Software Requirements Specification

For

Home Security System using IOT

Version 2.0

Prepared by RP9002 Team 3

Keshav Memorial Institute of Technology

1st March, 2025.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Revision - 1	16/4/2025	Inclusion of Database Design and Implementation of newer modules	2

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the functional and non-functional requirements of a home security system using Flask, ESP-WHO, and IoT devices. The system will detect and recognize faces using face recognition technology and differentiate between known and unknown individuals.

1.2 Document Conventions

This document follows standard **IEEE SRS formatting**. Key conventions include:

- **Bold text** for section headings.
- *Italicized text* for emphasis on critical terms.
- `Monospace font` for code snippets or technical terms.
- All functional requirements are labeled as **REQ-#** for easy reference.

1.3 Intended Audience and Reading Suggestions

This SRS is intended for the following stakeholders:

- **Developers:** To implement system functionality based on the defined requirements.
- **Security Analysts:** To validate security and performance expectations.
- **End Users/Homeowners:** To understand the system's capabilities and limitations.
- **IoT Engineers:** To integrate the system with smart home devices.

1.4 Product Scope

The Home Security System is a **real-time, AI-powered surveillance system** that utilizes **Flask, OpenCV, and IoT devices** to enhance residential security. This system will **detect, recognize, and classify faces** using advanced **face recognition algorithms**. The primary goal is to identify **known individuals** and alert homeowners about **unknown visitors or potential intruders**.

1.5 References

This project refers to:

1. ***IEEE Standard for Software Requirements Specifications (IEEE 830-1998)***.
2. ***Flask official documentation***: *https://flask.palletsprojects.com*
3. ***ESP-WHO official documentation***: espressif/esp-who: Face detection and recognition framework

2. Overall Description

2.1 Product Perspective

The Home Security System is a **standalone, AI-based security solution** designed for **residential properties**. It integrates **computer vision, IoT connectivity, and web-based control** to provide a **smart surveillance ecosystem**. The system replaces **traditional CCTV setups** by offering **automated facial recognition, remote access, and real-time alerts**.
It will be implemented as a **client-server model**, where:

- **Cameras and IoT devices** capture live video streams.
- A **Flask-based backend** processes images and performs face recognition.
- A **web interface** allows users to monitor security footage and receive alerts.

2.2 Product Functions

The Home Security System is a **standalone, AI-based security solution** designed for **residential properties**. It integrates **computer vision, IoT connectivity, and web-based control** to provide a **smart surveillance ecosystem**. The system replaces **traditional CCTV setups** by offering **automated facial recognition, remote access, and real-time alerts**.
It will be implemented as a **client-server model**, where:

- **Cameras and IoT devices** capture live video streams.
- A **Flask-based backend** processes images and performs face recognition.
- A **web interface** allows users to monitor security footage and receive alerts.

2.3 User Classes and Characteristics

- **Homeowners:** Primary users who access the system for real-time monitoring and alerts. They can manage known faces, review security logs, and configure system settings.
- **System Administrators:** Users responsible for maintaining the face recognition database, managing device configurations, and ensuring system security.
- **Guests/Visitors:** Individuals whose faces are detected but are not part of the known face database. They may be added to the system if authorized by the homeowner.

2.4 Operating Environment

The system runs on a combination of hardware and software components. It requires a processing unit such as a ESP-32 for local computation, along with inbuilt cameras for video input. IoT devices like smart doorbells and motion sensors enhance security functionality. The software stack includes Python with Flask, OpenCV, ESP-WHO Library for backend processing, while a web-based UI built with HTML, CSS, and JavaScript provides user interaction. The system operates over a local network but can be extended to cloud-based remote monitoring.

2.5 Design and Implementation Constraints

The system must handle real-time processing, requiring optimized algorithms for camera input. The performance depends on the computational power available. Privacy considerations require that face data be securely stored and managed according to regulations. The design must ensure scalability, allowing additional cameras and users without performance degradation.

2.6 Assumptions and Dependencies

The system assumes users have a basic network infrastructure to connect cameras and IoT devices. It requires adequate processing power to perform real-time face recognition and may need internet access for cloud-based remote monitoring. The accuracy of the face recognition system depends on pre-trained models and the quality of input images.

