Technical Report on AI Fairness Data Generation and Question Answering System

Aravind Narayanan[1], Ananya Raval[1], Sindhuja Chaduvula[1], Karanpal Sekhon[1], Amandeep Singh, Shweta Khushu[1], Shaina Raza[1,*]

[1] Vector Institute for Artificial Intelligence, Toronto, Canada

*Corresponding Author

shaina.raza@vectorinstitute.ai

October 27, 2025

Project: https://vectorinstitute.github.io/vector-aixpert/
Code: https://github.com/VectorInstitute/vector-aixpert

Contents

1 Overview

This document summarises the purpose and structure of an experimental framework for studying bias, fairness and safety in generative AI (GenAI) systems. The framework is designed to generate synthetic multimodal data: text, images and videos, under controlled conditions so that researchers can examine how models behave when demographic attributes or risk factors are varied. The codebase provides utilities to build baseline examples and matched "controlled" variations where instructions about demographic diversity and fairness are made explicit. The build instructions describe how to set up the project using the `uv` package manager and how to install core and development dependencies. Figure 1 depicts the main data-generation pipeline.

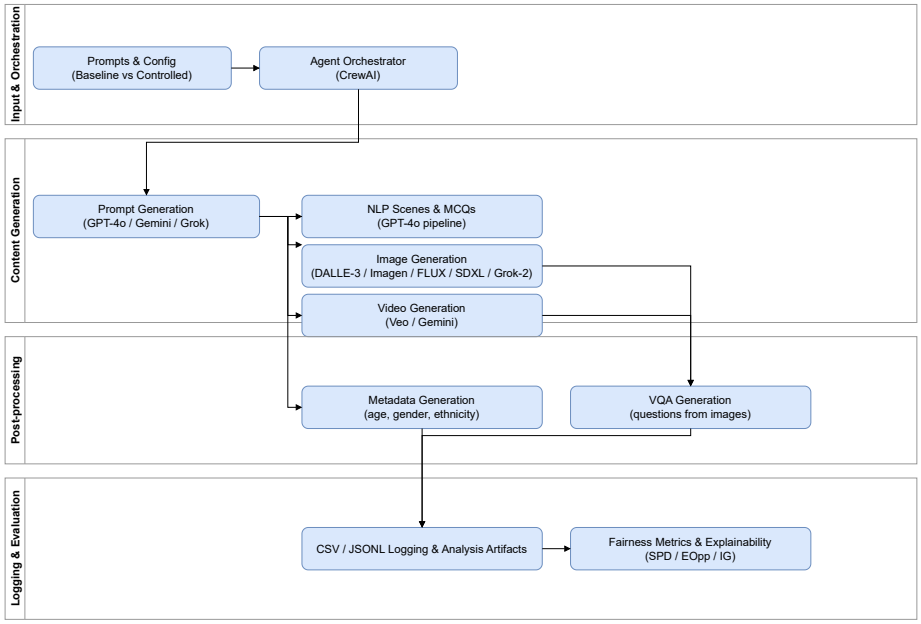

Figure 1: Layered data pipeline for multimodal content generation and evaluation from input & orchestration to fairness metrics. *Abbreviations:* SPD = Statistical Parity Difference; EOpp = Equal Opportunity; IG = Integrated Gradients.

2 Repository structure

The project root contains configuration files (e.g., `pyproject.toml`), a licence and contribution guidelines. Most of the code lives under `src/aixpert`. Table 1 summarises the major modules and their purpose.

Module / Folder	Description	Relevant Files
src/aixpert/controlled_images	Command-line tool for generating baseline and controlled images to study demographic bias. It wraps multiple image providers and logs metadata and checkpoints.	main.py, image_utils.py, prompts.py, utils.py, configs/img_gen_config.yaml
src/aixpert/data_generation /synthetic_data_generation/ images	Scripts for generating synthetic image prompts, images, and visual–question–answer (VQA) pairs. Designed to create domain- and risk-specific data sets.	main.py, system_utils.py, prompt_generation.py, dalle_3.py , gemini.py, vqa_generation.py, csr_vqa_generation.py, system_prompt .py, utils.py
src/aixpert/data_generation/ synthetic_data_generation/nlp	Pipeline for generating textual scenes and multiple-choice questions (MCQs) using OpenAI GPT-4o models. Uses domain- and risk-specific templates and schemas.	main.py, prompt_gen_utils.py, prompt_template.py, schema.py
src/aixpert/data_generation /synthetic_data_generation/ videos	Code for generating short videos via Google's Veo/Gemini models. Includes checkpoint logic and utilities for saving multiple videos per prompt.	veo.py, utils.py
src/aixpert/data_generation/ agent_pipeline	Orchestrates multi-step workflows for prompt creation, image synthesis, and metadata generation using CrewAI.	agent.py, crew_orchestrate.py, custom_llm.py, load_text_llm.py, flows/ image_generation_flow.py
src/aixpert/IE_Social_LLM	Contains scripts for computing group-fairness metrics and zero-shot classification with integrated gradients.	scripts/fairness_metrics.py, scripts/ llm_zero_shot_explain.py
docs	Documentation resources.	README.md, CONTRIBUTING.md, index.md, user_guide.md,

Table 1: Major modules and their purpose. Descriptions summarise the high-level functionality of each component.

