

Skin Cancer Detection Mobile App

Software Design Document

Version 1.0

Nikiraj Konwar, Lawson Darrow,
Nicolas Rincon-Sperenza, Christian Stevens

Faculty Advisor: Dr. Nematzadeh

Table of Contents

1.0 INTRODUCTION

- 1.1 Purpose
- 1.2 Background and Objectives
- 1.3 Definitions and Acronyms

2.0 SYSTEM OVERVIEW

- 2.1 Image Capture Module
- 2.2 Machine Learning Inference Engine
- 2.3 Results Processing and Display
- 2.4 Data Management System

3.0 SYSTEM ARCHITECTURE

- 3.1 Architectural Design
- 3.2 Decomposition Description

4.0 DATA DESIGN

- 4.1 Data Description
- 4.2 Data Dictionary

5.0 COMPONENT DESIGN

- 5.1 Presentation Layer Components
- 5.2 Application Logic Layer Components
- 5.3 Machine Learning Layer Components
- 5.4 Hardware Abstraction Layer Components

6.0 HUMAN INTERFACE DESIGN

- 6.1 Overview of User Interface
- 6.2 Screen Images and Mockups

7.0 ALGORITHM DESIGN

- 7.1 Image Processing Pipeline
- 7.2 Risk Classification Algorithm

8.0 PERFORMANCE AND OPTIMIZATION

9.0 REFERENCES

1.0 INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of the Skin Cancer Detection Mobile Application, a cross-platform mobile solution for preliminary skin lesion analysis using convolutional neural networks and on-device machine learning inference.

1.2 Background and Objectives

Skin cancer is one of the most common cancers worldwide, with early detection significantly improving treatment outcomes. Current barriers to early detection include limited access to dermatologists, high consultation costs, and geographical constraints. This project aims to develop an accessible mobile application that enables users to perform preliminary skin lesion assessments using their smartphone cameras.

The primary objectives are:

- Provide real-time skin lesion analysis with medical-grade accuracy
- Ensure complete user privacy through on-device processing
- Deliver intuitive user experience suitable for non-technical users
- Maintain cross-platform compatibility across iOS and Android devices

1.3 Definitions and Acronyms

Acronym	Term
CNN	Convolutional Neural Network
TFLite	TensorFlow Lite
ISIC	International Skin Imaging Collaboration
UI/UX	User Interface/User Experience
API	Application Programming Interface

Term	Definition
Benign Lesion	Non-cancerous skin abnormality requiring monitoring
Malignant Lesion	Potentially cancerous skin growth requiring medical attention
Confidence Score	Probability measure indicating prediction certainty
Risk Level	Categorized assessment (Low/Medium/High) based on analysis

2.0 SYSTEM OVERVIEW

The Skin Cancer Detection Mobile Application comprises four major subsystems that work collaboratively to deliver the complete user experience.

2.1 Image Capture Module

This module handles all aspects of image acquisition, including camera access, real-time preview, image quality validation, and user guidance. The system provides framing overlays to help users capture optimal images of skin lesions, with automatic quality checks to ensure analysis suitability.

2.2 Machine Learning Inference Engine

The core analytical component utilizes a pre-trained CNN model deployed via TensorFlow Lite. This engine processes captured images through a standardized pipeline including resizing, normalization, and quality enhancement before performing classification inference.

2.3 Results Processing and Display

This subsystem interprets raw model outputs into medically meaningful information. It converts probability scores into risk categories, generates appropriate recommendations, and formats results for clear user comprehension with necessary medical disclaimers.

