Software Test Plan (STP)

Project: Real-Time Whiteboard Transcription System

Author: Ariel Blinder, Saar Attarchi

# 1. Introduction

This Software Test Plan outlines the testing strategy for the Real-Time Whiteboard Transcription System.  
The system captures, processes, and transcribes handwritten text from whiteboard images with support for video-based transcription.  
This plan focuses on current image-based features while anticipating upcoming video stream integration.

# 2. Test Items

- OCR and Handwriting Recognition Engine  
- Text Normalization and Output Formatter  
- UI for uploading images and displaying results  
- Error Logging and Feedback System  
- Video Frame Capture & Streaming Transcription Engine

# 3. Features to be Tested

- Image upload and validation  
- Accurate OCR using LLaMA 4 API  
- Text cleanup and formatting  
- Download/export of output text  
- UI feedback for success/error states

# 4. Features Not to be Tested

-Hebrew handwriting recognition  
-Integration with collaborative platforms (e.g., Google Drive, MS Teams)  
-Advanced user authentication and session persistence  
-Accessibility and screen‑reader support

# 5. Testing Strategy

* - Unit Testing: Core functions in preprocessing, normalization, and text cleaning  
  - Integration Testing: Input to OCR to Output flow and API connectivity  
  - System Testing: Simulate full workflows via frontend interface  
  - Acceptance Testing: Validate image-to-text output matches manual transcription  
  Example Test Cases:
* TC‑01 (Unit): normalize\_text() with mixed‑case, punctuation‑rich string ➜ returns lowercase, punctuation‑free output.
* TC‑02 (Integration): Upload valid PNG image via UI ➜ triggers OCR ➜ returns JSON with extracted text within 5 seconds.
* TC‑03 (System): End‑to‑end flow — user uploads sample lecture image and downloads .txt file ➜ accuracy ≥ 90 %.

# 6. Test Environment

- Frontend: Localhost via React  
- Backend: Python server (Flask/FastAPI)  
- OCR: LLaMA 4 API  
- Video: FFmpeg for frame extraction  
- Tools: Postman, PyTest, BrowserStack  
- Browsers: Chrome, Firefox

# 7. Responsibilities

* Ariel: Backend functionality & unit tests
* Saar: Frontend testing & user‑interface validation

# 8. Schedule

|  |  |
| --- | --- |
| Phase | Dates |
| Unit Testing | May 7 – 13 |
| Integration Testing | May 14 – 20 |
| System Testing | May 21 – 24 |
| Acceptance Testing | May 25 – 31 |

# 9. Risks and Contingencies

- OCR API Unavailable : Use mock text or switch to a different API  
- Difficulty recognizing chronological order of written text: Enhance processing filters or use AI assistance to order strokes.  
- Video Frame Sampling Too Heavy: Add throttling & dynamic frame rate adjustment  
- API Rate Limits from LLaMA 4: Implement request queue/caching or switch API.