3. External Interface Requirements

3.1 User Interfaces

The system provides a web-based interface accessible from desktop and mobile devices. The dashboard displays live video feeds from multiple cameras, logs of detected faces, and system notifications. Users can add or remove authorized faces, configure system settings, and review security alerts. The interface follows a simple and intuitive design to ensure ease of use.

3.2 Hardware Interfaces

The system interacts with IP cameras, USB cameras, and IoT-enabled security devices such as smart doorbells and motion sensors. It uses standard communication protocols to retrieve video feeds and sensor data. Hardware components must support integration with the Flask-based backend for seamless operation.

3.3 Software Interfaces

The backend is built using Python with Flask and integrates with ESP-WHO for image processing and face recognition. It communicates with a database for storing user data, detected face logs, and security events. The system may expose APIs for third-party integration, allowing external applications to access video feeds and alert notifications.

3.4 Communications Interfaces

The system requires a network connection to transmit video data and alerts. Local operation is supported over Wi-Fi or Ethernet, while remote monitoring requires internet access. The communication between devices and the backend uses secure protocols such as HTTPS and WebSocket for real-time updates. Optional cloud integration enables users to receive alerts via email or mobile push notifications.

4. System Features

4.1 Real Time Facial Recognition

4.1.1 Description and Priority

This feature enables the system to detect and recognize faces in real-time using video streams from multiple cameras. It ensures that known individuals are identified, and unknown faces trigger alerts. This is a high-priority feature, as it forms the core functionality of the security system.

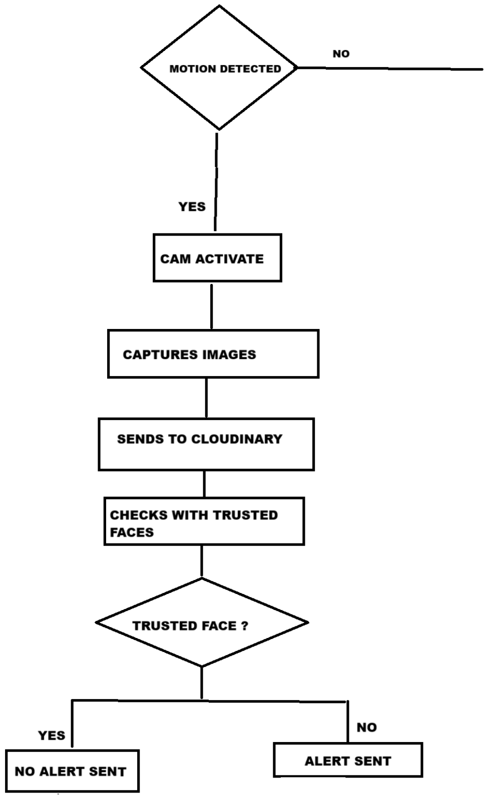
4.1.2 Stimulus/Response Sequences

- A camera captures a live video stream.
- The system detects a face in the frame.
- If the face is recognized as a known individual, it is marked as such.
- If the face is unknown, an alert is triggered, notifying the user.
- The detected face is logged in the database for future reference.

4.1.3 Functional Requirements

REQ-1: The system shall detect and recognize faces in real-time
REQ-2: The system shall differentiate between known and unknown individuals
REQ-3: The system shall trigger an alert for unknown faces.
REQ-4: The system shall log detected faces with timestamps.

LIVEFEED WORKFLOW



4.2 Intruder Alert System

4.2.1 Description and Priority

This feature ensures that any unknown individual detected by the system generates an immediate alert to the user. The alert system is a high-priority feature to enhance security.

