# **EdgeSync Project Documentation**

## 1. Project Overview

EdgeSync is a feature-rich, intelligent camera application built with Flutter. It goes beyond standard camera functionality by integrating on-device AI to provide smart features like automatic smile detection, voice-activated controls, and a unique privacy-preserving noise injection system. The application is designed to be performant on a wide range of devices through its adaptive performance-tuning capabilities.2. Technical Features & Implementation

This section details the advanced, technically-driven features of the EdgeSync application.

### **Privacy Mode (Noise Injection)**

- **Concept**: This mode applies a layer of Al-generated noise to an image, preserving privacy while maintaining the general structure of the photo.
- **Implementation**: It uses a TFLite model to predict noise parameters for the injection process.
- **Gallery Integration**: Images processed with Privacy Mode are saved directly to the public device gallery using the gal package to ensure they are immediately visible.

#### **Harmonizer Service**

The Harmonizer is a feature designed to process images for aesthetic improvements.
 When enabled, it presents a dialog after a photo is captured to apply its effects, showcasing post-processing capabilities.

### **Smile Capture**

- Automatic Photo Capture: The app uses on-device face detection (google\_mlkit\_face\_detection) to detect smiling faces in the camera's view.
- Countdown Timer: When a majority of detected faces are smiling, a 3-second countdown is automatically triggered, after which a photo is taken. This allows for hands-free group photos without a manual shutter press.

#### **Voice Commands**

- Hands-Free Control: A microphone button enables voice commands to control the camera, using the speech\_to\_text package for on-device recognition.
- **Functionality**: Users can execute commands like "take picture", "start video", "stop video", and "switch camera".

• **Implementation**: The voice command service listens for a command and then automatically turns off, preventing continuous listening.

### **Dynamic & Animated Gallery**

- **Dynamic Updates**: The gallery is not just a static view. When a new "privacy" image is generated via the Harmonizer or Privacy Mode, it is dynamically added to the gallery's PageView for immediate viewing.
- Animated UI: In the gallery, the Harmonizer and Privacy buttons are not always visible.
  They appear with a smooth fade and scale animation when the user taps on a photo, providing a clean and interactive UI.

## 2. Performance Optimization

The application includes a robust system to ensure it runs smoothly on both old and new devices.

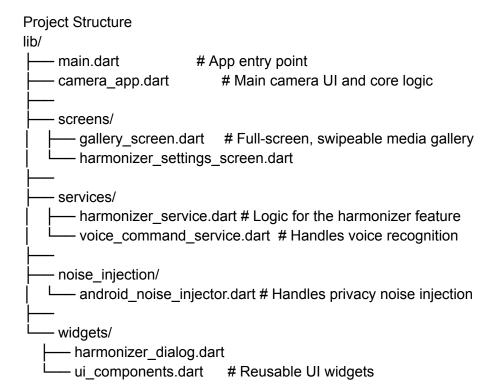
- **Automatic Detection**: On startup, the app runs a quick benchmark to determine if the device is "old" or "new".
- Dynamic Processing Intervals:
  - Old Device Mode: Uses a longer interval (1500ms) between face detection frames to reduce CPU load and prevent crashes.
  - New Device Mode: Uses a shorter interval (500ms) for more responsive detection.
- Manual Toggle: A "speed" icon in the UI allows the user to manually override the performance mode, with a SnackBar providing clear feedback.

### 3. Technical DetailsArchitecture

The app is built using Flutter and follows a standard widget-based architecture. Key components are separated into services (HarmonizerService, VoiceCommandService), screens (GalleryScreen), and the main camera logic (CameraApp). Key Dependencies

- camera: Core camera functionality.
- google\_mlkit\_face\_detection: For on-device smile detection.
- speech\_to\_text: For voice command recognition.
- tflite\_flutter: For running the noise injection model (currently partially disabled).
- image: For advanced image manipulation during noise injection.
- gal: For saving images and videos to the public device gallery.

- permission\_handler: For managing camera and microphone permissions.
- video\_player: For playing videos within the gallery.



## 4. Setup and UsagePrerequisites

- Flutter SDK
- Android Studio or Xcode
- A physical device with a camera

#### Installation

- 1. Clone the repository.
- Move the project to a file path that does not contain spaces (e.g., C:\dev\EdgeSync). This is critical to avoid Gradle build errors on Windows.
- 3. Install dependencies: flutter pub get
- 4. Run the app: flutter run