Software Test Documentation

Mobile App for Real-Time Skin Cancer Detection

Prepared by: Nikiraj Konwar, Lawson Darrow, Nicolas Rincon-Speranza, Christian Stevens

Faculty Advisor: Dr. Nematzadeh

1 Introduction

1.1 Overview

This test plan defines procedures for functional and non-functional requirements of the Skin Cancer Detection mobile application. Testing ensures that all required features and behaviors are implemented, and function reliably under normal and abnormal conditions.

1.2 Purpose

The purpose of this plan is to establish systemic testing guidelines for each requirement documented in the Software Requirements Specification. Systematic testing is documented for each requirement to confirm correct system behavior, identify failures, and validate compliance with user and technical expectations.

1.3 Approach

Each functional and non-functional requirement will be translated into test cases. Each test case specifies inputs, expected outputs, dependencies, and procedures. Test inputs include both typical and atypical values. Execution will cover unit testing, integration testing, system testing, and acceptance testing.

1.4 References

- IEEE Std 829-2008 (Software Test Documentation Standard).
- ISIC Dataset Documentation.
- Team Project Plan and Presentation documents.
- Research publications on skin lesion classification with CNNs.

1.5 Contents

The remainder of this document is structured as follows:

- Section 2: Test Case Scenarios for Functional Requirements
- Section 3: Test Case Scenarios for Non Functional Requirements
- Section 4: Conclusion

2 Test Case Scenarios for Functional Requirements

2.1 FR1 - Capture images using smartphone camera

Test Case: Capture image under valid conditions.

- **Objective**: Verify ability to capture lesion photo in real-time.
- **Inputs**: User taps camera icon. User takes photo of lesion.
- **Expected Outcome**: Image is stored in memory and passed to preprocessing.
- **Unusual Input**: Extremely low light, camera covered, blurred motion.
- **Dependencies**: Camera permission granted.
- **Procedure**: Open app, grant permission, capture image in varied lighting.

2.2 FR2 – Upload images from gallery

Test Case: Select existing image.

- **Objective**: Verify gallery import functions.
- **Inputs**: User selects lesion photo from gallery.
- **Expected Outcome**: Imported image is accepted and processed.
- **Unusual Input**: Invalid format (BMP), corrupted file, non-lesion images.
- **Dependencies**: Gallery access permission.
- **Procedure**: Upload valid PNG, invalid BMP, corrupted image, and selfie.

2.3 FR3 – Provide framing overlays

Test Case: Display overlay during live preview.

- **Objective**: Verify visual guide visibility.
- **Inputs**: User opens the camera interface.
- **Expected Outcome**: Circular guide shown consistently.
- **Unusual Input**: Device with unusual aspect ratio.
- **Procedure**: Run app on multiple devices, check overlay alignment.

2.4 FR4 – Preprocess images

Test Case: Resize and normalize.

- **Objective**: Verify preprocessing pipeline.
- **Inputs**: User takes valid lesion photo.
- **Expected Outcome**: 224x224 normalized tensor produced.
- **Unusual Input**: Image with extreme aspect ratio.
- **Procedure**: Input images 16:9, 4:3, portrait, confirm standard output.

2.5 FR5 – Classify lesions

Test Case: Run classification model.

- **Objective**: Confirm inference runs correctly.

- **Inputs**: Preprocessed tensor.
- **Expected Outcome**: Benign/malignant probability returned.
- **Unusual Input**: Random noise image.
- **Procedure**: Input valid lesion and blank/noise image, observe result.

2.6 FR6 - Display confidence percentage

Test Case: Show confidence score.

- **Objective**: Ensure transparency.
- **Inputs**: Model output probabilities; risk assessment.
- **Expected Outcome**: Confidence displayed as percentage.
- **Procedure**: Run classification, verify UI shows number.

2.7 FR7 – Present disclaimers

Test Case: Disclaimer always shown.

- **Objective**: Confirm legal compliance.
- **Inputs**: Display result screen.
- **Expected Outcome**: Disclaimer present for all outputs.
- **Procedure**: Run multiple analyses, verify disclaimer presence.