4.2.2 Stimulus/Response Sequences

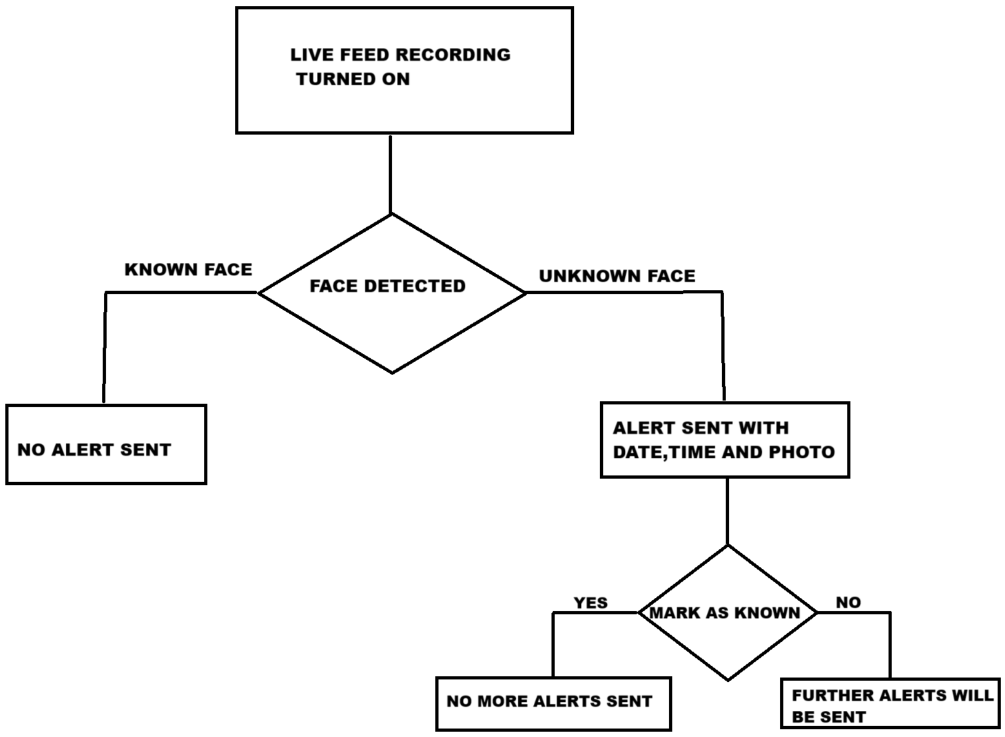
- A face is detected and recognized as unknown.
- The system logs the unknown face.
- A notification is sent via web or IoT device.

4.2.3 Functional Requirements

REQ-5: The system shall notify the user when an unknown face is detected.

REQ-6: The system shall allow users to configure notification settings.

REQ-7: The system shall store unknown face data for review.


LIVE FEED RECORDING TURNED ON
KNOWN FACE
FACE DETECTED
UNKNOWN FACE
NO ALERT SENT
ALERT SENT WITH DATE,TIME AND PHOTO
YES
MARK AS KNOWN
NO
NO MORE ALERTS SENT
FURTHER ALERTS WILL BE SENT

4.3 Web-Based Dashboard

4.3.1 Description and Priority

The system provides a web-based interface for users to monitor live feeds, manage the database of known faces, and configure security settings. This is a high-priority feature for accessibility and user control.

