Test Plan Document

Project Name: Automatic Face Recognition Attendance System

Version: 1.0

Prepared By: Manogya Singh

Date: Jun 26, 2025

Document Status: Final

1. Introduction

This test plan outlines the testing approach for the Automatic Face Recognition Attendance System. The application is designed to automate student attendance using facial recognition technology, making the attendance process faster and more reliable compared to traditional methods.

The system allows users to register, log in, capture student images, and automatically mark attendance when students are detected by the webcam. This document serves as a guide for the testing team to understand what needs to be tested and how to execute testing activities.

2. Objectives

The main goals of testing this application are:

- Ensure users can register, log in, and reset passwords securely.

- Verify that student images are captured properly and stored locally and in Firebase.

- Confirm that face encodings are generated accurately from saved images.

- Validate that real-time face recognition works for attendance marking and prevents duplicates within 30 seconds.

- Ensure student records can be added, viewed, updated, and deleted correctly.

- Verify that attendance records are displayed and exported to Excel successfully.

- Test manual attendance adjustment functionality.

- Validate user interface responsiveness and input handling.

- Confirm the system regenerates encodings upon new student additions.

3. Scope of Testing

- **What We'll Test:**

- User registration, login, and password reset functionality.

- Student image capture and storage (local + Firebase).

- Face encoding generation and regeneration.

- Real-time face recognition and attendance marking with duplicate prevention.

- Manual adjustment of student attendance counts.

- Student data management (view, update, delete).

- Attendance reporting and Excel export.

- User interface responsiveness and stability across all screens.

- Input validation and error handling for empty or invalid data.

- Application exit functionality.

- Password hashing security.

- **What We Won't Test:**

- Firebase backend infrastructure (assuming it's stable).

- Webcam hardware functionality (not part of application logic).

- Network connectivity issues (assumed to be available).

- Third-party library internals (face_recognition library).

- Operating system specific bugs.

4. Test Items

The following modules will be tested:

1. User Authentication Module - Registration, Login, and Password Reset.

2. Student Image Capture Module - Photo capture and storage.

3. Face Encoding Module - Encoding generation and regeneration.

4. Attendance Marking Module - Face detection, recognition, duplicate prevention, and manual adjustment.

5. Attendance Reporting Module - Display and Excel export of records.

6. Student Management Module - CRUD operations on student records.

7. User Interface Module - Overall interface, navigation, and button functionality.

8. Input Validation Module - Handling of invalid or empty inputs.

5. Features to be Tested

- User registration with a unique email and strong password (with password hashing).

- Login with valid and invalid credentials.

- Password reset functionality for existing users.

- Capturing student images via webcam and saving them locally and to Firebase.

- Generating and updating the face encoding file (`EncodeFile.p`).

- Real-time face recognition during attendance sessions.

- Preventing duplicate attendance marks for the same student within 30 seconds.

- Manually adjusting a student's attendance count.

- Viewing, updating, and deleting student details in Firebase and locally.

- Displaying attendance records in a separate window.

- Exporting attendance data to an Excel file (`student_attendance.xlsx`).

- Real-time display of the student list in the main interface.

- All UI buttons functioning correctly without application crashes.

- Graceful handling of empty fields and invalid inputs during form submission.

- Clean exit from image capture mode and attendance sessions.

- Application closes properly using the exit button.

6. Features Not to be Tested

- Webcam driver compatibility.

- Firebase server uptime and performance.

- Email verification (if not implemented).

- Multi-language support (not in scope).

- Mobile app version (desktop only).

7. Test Strategy

- **Testing Approach:**

We'll follow a systematic approach starting with basic functionality and moving to complex scenarios.

- **Testing Levels:**

- Component Testing: Individual module testing.

- Integration Testing: Testing interactions between modules.

- System Testing: End-to-end workflow validation.

- User Acceptance Testing: Final validation before deployment.

- **Testing Types:**

- Functional Testing - Verify all features work as expected.

- Positive Testing - Test with valid inputs.

- Negative Testing - Test with invalid/boundary inputs.

- Usability Testing - Check user-friendliness.

- Security Testing - Password hashing and data protection.

- Regression Testing - Retest after bug fixes.

- **Test Design Techniques:**

- Equivalence Partitioning.

- Boundary Value Analysis.

- Error Guessing.

- Decision Table Testing.

8. Test Environment

- **Hardware:**

- Processor: Intel Core i5 or equivalent.

- RAM: 8 GB minimum.

- Webcam: HD webcam (720p or higher).

- Internet: Stable broadband connection.

- **Software:**

- Operating System: Windows 10/11 (64-bit).