3 Controlled image generation

The `controlled_images` module provides a unified command-line interface to generate matched pairs of images. It defines several categories: *CEO*, *nurse*, *software engineer*, *teacher* and *athlete*. For each category there is a short baseline description (e.g., "A CEO in an office") and a longer controlled description that instructs the model to ensure demographic diversity across gender and ethnicity and to avoid stereotypes. These descriptions are used to produce pairs of images: one generic and one fairness-aware. Key elements of the controlled image pipeline include:

- **Image generators.** Image generators wraps several providers. It implements helper functions to call FLUX.1-dev [4], Google Imagen [3] via the Gemini API, OpenAI's image API (DALL-E 3 and GPT-Image 1) [1], xAI's Grok-2 image API [**?**], and Stable Diffusion XL Turbo [**?**]. Each provider returns PNG bytes. A retry decorator adds exponential backoff and multiple attempts, and helpers support saving PNG files and decoding base64-encoded results.

- **Configuration.** A YAML file defines global defaults such as the number of samples per setting, retry behaviour and flush frequency. It lists provider-specific options (e.g., model names, output directories, environment variables) and allows enabling or disabling providers. API keys are pulled from the YAML or from environment variables using the `decouple` library. Annotations are stored in CSV files in the configured output directory.

- **Main script.** `main.py` implements a command-line interface (CLI). It reads the prompts and configuration, binds the appropriate generator for each provider and iterates over all categories,

baseline/controlled settings and sample indices. For each target it either skips generation if the file already exists or calls the provider to produce a PNG. It writes results to CSV and uses a small checkpoint file to resume aborted runs. Users can run `python main.py -provider gpt` to generate only DALL-E images or `-provider all` to generate across all enabled providers.

These components allow researchers to produce matched pairs of images for fairness auditing. The prompt explicitly instructs the model to depict a person in a realistic scene while ensuring diverse gender and ethnicity representation, enabling controlled comparisons between the generic and fairness-aware versions.

4 Synthetic data generation

The framework includes a suite of modular pipelines for generating synthetic multimodal data encompassing text, images, metadata, and VQA pairs. All components follow a consistent execution pattern: configuration parameters are loaded from YAML specifications, model credentials are securely retrieved from environment variables, and structured prompts guide the generation process. Checkpointing mechanisms ensure that progress is preserved across runs, enabling robust and resumable execution.

This design promotes scalability, transparency, and reproducibility in synthetic data creation. By standardizing orchestration across modalities, the framework supports systematic experimentation and controlled dataset generation for diverse Responsible AI evaluation scenarios.

4.1 Image and VQA generation

The image and visual question answering (VQA) generation module provides a unified framework for producing multimodal synthetic datasets that combine text-based prompts, generated images, and associated question–answer pairs. It supports domain- and risk-specific content creation across areas such as hiring, legal, and healthcare, with configurable parameters controlling risk categories including bias, toxicity, representation gaps, and security concerns.

The pipeline begins by generating detailed image prompts using large language models, which are then rendered into images through high-fidelity generative models such as OpenAI's DALL·E and Google's Gemini. These models enable fine-grained control over quality, style, and composition, facilitating diverse and demographically balanced visual outputs. The system also produces corresponding VQA pairs that probe ethical, social, and commonsense reasoning aspects of the generated imagery, enhancing the dataset's utility for Responsible AI benchmarking.

A single orchestrated interface integrates all stages: prompt creation, image synthesis, metadata generation, and VQA formulation within a checkpoint-safe, configuration-driven workflow. This design ensures reproducibility, scalability, and modular extensibility, allowing users to execute individual stages or full pipelines as needed. Together, these capabilities enable systematic multimodal data generation for evaluating fairness, robustness, and interpretability in AI systems.

4.2 NLP synthetic scenes and MCQs

The NLP module implements a comprehensive pipeline for generating synthetic textual scenes, multiple-choice questions (MCQs), and corresponding answers to support Responsible AI evaluation across diverse domain–risk contexts. The system leverages large language models, such as OpenAI's

GPT-4o family, to produce structured and contextually grounded content following configurable domain and risk specifications.

