8. Repository Structure and Verifiability.

This ensures your private GitHub repository serves as robust, verifiable evidence of independent development, technical expertise, and progressive iteration. It's designed for easy assessor access (grant read-only permissions via invite), with all sensitive elements excluded (e.g., no patient data, no full model weights). The structure follows best practices for medical AI projects (e.g., aligned with NHS Digital Open Source Guidance and MHRA transparency requirements).

Repository Setup Recommendations

- **Repository Name**: ai-diagnostic-reporting-support (private, GitHub.com).
- **Visibility**: Private – share access link in your report (e.g., https://github.com/hitendrasinh/ai-diagnostic-reporting-support).
- **Branches**: main for stable releases; dev for ongoing work; feature branches (e.g., feature/vit-integration).
- **Tools**: Use GitHub Actions for CI/CD (automated testing on push); add .gitignore for Python projects.
- **Access Instructions**: In README.md, include: "Contact hitendrasinh@aksharaiworks.com for access. All code is original and authored by me."
- **Verifiability Features**:
    - Commit history: 50+ commits showing single authorship (your username), with descriptive messages (e.g., "Implemented ViT inference pipeline #3").
    - Releases: Tagged versions (e.g., v0.1: Basic pipeline; v0.2: UI integration).
    - Issues/Boards: Use for tracking tasks (e.g., "Add explainability maps").

Detailed Repository Structure

Here's the folder layout with key files and purposes. Total repo size should be <500MB (use Git LFS for large samples if needed).

- **/backend** (Core services – Python-based API and orchestration)
    - app.py: FastAPI/Flask server for endpoints (e.g., /upload, /analyze, /generate-report).
    - /image_processing: ingest.py, anonymizer.py (DICOM handling).
    - /ai_engine: orchestrator.py (calls ViT and LLM).
    - Purpose: Handles secure data flow and processing.
- **/models** (AI model configurations and scripts)
    - vit_model.py: Inference code using Hugging Face (e.g., load ViT-B/16).
    - llm_report.py: BioGPT/Llama inference for drafting.
    - config.yaml: Hyperparameters, dataset paths (public only).
    - checkpoints/: Small sample checkpoints (e.g., fine-tuned on subset of CheXpert; no full weights – describe training in training_guide.md).

    - Purpose: Reproducible AI components without proprietary data.
- **/frontend** (User interface – Streamlit for simplicity)
    - app.py: Main dashboard script (upload, viewer, editor).
    - /components: Custom widgets (e.g., heatmap overlay, report editor).
    - /static: CSS/images for branding/accessibility.
    - Purpose: Clinician-facing app.
- **/docs** (Documentation and evidence)
    - architecture_diagrams/: All figures (e.g., fig1-high-level-system-architecture.pdf).
    - compliance/: data_protection_statement.md, dtac_self_assessment.xlsx.
    - design_decisions.md: Rationale for choices (e.g., "Why ViT over CNN").
    - samples/: Anonymised inputs/outputs (e.g., report_sample.txt, heatmap_example.png).
    - user_guide.md: Clinician instructions.
    - Purpose: Comprehensive, searchable evidence.
- **/tests** (Testing artefacts)
    - test_pipeline.py: End-to-end integration tests.
    - test_vit.py: Unit tests for analysis engine.
    - results/: coverage_report.html, test_log.txt.
    - Purpose: Verifiable quality assurance.
- **Root Level Files**
    - README.md: Overview, setup, demo guide (see sample below).
    - requirements.txt: Dependencies (e.g., pydicom, torch, transformers, streamlit).
    - .gitignore: Ignores venv, **pycache**, .DS_Store.
    - LICENSE: MIT (or Apache 2.0) for open-source readiness.
    - setup.sh: Bash script for environment setup.

AI-Assisted Diagnostic Reporting Support

Overview

This repository contains the source code for an AI system assisting UK clinicians in drafting radiology reports. Developed independently by Hitendrasinh Rathod.

Key Features:

- Secure DICOM ingestion and anonymisation.

- Vision Transformer for image highlighting.

- LLM-based draft report generation.

- Clinician dashboard with human-in-the-loop review.

Compliance: Fully GDPR/MHRA aligned; no patient data included.

Setup Instructions

1. Clone repo: `git clone https://github.com/yourusername/ai-diagnostic-reporting-support.git`

2. Install dependencies: `pip install -r requirements.txt`

3. Download sample public data: Place in `/data/samples/` (e.g., from MIMIC-CXR).

4. Run backend: `uvicorn backend.app:app --reload`

5. Run frontend: `streamlit run frontend/app.py`

Demo Guide

- **Step 1**: Upload DICOM sample via dashboard.

- **Step 2**: View AI heatmap overlay.

- **Step 3**: Edit/approve AI-generated draft report.

- Watch `/docs/demo_workflow.mp4` for walkthrough.

- Run tests: `pytest tests/`

Repository Structure

- `/backend`: Core API and processing.

- `/models`: AI scripts and configs.

- `/frontend`: Dashboard UI.

- `/docs`: Diagrams, compliance, samples.

- `/tests`: Unit/integration tests.

Development History

See commit log for progressive development (all by @hitendrasinh).

Contact for access/questions: hitendrasinh@aksharai.com

Verifiability Enhancements

- **Commit History Proof**: Ensure messages like "Initial commit: Set up backend ingestion" show progression. Use git log --author="Hitendrasinh Rathod" to verify single authorship.
- **CI/CD**: Add GitHub workflow for auto-testing on PRs.
- **Size Optimization**: Use GitHub LFS for diagrams/videos.

This structure makes your repo a self-contained, professional portfolio piece—easy to navigate, fully verifiable, and directly supporting your report's claims. Upload it now and test access yourself! If needed, I can help with a sample commit script.