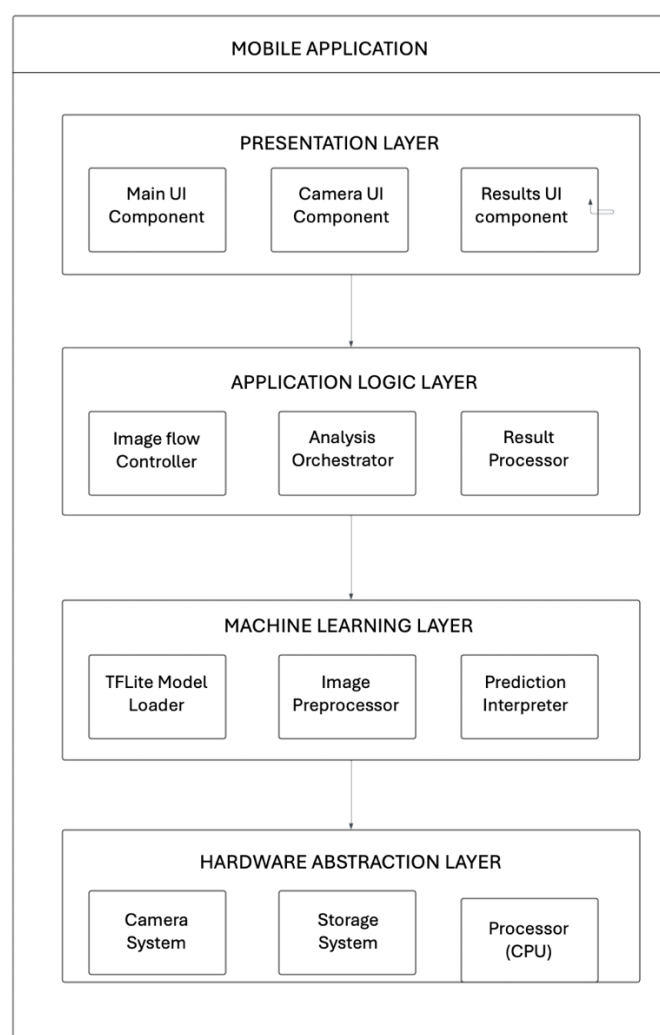
2.4 Data Management System

Responsible for local storage of analysis history, user preferences, and application data. This system ensures efficient data retrieval for historical tracking while maintaining strict privacy standards through on-device storage only.

3.0 SYSTEM ARCHITECTURE

3.1 Architectural Design

The system follows a layered architecture pattern with clear separation of concerns:



3.2 Decomposition Description

Each layer maintains strict interfaces with adjacent layers, ensuring modularity and testability. The Presentation Layer handles all user interactions, the Application Logic Layer coordinates workflow, the Machine Learning Layer performs analytical computations, and the Hardware Abstraction Layer interfaces with device capabilities.

4.0 DATA DESIGN

4.1 Data Description

The system manages three primary data types:

Image Data: Raw camera captures and processed images stored in JPEG/PNG format with standardized 224x224 pixel dimensions for model compatibility.

Model Data: TensorFlow Lite model files containing pre-trained CNN parameters for skin lesion classification, optimized for mobile deployment.

Analysis Metadata: Structured records containing timestamps, classification results, confidence scores, and user annotations stored in local database.

4.2 Data Dictionary

Data Element	Format	Size	Description
Camera Image	JPEG	2-5 MB	High-resolution lesion capture
Processed Image	Tensor	150 KB	Normalized 224x224x3 input tensor
TFLite Model	.tflite	0.42 MB	Optimized CNN model weights
Analysis Record	JSON	5-10 KB	Result metadata and user data
User Preferences	JSON	1-2 KB	Application configuration settings

5.0 COMPONENT DESIGN

5.1 Presentation Layer Components

Main UI Component

- **Responsibility:** Application navigation and dashboard management
- **Key Methods:**
 - `navigateToScreen(screenName)`: Handles transitions between application views
 - `displayDashboard()`: Renders main navigation interface with recent activities
 - `handleUserPreferences()`: Manages application settings and configurations

Camera UI Component

- **Responsibility:** Real-time camera interface with user guidance
- **Key Methods:**
 - `initializeCamera()`: Configures camera hardware with optimal settings
 - `showFramingOverlay()`: Displays lesion centering guidelines
 - `captureHighQualityImage()`: Executes image capture with quality validation

Results UI Component

- **Responsibility:** Presentation of analysis outcomes and recommendations
- **Key Methods:**
 - `displayRiskAssessment()`: Shows classification results with visual indicators
 - `presentMedicalDisclaimer()`: Ensures appropriate usage context
 - `generateRecommendations()`: Creates actionable medical guidance

5.2 Application Logic Layer Components

Image Flow Controller

- **Responsibility:** Orchestrates complete analysis workflow from capture to results
- **Key Methods:**
 - `orchestrateAnalysisPipeline()`: Coordinates sequential processing steps
 - `validateInputQuality()`: Ensures image suitability for analysis
 - `handleProcessingErrors()`: Manages pipeline failures gracefully