The framework integrates configurable prompt templates and structured output schemas to ensure consistency and validity across generated data. It supports a range of risk categories—including bias, toxicity, representation gaps, and security—by guiding model behavior through domain-specific system prompts. Each generation stage, from scene creation to question formulation and answer justification, can be executed independently or as part of an end-to-end workflow, allowing flexible experimentation and control.

Beyond scene and question generation, the pipeline emphasizes robustness and reproducibility through structured output validation, checkpointing, and automated recovery mechanisms. Generated MCQs are designed to probe the ethical, social, or safety dimensions of the scenes, with answer rationales that enhance interpretability and downstream evaluation. Overall, the NLP pipeline provides a scalable, transparent, and extensible framework for constructing high-quality textual benchmarks in Responsible AI research.

4.3 Video generation

The `videos` submodule extends synthetic data generation to video. Its `veo.py` script uses Google's Veo (Gemini) [1]model to generate short clips. Configuration parameters specify the model variant, aspect ratio, person generation setting and number of videos per prompt. The script loads prompts from JSONL files, uses the Gemini API to generate videos and saves them with deterministic filenames (e.g., `hiring_bias_video_1_1.mp4`). A checkpoint file records the last processed index and the names of already generated videos to support resumption. Utility functions implement exponential-backoff retries and jitter for robustness, manage checkpoints and provide helper routines for saving videos. Recent releases of Veo emphasise richer native audio, greater narrative control and enhanced image-to-video capabilities, with features such as reference image guidance, scene extension and first/last frame control [**?**].

5 Agentic Data Generation Framework

The agentic data generation framework constitutes a unified system for end-to-end synthetic data generation, integrating prompt engineering, image synthesis, metadata creation, and optional visual question answering (VQA) within a coordinated execution architecture. Built upon the *CrewAI* orchestration library [**?**], the framework adopts an agent-based paradigm in which each agent is defined by a specific role, goal, and model configuration. This design enables modular yet composable task execution, ensuring coherent coordination across diverse domains and risk categories while maintaining traceability and control.

CrewAI-driven orchestration allows multiple agents—each powered by a large language model (LLM) or multimodal generator—to collaborate in a structured workflow. The framework supports both cloud-hosted and locally deployed models, enabling seamless switching among model families such as OpenAI's GPT-4o [6], Google's Gemini [**?**], and xAI's Grok-2 [12] without code-level modification. Its configuration-driven structure facilitates scalable experimentation and straightforward extensibility to new domains or Responsible AI evaluation contexts.

[1] `Veo3VideoGenerationAPI`

Through checkpointing, structured outputs, and consistent configuration management, the system ensures reproducible and resumable generation workflows. Overall, the framework provides a robust, extensible, and transparent foundation for responsible synthetic data creation, unifying text, image, and metadata generation under a single agentic and orchestrated pipeline.

6 Fairness metrics and zero-shot explainability

The module contains scripts to evaluate fairness and interpretability [5] of language models on social tasks:

Group-fairness metrics. `fairness_metrics.py` computes statistical parity difference (SPD) and equal-opportunity (EOpp) metrics [8] difference across demographic identities for binary classification. Given predictions, labels and identity columns, it calculates per-group accuracy, F1, true positive rate, false positive rate and positive prediction rate. For each identity attribute the script computes SPD $= P(\hat{Y} = 1 \mid A = 1) - P(\hat{Y} = 1 \mid A = 0)$ and EOpp difference $= \text{TPR}(A = 1) - \text{TPR}(A = 0)$. It also records the worst-case absolute disparity and the worst group accuracy/F1 across identities. The output is saved into CSV files summarising per-group and per-identity results.

Zero-shot classification with integrated gradients. `llm_zero_shot_explain.py` performs zero-shot classification of social texts (e.g., toxicity, hate, offence) using a language model such as DistilGPT-2 [10]. It is designed to be model-agnostic, allowing the use of alternative large language models, including open-source variants like LLaMA [2] and Mistral [11], as well as closed-source models such as GPT-4/5 or Gemini.

For each text it constructs a prompt instructing the model to decide whether the text is toxic/non-toxic (or hateful/not hateful, etc.) and computes the difference in log-probability between the positive and negative label. The sign of this difference forms the prediction. To interpret the decision, the script implements integrated gradients: it interpolates between a zero embedding baseline and the actual prompt embedding, accumulates gradients of the log-probability difference with respect to the input embeddings and attributes the score to each token. The result is a heatmap over tokens illustrating which words contributed most to the classification. The script includes options to save intermediate heatmaps and supports CPU, CUDA and Apple MPS backends.

These metrics and interpretability tools enable researchers to quantify and explain how models behave across demographic groups, complementing the controlled data generation pipeline.

