Project Proposal

# Project Overview

## Project Title

Hand Control App

## Client Name

## Project Summary

Develop an Android application that uses the front‑facing camera to recognize hand gestures and translates them into system commands, providing a touch‑free experience similar to Huawei/Honor hand‑control features.

## Objectives

* Enable users to scroll, swipe and take screenshots using hand gestures.
* Implement smart activation that works only when the user is looking at the screen.
* Provide a robust, on‑device gesture recognition solution with low latency and minimal battery impact.
* Improve accessibility for users who cannot easily touch the screen.

# Business Requirements

## Goals

* Create a market‑ready hands‑free control app for Android.
* Increase accessibility and convenience for a broad user base.
* Establish a revenue stream through app sales or premium features.

## Target Audience

Android smartphone users, especially those seeking hands‑free interaction or requiring accessibility assistance.

## Expected Outcomes

* A functional Android app that reliably detects predefined hand gestures.
* System‑wide actions (scroll, swipe, screenshot) triggered via gestures.
* Smart activation that reduces accidental triggers and conserves battery.

## Success Metrics

* Gesture detection accuracy ≥ 90% in real‑world conditions.
* User satisfaction score ≥ 4 out of 5 in beta testing.
* Battery consumption increase ≤ 5% during continuous use.
* App store rating ≥ 4 stars within the first 3 months.

## Monetization Strategy

Paid app on Google Play with optional in‑app purchase for advanced gesture sets or ad‑free experience.

# Technical Requirements

## Core Features

* Scroll (hand wave up/down)
* Swipe (left/right palm swipe)
* Screenshot (fist clench)
* Smart Activation (face detection to confirm user attention)

## Core Features & Functionalities

* Real‑time hand landmark detection using ML Kit Pose Detection.
* Mapping of landmark patterns to predefined gestures.
* Android Accessibility Service to execute system‑wide actions.
* Camera feed handling via CameraX.
* Battery‑optimised processing and background handling.

## Tech Stack

* Android SDK
* Kotlin
* CameraX
* Google ML Kit (on‑device pose detection)
* Android Accessibility Service API

## Integration Needs

* Android OS accessibility framework
* Camera hardware via CameraX
* ML Kit libraries
* Face detection module for smart activation

## Security & Compliance

* Runtime permissions for camera and accessibility service.
* No data is transmitted off‑device; all processing is on‑device.
* Compliance with Google Play privacy policies.

## Performance Criteria

* Gesture recognition latency ≤ 150 ms.
* Frame processing at ≥ 15 fps on mid‑range devices.
* Battery impact ≤ 5 % per hour of continuous use.

# App Flow

## App Flow Summary

* Launch app → Request camera & accessibility permissions → Initialize CameraX feed.
* Run ML Kit pose detection on each frame → Detect hand landmarks.
* Apply gesture‑recognition logic → Verify smart activation via face detection.
* If activation confirmed, trigger corresponding system action via Accessibility Service.
* Provide visual/audio feedback → Continue loop until app is closed.

# Project Scope

## Inclusions

* Full Android app development (Kotlin, CameraX, ML Kit).
* Gesture detection engine for scroll, swipe, screenshot.
* Smart activation using on‑device face detection.
* Accessibility Service integration for system actions.
* Beta testing, performance optimisation, and documentation.

## Exclusions

* iOS version development.
* Custom machine‑learning model training from scratch.
* Hardware accessories or external sensors.
* Post‑launch marketing or app store optimisation services.

## Deliverables

* Source code repository.
* Signed APK / AAB ready for Google Play submission.
* Technical design document.
* Test plan and test reports.
* User guide and developer documentation.

## Milestones

* Phase 1 – Proof of Concept (camera feed & hand landmark detection).
* Phase 2 – Gesture Logic & On‑Screen Actions.
* Phase 3 – System Integration & Smart Activation.
* Phase 4 – Polishing, Performance Tuning & Beta Testing.

## Estimated Timeline & Pricing

Approximately 4 months of development; budget to be defined in the Budget & Costing section.

# Timeline & Resources

## Estimated Duration

4 months (16 weeks)

## Team Roles

* Project Manager
* Lead Android Developer
* ML Engineer (ML Kit integration)
* UI/UX Designer
* QA/Test Engineer

## Dependencies

* Google ML Kit SDK availability
* CameraX library compatibility with target Android versions
* Access to Android devices for testing (various manufacturers & OS versions)

# Budget & Costing

## Estimated Budget

$80,000 USD

## Cost Breakdown

|  |  |
| --- | --- |
| Item | Amount |
| Project Management | $10,000 |
| Android Development | $35,000 |
| Machine Learning Integration | $12,000 |
| UI/UX Design | $8,000 |
| Quality Assurance & Testing | $10,000 |
| Contingency (10%) | $5,000 |

# Risk Assessment

## Potential Risks

* False positive/negative gesture detection leading to poor user experience.
* High battery consumption on older devices.
* Compatibility issues across diverse Android OEM customizations.
* Permission denial by users (camera or accessibility).

## Mitigation Strategies

* Iterative testing with diverse device pool and fine‑tuning thresholds.
* Implement adaptive frame rate and power‑saving modes.
* Follow Android compatibility guidelines and conduct OEM‑specific testing.
* Provide clear permission rationale and fallback UI.

# Other Notes

The app targets the accessibility and hands‑free utility markets, offering a differentiating feature set that can be expanded with additional gestures in future releases.