1. ⚙️ Project Architecture: The Asynchronous Core

The foundation of your project is designed to be fast and fail-safe, separating the user interface from the heavy computation.

Phase	Technology	Purpose
Frontend/API	**Django REST Framework (DRF)**	**Ingestion:** Receives the uploaded image file (via POST `/api/upload/`).
Queueing	**Celery**	**Asynchrony:** Immediately offloads the heavy work to the background, ensuring the web response is instant.
Message Broker	**Redis**	**Queue:** Acts as the high-speed communication channel where the Celery task ID is temporarily stored until a worker is available.
Worker	**Celery Worker**	**Execution:** Listens to the Redis queue, picks up the task, and executes the entire **processing pipeline** defined in `core/tasks.py`.

2. 🧠 The Processing Pipeline (The `process_image_pipeline` Task)

This task executes the image analysis in a strict, ordered flow.

A. Gatekeeping and Initial Data Capture

Step	Model/Technology	Purpose
Validate Image	**Pillow / Python OS**	**Pruning:** Checks if the file is readable, a valid format, and meets minimum size requirements. If it fails here, the task stops immediately.
Human Detection	**MediaPipe Pose Landmarker**	**Filter:** Checks for the presence of a human subject (if `pose_landmarks` are found). If no body is detected, the image is tagged as `REJECTED`.

Get Dimensions	`get_image_dimensions` (Helper)	**Calibration:** Gets the original image width (w) and height (h) in pixels, which is essential for converting the model's normalized coordinates (0.0 to 1.0) into usable pixel coordinates.

B. Core Extraction and Asset Creation

Step	Model/Technology	Purpose
Pose Keypoints & BBox	**MediaPipe Pose Landmarker**	**Core Data:** Extracts the 33 keypoints (like Nose, Elbows, Knees) and uses these extreme points to calculate the overall **Human Bounding Box** and **Detection Confidence**.
Background Removal	**Rembg (U-2-Net)**	**Asset Cleanup:** Creates a clean, ready-to-use digital asset (transparent **PNG**).
Clothing Mask	**MediaPipe Multi-class Segmenter**	**Analysis Tool:** Creates a precise binary mask showing only the clothing area, which is required for the next step.
Refined Dominant Colors	**Scikit-learn (K-Means)**	**Accuracy:** Clusters **only** the pixels within the clothing mask. This prevents background colors (e.g., blue walls) from contaminating the final color output.

C. Final Attributes and Data Saving

Step	Model/Technology	Purpose
Age/Sex Estimation	**OpenCV DNN (Caffe Models)**	**Optional Data:** Uses the calculated positions of the nose and eyes (from Pose Keypoints) to generate a tightly cropped image of the face, which is then classified for **Estimated Age** and **Gender**.
Smart Error Handling	**Python `try/except` + `kombu.OperationalError`**	**Resilience:** Differentiates between transient errors (which trigger a retry) and permanent structural failures (which mark the task as `FAILED` immediately).
Data Output	**Django ORM**	**Finalization:** Assembles all the results (BBox, Keypoints, Colors, Confidence, Age, Sex, Mask Path) into a comprehensive

		JSON Metadata field and saves the record as `COMPLETED`.