Analysis Orchestrator

- **Responsibility:** Coordinates machine learning pipeline execution
- **Key Methods:**
 - `executeClassificationWorkflow()`: Runs end-to-end ML analysis
 - `monitorPerformanceMetrics()`: Tracks processing time and resource usage
 - `cacheIntermediateResults()`: Optimizes repeated operations

Result Processor

- **Responsibility:** Interpretation and formatting of model outputs
- **Key Methods:**
 - `interpretModelOutput()`: Converts probabilities to medical insights
 - `generateRiskCategorization()`: Determines Low/Medium/High risk levels
 - `formatForDisplay()`: Prepares results for user presentation

5.3 Machine Learning Layer Components

TFLite Model Loader

- **Responsibility:** Management of machine learning model lifecycle
- **Key Methods:**
 - `loadModelWeights()`: Initializes TensorFlow Lite interpreter
 - `verifyModelIntegrity()`: Ensures model file validity and compatibility
 - `optimizeInferenceSettings()`: Configures for device-specific performance

Image Preprocessor

- **Responsibility:** Preparation of images for model consumption
- **Key Methods:**
 - `standardizeImageDimensions()`: Resizes to 224x224 pixels
 - `normalizePixelValues()`: Adjusts color ranges and contrast
 - `enhanceImageQuality()`: Improves analysis suitability through filtering

Prediction Interpreter

- **Responsibility:** Conversion of model outputs to medical assessments
- **Key Methods:**
 - `calculateConfidenceMetrics()`: Computes prediction certainty scores
 - `applyClinicalKnowledge()`: Incorporates medical domain expertise
 - `generateExplanatoryContent()`: Creates user-friendly result explanations

5.4 Hardware Abstraction Layer Components

Camera System Interface

- **Responsibility:** Abstraction of device camera capabilities
- **Key Methods:**
 - `acquireCameraAccess()`: Handles permission management
 - `configureCaptureSettings()`: Optimizes for medical imaging
 - `manageHardwareResources()`: Coordinates camera with other system functions

Storage System Manager

- **Responsibility:** Local data persistence and retrieval
- **Key Methods:**
 - `saveAnalysisHistory()`: Stores results with metadata
 - `retrieveHistoricalData()`: Provides access to previous analyses
 - `manageStorageCapacity()`: Handles space allocation and cleanup

Processor Resource Manager

- **Responsibility:** Computational resource allocation and monitoring
- **Key Methods:**
 - `allocateComputeResources()`: Manages CPU/GPU usage for ML tasks
 - `monitorThermalConditions()`: Prevents device overheating
 - `optimizePowerConsumption()`: Balances performance with battery life

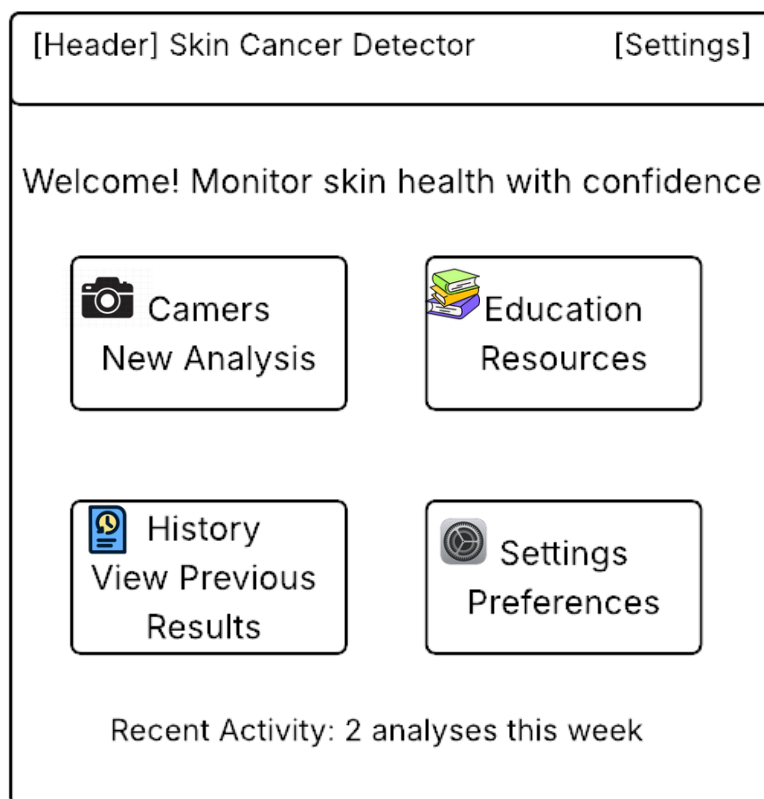
6.0 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface


