Software Requirements Specification (SRS)

Project Title: Three Level Password System Using Python

Created by - Creative TechTians

GitHub Repository: https://github.com/thisisdhritiman1439/Three-Level-Password-System

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for a secure **Three-Level Password System** developed using Python and Streamlit. The system enhances access control by using **three layers of authentication**—something the user has (email/password), something the user knows (QR code with 6-digit passcode), and something the user is (User's Picture Recognition).

1.2 Scope

Provides a **highly secure authentication system** for protecting sensitive user data
Supports:

- **User registration** with personal credentials and face image
- **Multi-factor login** with three levels of authentication
- **Secure file upload and access** (only after full authentication)
- **Login activity logging** for auditing and traceability

Grants access **only after successful completion** of all three authentication levels.

1.3 Definitions, Acronyms, and Abbreviations

- **MFA:** Multi-Factor Authentication
- **QR:** Quick Response
- **CSV:** Comma-Separated Values
- **UI:** User Interface
- **SHA256:** Secure Hash Algorithm 256-bit
- **JSON:** JavaScript Object Notation

2. Overall Description

2.1 Product Perspective

This is a **standalone Python-Streamlit application**. It does not require third-party APIs for core functionality. All user credentials and secure files are stored locally for privacy and security.

2.2 Product Functions

- Register new users with credentials and biometric image
- Login using 3-level authentication
- Generate a QR code with a time-sensitive code
- Validate the QR code manually by input
- Perform face matching using uploaded image
- Grant or deny access to secure file based on validation
- Record login attempts in CSV log

2.3 User Characteristics

- Should have basic digital literacy
- Comfortable using QR scanners and uploading images
- No programming knowledge required

2.4 Operating Environment

- Python 3.10+
- Streamlit Web App
- Compatible with Chrome, Firefox, Edge
- Deployable on Streamlit Cloud or local systems

2.5 Design and Implementation Constraints

- Face recognition only through image upload (webcam not used)
- No external database used; local storage only
- Sensitive files and facial data stored in dedicated folders
- QR code not scanned automatically—manual entry used instead

2.7 User Documentation

- README.md for setup and instructions
- Inline user guidance within Streamlit UI
- Code comments for maintainers/developers

2.8 Assumptions and Dependencies

- Users will have valid and clear face images
- Users can access QR scanner apps
- Stable browser support (modern web browser)

3. Specific Requirements

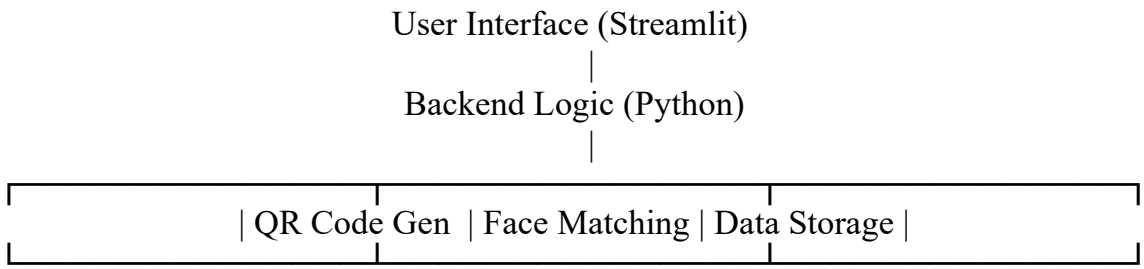
3.1 Functional Requirements

- **FR1**: User Registration (Email, Password, Face Image, Secure File Upload)
- **FR2**: Login with registered email and password
- **FR3**: QR Code Generation with time-based 6-digit token
- **FR4**: QR Code input validation
- **FR5**: Face Image Upload for Verification
- **FR6**: Face matching using pre-stored image with encoding
- **FR7**: Grant access only after successful validation of all three factors
- **FR8**: Log every successful login attempt to login_log.csv

