Software Test Plan

Project: Breast Cancer Detection System
Prepared by: Woroma Dimkpa, Kahlel Cardona, Taratong Dolinsky
Date: 02/17/2026

1. Introduction

1.1 Purpose

This Software Test Plan (STP) defines the testing strategy, scope, approach, resources, schedule, and deliverables for the Breast Cancer Detection System. The system is designed to classify mammogram images as benign or malignant using Convolutional Neural Networks implemented in PyTorch.

The purpose of this document is to ensure that the developed system satisfies all functional and non-functional requirements defined in the Software Requirements Specification (SRS). This document provides structured guidance for validating correctness, reliability, performance, and usability of the system.

1.2 Scope

The testing process covers the following modules:

- Data Loading Module
- Preprocessing Module
- Dataset & DataLoader Module
- Model Architecture Module
- Training Engine
- Evaluation Module
- Visualization Module
- Model Export Function

Testing includes:

- Functional testing
- Integration testing
- Performance testing
- Model validation testing
- System testing

1.3 References

- Software Requirements Specification (SRS)

- Software Design Document (SDD)
- IEEE 829 Standard for Software Test Documentation
- IEEE 29119 Software Testing Standard

2. Test Items

The following components will be tested:

1. Image loading and metadata parsing
2. Image preprocessing operations (resizing, normalization, augmentation)
3. Dataset batching and stratified splitting
4. CNN model forward propagation
5. Backpropagation and weight updates
6. Loss computation and optimizer functionality
7. Evaluation metrics computation
8. Model saving and loading

3. Test Strategy

3.1 Testing Levels

3.1.1 Unit Testing

Each module is tested independently:

- Data loading functions
- Image transformation functions
- Model forward pass
- Loss function computation
- Optimizer updates

Unit testing ensures individual components behave as expected.

3.1.2 Integration Testing

Integration testing verifies interaction between:

- Dataset → DataLoader
- DataLoader → Model
- Model → Training Engine
- Training Engine → Evaluation Module

Data flow correctness is validated during forward and backward passes.

3.1.3 System Testing

System testing validates the entire pipeline:

Raw Image → Preprocessing → Tensor Conversion → Model → Evaluation → Visualization

The objective is to confirm that the complete system works correctly under realistic conditions.

3.1.4 Performance Testing

Performance testing evaluates:

- Training time per epoch
- GPU utilization
- Memory consumption
- Convergence stability

Testing is performed on GPU-enabled environments (e.g., Tesla T4 or P100).

3.1.5 Model Validation Testing

Model performance is validated using:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion matrix

Stratified train/validation/test splitting ensures unbiased evaluation.

4. Test Environment

4.1 Hardware Requirements

- GPU: NVIDIA Tesla T4 / P100
- RAM: Minimum 8 GB
- Storage: Minimum 10 GB for dataset

4.2 Software Requirements

- Python 3.x
- PyTorch
- NumPy
- Pandas
- Matplotlib
- scikit-learn

Testing is conducted in a cloud-based notebook environment.

5. Test Cases

5.1 Functional Test Cases

Test Case 1: Image Loading

Test ID	**TC-01**
Objective	Verify images load correctly
Input	Valid image path
Expected Result	Image tensor returned without error

Test Case 2: Missing File Handling

Test ID	**TC-02**
Objective	Verify error handling for missing images
Input	Invalid file path
Expected Result	Error logged, system continues execution

Test Case 3: Preprocessing

Test ID	**TC-03**

Objective	Verify resizing and normalization
Input	Raw image
Expected Result	Tensor of correct shape (C,H,W)

Test Case 4: Forward Pass

Test ID	**TC-04**
Objective	Verify model produces predictions
Input	Batch tensor
Expected Result	Output tensor with shape (batch_size, 2)

Test Case 5: Backpropagation

Test ID	**TC-05**
Objective	Verify gradients update weights
Input	Loss value
Expected Result	Model parameters updated

Test Case 6: Model Export

Test ID	**TC-06**
Objective	Verify trained model saves successfully
Input	Trained model
Expected Result	.pth file generated

6. Acceptance Criteria

The system will be considered acceptable if:

1. All functional test cases pass.
2. No critical runtime errors occur during training.
3. Model achieves acceptable validation accuracy (≥ 75% baseline for Tiny CNN).
4. Stratified splitting maintains class distribution.
5. Model export and reload functionality works correctly.

7. Risks and Mitigation

Risk	Impact	Mitigation
Class imbalance	Biased model predictions	Use class-weighted loss
Overfitting	Poor generalization	Use validation monitoring
GPU memory overflow	Training interruption	Reduce batch size
Corrupted images	System crash	Exception handling in data loader

8. Test Deliverables

The following deliverables will be produced:

- Test Plan document
- Test case specification
- Test execution report
- Performance evaluation report
- Confusion matrix and accuracy plots
- Trained model file (.pth)

9. Schedule

Testing activities are performed in the following order:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Performance Testing
5. Final Validation Testing

Testing is conducted iteratively alongside model development to ensure early detection of defects.

10. Conclusion

This Software Test Plan ensures systematic validation of the Breast Cancer Detection System. By following structured testing procedures aligned with IEEE standards, the system's correctness, reliability, and performance are verified before deployment.

The testing strategy ensures traceability to all functional requirements defined in the SRS and supports confidence in the system's ability to perform accurate mammogram classification in practical settings.