- Python Version: 3.8 or higher.

- Required Libraries: opencv-python, face_recognition, firebase-admin, openpyxl, cvzone.

- Database: Firebase Realtime Database.

- Storage: Firebase Storage.

- **Tools Used:**

- Manual Testing using Excel for test case management.

- Firebase Console for backend verification.

- Screenshots for defect reporting.

9. Test Data

We'll prepare the following test data:

- **Valid Users:**

- Email: testuser1@gmail.com, Password: Test@1234

- Email: testuser2@gmail.com, Password: Secure#567

- Stored in: `login_details.xlsx`

- **Invalid Login Attempts:**

- Unregistered email: test123@gmail.com

- Incorrect password: bing0@123

- **Student Data:**

- Student ID: 2024001, 2024002, 2024003

- Names: Rajesh Kumar, Priya Sharma, Amit Verma

- Branches: CSE, ECE, ME

- Years: 1, 2, 3

- **Invalid Student Data:**

- Empty fields.

- Special characters in ID.

- Future starting years.

10. Entry and Exit Criteria

- **Entry Criteria (Before we start testing):**

- Application build is ready and deployed in the test environment.

- All required dependencies and libraries are installed.

- Firebase is configured and accessible.

- Test environment setup is complete.

- Test cases are written and reviewed.

- Test data is prepared.

- **Exit Criteria (When we can stop testing):**

- All planned test cases are executed.

- No critical or high-severity bugs are open.

- All medium and low-priority bugs are documented.

- Test summary report is prepared.

- Code coverage meets minimum requirements (80%).

- Sign-off received from stakeholders.

11. Roles and Responsibilities

Role	Name	Responsibilities
QA Tester	Manogya Singh	Design test cases, execute tests, log defects, retest fixes

12. Test Deliverables

The following documents will be created during testing:

1. Test Plan (this document).
2. Test Scenarios - High-level test conditions.
3. Test Cases - Detailed step-by-step test procedures.
4. Defect Report - Bug tracking document.
5. Test Execution Report - Results of test execution.
6. Test Summary Report - Final testing summary.

13. Defect Management

Bug Reporting Process:

1. Tester discovers a bug during execution.
2. Log the bug with all details (steps, screenshots, environment).
3. Assign severity and priority.
4. Developer reviews and fixes.

5. Tester retests after the fix.

6. Close if passed, reopen if still failing.

- **Severity Levels:**

- Critical: Application crashes, data loss.

- High: Major feature not working.

- Medium: Feature works with issues.

- Low: Minor UI issues, typos.

- **Priority Levels:**

- High: Fix immediately.

- Medium: Fix before release.

- Low: Can be fixed in the next release.

14. Risks and Mitigation

| Risk | |Impact | |Mitigation Strategy |

|---|---|---|

| Test environment not available | High | Arrange backup environment, coordinate with DevOps early.|

| Webcam not working properly | Medium | Keep a spare webcam, test with different cameras.|

| Firebase connection issues | High | Use local mock data for initial testing if possible.|

| Delayed bug fixes | Medium | Prioritize critical bugs, maintain clear communication with developers.|

| Insufficient test data | Low | Create a test data generator script.|

| Face recognition accuracy issues | High | Test with diverse lighting conditions and angles. |

15. Testing Schedule

| Activity | Duration | Start Date | End Date |

|---|---|---|---|

| Test Plan Preparation | 2 days | Jun 22, 2025 | Jun 23, 2025 |

| Test Scenario Creation | 2 day | Jun 24, 2025 | Jun 25, 2025 |

| Test Case Design | 2 days | Jun 26, 2025 | Jun 27, 2025 |

| Test Environment Setup | 1 day | Jun 28, 2025 | Jun 28, 2025 |

| Test Execution | 2 days | Jun 29, 2025 | Jun 30, 2025 |

| Defect Reporting & Retesting | 1 day | Jul 1, 2025 | Jul 1, 2025 |

| Regression Testing | 1 day | Jul 2, 2025 | Jul 2, 2025 |

| Test Summary Report | 1 day | Jul 3, 2025 | Jul 3, 2025 |

Total Duration: 12 days

16. Assumptions and Dependencies

- **Assumptions:**

- Python and all required libraries are correctly installed.

- Firebase account is set up and configured.

- Webcam provides clear images for face detection.

- Network connectivity is stable during testing.

- Test environment mirrors production setup.

- **Dependencies:**

- Timely bug fixes from the development team.

- Firebase service availability.

- Access to test devices and webcam.

- Stakeholder availability for clarifications.

End of Test Plan Document