EdgeSync — Feature Logic Documentation

This document provides a comprehensive overview of the core logic and behavior that power EdgeSync's key user-facing features. It outlines how each component processes inputs, performs intelligent transformations, and delivers real-time user experiences. The focus is on conceptual design and high-level workflows — covering data flow, decision logic, and system. The goal is to clearly describe how EdgeSync integrates machine learning, sensing, and user interaction to deliver a seamless, privacy-aware camera experience.

1. Overview

EdgeSync is an intelligent camera experience that combines real-time sensing, lightweight on-device ML, and optional cloud-assisted analysis to provide privacy-preserving transformations, content-aware assistance, and natural capture workflows. The core user-facing features are:

- Privacy Mode (noise injection): This mode applies a layer of AI-generated noise to an image, preserving privacy while maintaining the general structure of the photo.
- Harmonizer: extracts structured information from images (for example, text and dates) and suggests useful actions such as saving reminders or creating calendar events.
- Smile Capture: automatically triggers capture when a majority of visible faces are detected as smiling.
- Gesture and voice interactions: natural controls for focusing, toggling modes, and invoking features without menus.

Each feature is designed to be robust to common mobile-device constraints (limited CPU, variability of camera hardware, and intermittent connectivity) and to fail gracefully with clear user feedback.

2. Privacy Mode — TFLite-driven Pipeline (Logic and Behavior)

Purpose

- Reduce the effectiveness of automated image analysis (face recognition, text extraction, scene parsing) while keeping the image visually usable for humans.

Overall design

- Privacy Mode is implemented as a parameterized image transformation pipeline where the per-image parameters are predicted by an on-device TFLite ensemble model. The model observes lightweight image features and returns a compact set of transformation parameters that control a frequency-aware perturbation generator.

Inputs
- Raw image bytes (RGB or encoded JPEG).
- Optional runtime flags: whether to use the TFLite predictor, a seed for reproducibility, and manual overrides for specific parameters (amplitude, frequency, phase, spatial weighting, blend factor).

Model input features
- The predictor uses a small, fast-to-compute feature vector that summarizes the image at a coarse level. Typical features include normalized width and height, aspect ratio, and other lightweight descriptors that can be computed without full-resolution processing. These features are intentionally small to reduce inference cost and memory pressure.

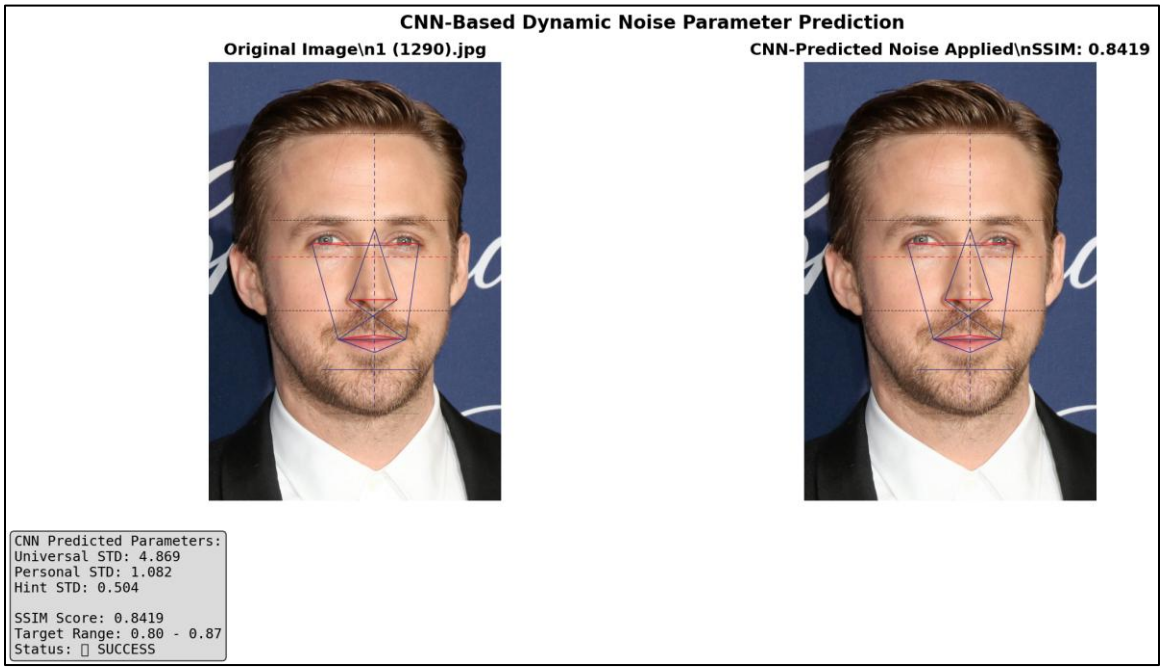
Model outputs
- The model returns a compact set of scalar parameters that control the noise generator, for example:
  - amplitude_factor: controls perturbation strength
  - frequency_factor: selects the frequency bands to emphasize
  - phase_factor: phase offsets for sinusoidal components
  - spatial_factor: controls spatial modulation and weighting
  - temporal_factor: controls dynamic/animation aspects for video
  - noise_seed: integer seed for deterministic randomness
  - blend_factor: final blending strength with the original image

Processing pipeline
1. Preprocess: compute the predictor feature vector from the input image (lightweight sampling or metadata-based features). Convert the image to a representation that isolates luminance where needed without expensive full-resolution changes.
2. Predict parameters: run the TFLite ensemble model on-device to receive the transformation parameters.
3. Generate perturbation: create a frequency-aware noise field using parameterized sinusoids and band-limited filters.
4. Blend and reconstruct: combine the perturbation with the original image using the blend factor and encode for storage.

Runtime integration and safety
- On-device TFLite inference is performed with resource constraints in mind.
- Run inference on a background worker (isolate) to keep the UI responsive.
- Limit native interpreter threads (e.g., numThreads=1) to reduce native thread creation pressure.
- Prefer quantized or optimized model variants to reduce memory and CPU usage.
- Timebox inference and fall back to default parameters if inference fails or times out.

Outputs and metadata
- The system returns the transformed (noised) image and metadata describing parameters used, processing time, and whether model-predicted parameters were applied.

Performance and quality trade-offs
- Model-driven parameter selection lets the system adapt to different scenes while keeping inference cost low.
- The generation step can be tuned for quality vs. speed; simplified perturbation for low-end devices reduces CPU/memory needs.



3. Harmonizer — Logic and Behavior

Purpose
- Convert captured image content into actionable, structured suggestions (calendar events, reminders, contacts, notes).

Inputs
- A captured image containing potentially useful textual or contextual information.

Processing pipeline
1. Text extraction: OCR to extract textual content.

2. Content analysis: identify entities (dates/times, contact details, tasks).
3. Suggestion synthesis: produce ranked, structured suggestions with confidence scores.
4. Present and execute: show suggestions and offer one-tap actions to execute.

Outputs
- Actionable suggestions with confidence metadata; execution only after user confirmation.

4. Smile Capture — Logic and Behavior

Purpose
- Enable hands-free group photos by automatically taking a photo when most faces in the scene are smiling.

Inputs
- Live camera preview frames with detected faces and smile-confidence scores.

Processing pipeline
1. Continuous detection: facial detection yields smile probabilities.
2. Aggregation: majority-based decision across faces.
3. Debounce & countdown: validates condition and avoids spurious captures.
4. Trigger capture: capture issued and image saved if condition holds.

5. Gesture Features

Purpose
- Provide natural, responsive camera interactions (focus, mode toggles, feature invocation via touch/gestures).