The user interface follows material design principles with emphasis on accessibility and simplicity. The design prioritizes clear information hierarchy, consistent navigation patterns, and responsive layouts adaptable to various screen sizes and orientations.

6.2 Screen Images and Mockups

Main Dashboard Screen




Camera Interface Screen

[Back]	Capture Lesion Image	[Flash: 
--------	----------------------	---

LIVE PREVIEW

CAPTURE GUIDELINES


- Center lesion
- Good lighting
- Steady position


[Gallery]	[ CAPTURE]	[Help]
-----------	--	--------

Distance: 6-8 inches from skin recommended

Results Display Screen

[Back]	Analysis Complete	[Save] [Share]
--------	-------------------	----------------

 **IMPORTANT:** Educational tool only – Not a diagnosis
Always consult healthcare professionals

Result: BENIGN LESION
Confidence: 87% []

Risk Level: LOW

Recommendations:

- Regular self-monitoring
- Annual professional checkup
- Sun protection practices

[Save Result]	[Learn More]	[New Analysis]
---------------	--------------	----------------

7.0 ALGORITHM DESIGN

7.1 Image Processing Pipeline

BEGIN Image Processing Algorithm

1. CAPTURE raw image from camera hardware
2. VALIDATE image quality metrics:
 - Check focus clarity using Laplacian variance
 - Assess lighting conditions via histogram analysis
 - Verify lesion visibility through region detection
3. IF quality inadequate THEN REQUEST recapture
4. RESIZE image to 224x224 pixels maintaining aspect ratio
5. NORMALIZE pixel values to range [0, 1]
6. APPLY color correction and contrast enhancement
7. RETURN processed image tensor

END Algorithm

7.2 Risk Classification Algorithm

BEGIN Risk Classification Algorithm

INPUT: model_output, confidence_score

IF confidence_score < 0.6 THEN

 RETURN "INCONCLUSIVE - Poor image quality"

END IF

IF model_output > 0.7 THEN

 risk_category = "HIGH"

 recommendation = "Consult dermatologist promptly"

ELSE IF model_output > 0.4 THEN

 risk_category = "MEDIUM"

 recommendation = "Schedule professional evaluation"

ELSE

 risk_category = "LOW"

 recommendation = "Continue regular monitoring"

END IF

RETURN (risk_category, recommendation)

END Algorithm

8.0 PERFORMANCE AND OPTIMIZATION

8.1 Performance Targets

- **Application Startup:** <3 seconds cold start time
- **Image Processing:** <2 seconds from capture to analysis ready
- **Model Inference:** <3 seconds for complete classification
- **Memory Usage:** <200MB peak memory consumption
- **Battery Impact:** <5% battery drain per 10 analyses

8.2 Optimization Strategies

Memory Management: Implement streaming image processing to avoid loading full-resolution images into memory simultaneously. Use efficient data structures and automatic cache management.

Computational Efficiency: Leverage hardware acceleration through TensorFlow Lite delegates. Implement batch processing where possible and optimize model architecture for mobile inference.

Storage Optimization: Use intelligent compression for stored images and implement configurable retention policies for analysis history.

Power Management: Implement power-aware processing modes that adjust computational intensity based on battery level and thermal conditions.

9. Reference

"Convolutional Neural Network." Wikipedia, Wikimedia Foundation, 2023, en.wikipedia.org/wiki/Convolutional_neural_network.

"TensorFlow Lite." TensorFlow, Google, 2023, www.tensorflow.org/lite.

"ISIC Archive." International Skin Imaging Collaboration, 2023, www.isic-archive.com.

"ABCDE Rule of Melanoma." American Academy of Dermatology, 2023, www.aad.org/public/diseases/skin-cancer/types/common/melanoma/abcdes.