7 Tests and documentation

Automated testing and documentation workflows are integrated to ensure code quality, consistency, and maintainability across the project.

7.1 Testing

Unit tests are located in the `tests/` directory and include basic checks to verify that the `aixpert` package imports correctly and that the testing framework is properly configured. The test suite uses `pytest`, and a dedicated pre-commit hook runs all non-integration tests automatically before

commits to catch issues early. Future contributors are encouraged to extend the test coverage to individual modules and integration scenarios.

Code linting, formatting, and type checking are enforced through `pre-commit` hooks, including:

- **ruff** for linting and formatting Python and Jupyter files;
- **mypy** for static type checking (configured via `pyproject.toml`);
- **typos** for spell checking;
- **nbQA** for running `ruff` on Jupyter notebooks.

These checks are automatically run locally via `pre-commit` and in continuous integration through `pre-commit.ci`, ensuring consistent style and correctness across contributions.

7.2 Documentation

Project documentation is built using `MkDocs` with the *Material for MkDocs* theme.

- The `docs/` directory contains all documentation sources.
- The `user_guide.md` provides detailed instructions for running pipelines and configuring providers.
- The `api.md` file contains the API reference, automatically generated using `mkdocstrings` with the NumPy docstring style.
- Additional customization (e.g., themes, icons, and styles) is managed via `mkdocs.yml` and custom CSS under `docs/stylesheets/extra.css`.

The documentation site includes integrated search, syntax highlighting, emoji, and dynamic navigation features. The live documentation can be built locally using `mkdocs serve`.

8 Conclusion

This report has summarised a codebase for fairness-focused data generation. The project offers building blocks for producing matched pairs of images using multiple providers, generating diverse image prompts, synthesising textual scenes and MCQs, orchestrating prompt and metadata pipelines via an agentic framework, and producing short videos. It leverages modern generative models: OpenAI GPT-4o, DALL-E 3, Gemini Imagen, Veo, xAI Grok-2 and Stable Diffusion XL Turbo, while providing configuration-driven pipelines, retry logic and checkpointing to make experiments reproducible. Beyond generation, the framework includes tools to calculate fairness metrics and perform zero-shot classification with integrated gradients, giving researchers the means to audit model behaviour across demographic groups and risk contexts, as per responsible AI principles [7].

Acknowledgements

Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute.

Funding for the research was partly provided through Horizon Europe project AIXpert: An agentic, multi-layer, GenAI-powered backbone to make an AI system explainable, accountable, and transparent (ID: 101214389).

License

References

[1] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[3] Google DeepMind. Google gemini imagen 4: Ai image generation. `https://gemini.google/overview/image-generation/`, 2025. Accessed: 2025-08-25.

[4] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[5] Zihao Lin, Samyadeep Basu, Mohammad Beigi, Varun Manjunatha, Ryan A. Rossi, Zichao Wang, Yufan Zhou, Sriram Balasubramanian, Arman Zarei, Keivan Rezaei, Ying Shen, Barry Menglong Yao, Zhiyang Xu, Qin Liu, Yuxiang Zhang, Yan Sun, Shilong Liu, Li Shen, Hongxuan Li, Soheil Feizi, and Lifu Huang. A survey on mechanistic interpretability for multi-modal foundation models, 2025.

[6] OpenAI. GPT-4o System Card, August 2024. White-paper style system card, version released August 8, 2024. Accessed 2025-04-24.

[7] Shaina Raza, Oluwanifemi Bamgbose, Shardul Ghuge, Fatemeh Tavakol, Deepak John Reji, and Syed Raza Bashir. Developing safe and responsible large language model: Can we balance bias reduction and language understanding in large language models? *arXiv preprint arXiv:2404.01399*, 2024.

[8] Shaina Raza, Arash Shaban-Nejad, Elham Dolatabadi, and Hiroshi Mamiya. Exploring bias and prediction metrics to characterise the fairness of machine learning for equity-centered public health decision-making: A narrative review. *IEEE Access*, 2024.

[9] Jerome H Saltzer. The origin of the "mit license". *IEEE Annals of the History of Computing*, 42(4):94–98, 2020.

[10] Hugging Face Team. Distilgpt2: A distilled version of gpt-2. https://dataloop.ai/library/model/distilbert_distilgpt2/, 2024. Model hosted on Dataloop AI Platform, license Apache-2.0.

[11] Mistral AI Team. Mistral 7b: A 7-billion-parameter language model. https://huggingface.co/mistralai/Mistral-7B-v0.3, 2023. Model hosted on Hugging Face, license Apache-2.0.

[12] xAI. xai api guide: Image generations. https://docs.x.ai/docs/guides/image-generations, 2025. Accessed: 2025-08-25.