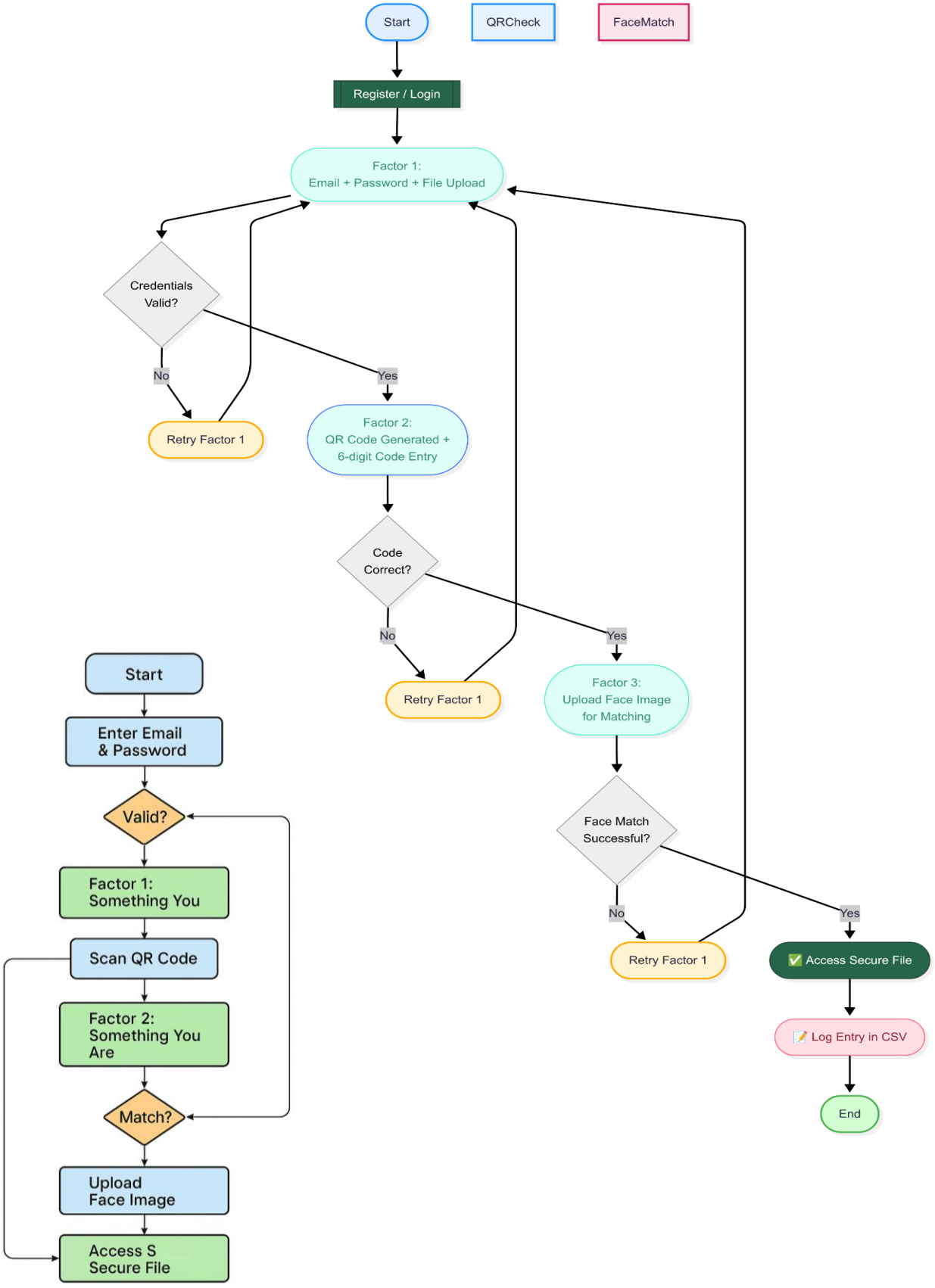
3.2 Non-Functional Requirements

- **NFR1**: Passwords stored as SHA256 hashes
- **NFR2**: Maximum 5 seconds processing time per step
- **NFR3**: Clean, responsive UI with clear instructions
- **NFR4**: Images and secure files stored safely in structured folders
- **NFR5**: Error handling for file corruption, wrong input, or missing data

3.4 Technical Architecture Flowchart



3.3 System Workflow Diagram (Flow Chart)



4. Glossary

Terms	Description
Authentication	Process of verifying a user's identity before granting access to a system.
Three-Factor Authentication	A security process that uses three different types of credentials: something you have, something you know, and something you are.
QR Code	A machine-readable code used as the second authentication factor, decoded by a scanner or camera.
Facial Recognition	Biometric identification method using facial features as the third authentication factor.
OTP (One-Time Password)	A temporary password sent via email to validate identity.
Streamlit	A Python-based open-source framework used to build and deploy interactive web apps.
CSV File	A simple file format used to store tabular data, such as user login timestamps.
Pandas	A Python library used for data manipulation and analysis, especially CSV operations.
Secure File Upload	Functionality allowing authenticated users to upload or access sensitive files.
Encryption	The process of converting data into a coded format to prevent unauthorized access.
User ID	A unique identifier assigned to each user for authentication and logging purposes.

5. Appendix

A. Tools and Libraries Used

- **Streamlit**: Web interface and frontend logic
- **Python 3:** Backend
- **OpenCV (cv2)**: Face recognition
- **pyqrcode / pypng**: QR code generation
- **Pandas**: Data manipulation and storage
- **datetime**: To log login timestamps
- **base64**: For secure image encoding
- **CSV**: Used as backend data store for user logs and face image matching

B. Sample CSV File Structure

csv
CopyEdit
email,password,image_data
user@example.com,hashed_password,

C. Sample Use Case Flow

1. **User registers** with email, password, and uploads their face.
2. **Login initiated**: Email/password verified.
3. **QR Code** displayed → User scans and enters 6-digit code.
4. **Facial Recognition** by uploading the user's Photo
5. Upon success, secure files are displayed and **login timestamp** is saved.

6. Team Members Roles

Name	Role
Khandakar Nafees Hossain	Project Lead & Developer
Dhritiman Bera	Developer & Designer
Parthib Mahapatra	Tester & Deployment

7.Sign-off and Approval

Name	Signature	Date
Mr. Subhabrata Sengupta		
Dr. Rupayan Das		
Khandakar Nafees Hossain		
Dhritiman Bera		
Parthib Mahapatra		