2.8 FR8 – Provide links to medical resources

Test Case: Resource links accessible.

- **Objective**: Verify resource availability.
- **Inputs**: View result screen.
- **Expected Outcome**: Clickable links to healthcare resources. User taps link.
- **Dependencies**: Internet required.
- **Procedure**: Test link opening on both iOS and Android.

29. FR9 – Operate offline

Test Case: Run app with no connectivity.

- **Objective**: Validate local inference.
- **Inputs**: Airplane mode enabled. User capture/upload photo.
- **Expected Outcome**: Analysis completes without connectivity.

2.10 FR10 – Store no images externally

Test Case: Confirm storage restrictions.

- **Objective**: Validate privacy.
- **Inputs**: User capture/upload lesion.
- **Expected Outcome**: Images only stored locally, no cloud sync.
- **Procedure**: Inspect storage, confirm no external transfer.

2.11 FR11 – Allow image retakes

Test Case: Retake option visible.

- **Objective**: Verify retry support.
- **Inputs**: User capture/upload blurry image.
- **Expected Outcome**: User prompted to retake.

2.12 FR12 – Provide educational tips

Test Case: Display tips section.

- **Objective**: Confirm awareness material present.
- **Inputs**: Access educational section.
- **Expected Outcome**: Text/images on lesion warning signs shown.

2.13 FR13 – Log anonymized usage data

Test Case: Log generation without sensitive data.

- **Objective**: Confirm analytics privacy.
- **Inputs**: Perform multiple analyses.
- **Expected Outcome**: Logs contain timestamps and app usage, no personal health data.

2.14 FR14 – Display states (loading, analyzing, result)

Test Case: UI state transitions.

- **Objective**: Verify UX flow.
- **Inputs**: Run analysis.
- **Expected Outcome**: Loading → Analyzing → Result displayed.

2.15 FR15 – Deploy on Android and iOS

Test Case: Install and run on both platforms.

- **Objective**: Verify cross-platform support.
- **Inputs**: Use iPhone and Android phone.
- **Expected Outcome**: Full functionality on both.

3 Test Case Scenarios for Non-Functional Requirements

3.1 NFR1 – Predictions within 5 seconds

Test Case: Time analysis.

- **Inputs**: Capture lesion.
- **Expected Outcome**: Classification < 5 seconds.

3.2 NFR2 – Processing local only

Test Case: Monitor network activity.

- **Expected Outcome**: No inference data leaves device.

3.3 NFR3 – Handle corrupted/irrelevant images

Test Case: Input corrupted file or selfie.

- **Expected Outcome**: Error prompt or feedback without crash.

3.4 NFR4 – Intuitive interface

Test Case: First-time user task completion.

- **Expected Outcome**: Successful navigation without tutorial.

3.5 NFR5 – Android/iOS compatibility

Test Case: Functional equivalence test.

- **Expected Outcome**: Identical behavior across devices.

3.6 NFR6 – No exposure of sensitive data

Test Case: Inspect logs.

- **Expected Outcome**: No health-related identifiers.

3.7 NFR7 – Accessibility support

Test Case: Enable large fonts/high contrast.

- **Expected Outcome**: UI elements scale and contrast correctly.

3.8 NFR8 – Modular architecture

Test Case: Swap ML model.

- **Expected Outcome**: Minimal code changes required.

3.9 NFR9 – Future dataset/model support

Test Case: Load updated TFLite model.

- **Expected Outcome**: New model integrated successfully.

3.10 NFR10 – Healthcare ethics standards

Test Case: Audit app compliance.

- **Expected Outcome**: Disclaimer, privacy, ethical design verified.

4 Conclusion

This test plan details all functional and non-functional requirements, ensures verification through normal and unusual input cases, and validates compliance with privacy, usability, and performance constraints.

Functional requirements define the "what," objectives of specific features. The section defines various normal and unusual inputs and expected outputs. Inputs are user-triggered behaviors, and follow procedural step-by-step actions.

Non functional requirements describe qualities of the system. The section defines constraints like speed, security, and portability, focusing on how these qualities should behave. The conditions, time, memory usage, system logs, are measured by the system.