` and `VideoPromptCompiler` transform analysis into provider-aware prompt payloads and control parameters.
- Execution layer: `VideoGenerationPipeline` builds unified requests, invokes provider adapters, and wraps execution with retries/fallback/cache/metrics.
- Continuity layer: `ConsistencyOrchestrator` and `ConsistencyEngine` persist reference assets under `data/reference_library` and score cross-shot consistency.
- Output flow: unified responses return status, metadata, and provider result payloads (for example generated video URLs).
- Not found in repo: canonical runtime API/bootstrap implementation for `ANIMAtiZEFramework` referenced by README and `src/main.py` (import target `core.framework` is absent).
- Not found in repo: concrete SQLAlchemy model/schema usage despite SQLAlchemy listed in dependencies.

How to run (minimal)

- Create and activate a virtual environment: `python3 -m venv .venv && source .venv/bin/activate`.
- Install dependencies: `pip install -r requirements.txt -r requirements-cv.txt`.
- Run test suite baseline: `pytest tests/ -v`.
- Run a local analyzer with your own image: `python src/analyzers/movement_predictor.py /absolute/path/to/image.jpg`.
- Optional provider demo (after adding real API keys in example code): `python examples/video_pipeline_usage.py`.
- Not found in repo: a verified, end-to-end single startup command for the full app runtime.