4.3.2 Stimulus/Response Sequences

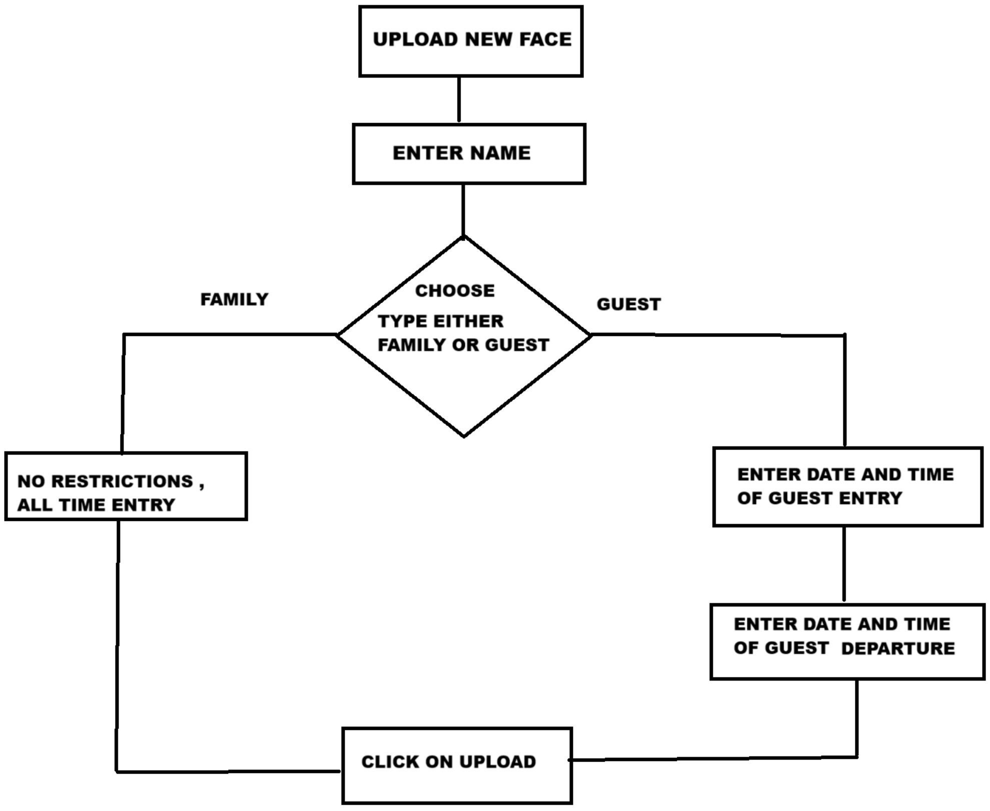
- The user accesses the dashboard through a browser.
- The interface displays live video feeds from cameras.
- The user can view detected faces and manage system settings.
- Alerts and logs are displayed in real-time.

4.3.3 Functional Requirements

REQ-11: The system shall provide a web-based interface for monitoring.

REQ-12: The system shall allow users to add or remove known individuals.

REQ-13: The system shall display real-time logs of detected faces and alerts.


UPLOAD NEW FACE
ENTER NAME
CHOOSE TYPE EITHER FAMILY OR GUEST
FAMILY
GUEST
NO RESTRICTIONS , ALL TIME ENTRY
ENTER DATE AND TIME OF GUEST ENTRY
ENTER DATE AND TIME OF GUEST DEPARTURE
CLICK ON UPLOAD

4.4 Secure Data Storage and Logging in Cloud

4.4.1 Description and Priority

This feature ensures that all detected face data, security events, and user actions are logged securely. It is a high-priority feature for security auditing and forensic investigation.

4.4.2 Stimulus/Response Sequences

- A face is detected and classified.
- The system logs the detected face with a timestamp.
- All security events (e.g., alerts, user actions) are recorded.

4.4.3 Functional Requirements

REQ-14: The system shall store detected face data with timestamps.

REQ-15: The system shall maintain logs of all security events.

REQ-16: The system shall allow users to review past detections and alerts.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system must be capable of processing a video feed with minimal latency. Face recognition should occur in real-time with an average response time of under one second per frame. The system should maintain at least 30 FPS for smooth video streaming and ensure low memory consumption for efficient resource utilization.

5.2 Safety Requirements

The system must ensure that data storage and access comply with security best practices to prevent unauthorized access. In the event of a system failure, an automatic alert should notify the user. Power backup solutions should be considered for uninterrupted monitoring.

5.3 Security Requirements

User authentication is required to access system settings and face database management. The system should support encrypted data storage and transmission using secure protocols. Logs of all detected faces should be maintained to facilitate forensic investigations if needed. Unauthorized access attempts should trigger security alerts.

5.4 Software Quality Attributes

The system must be reliable and operate continuously without crashes. It should be scalable to support additional cameras and users without performance degradation. Maintainability should be ensured with modular code design, and usability should be prioritized with an intuitive user interface.

5.5 Business Rules

Only authorized users should be able to modify face recognition databases and access security logs. Alert notifications should be configurable based on user preferences, allowing customization of security levels. The system should comply with data privacy laws regarding facial recognition and biometric data handling.

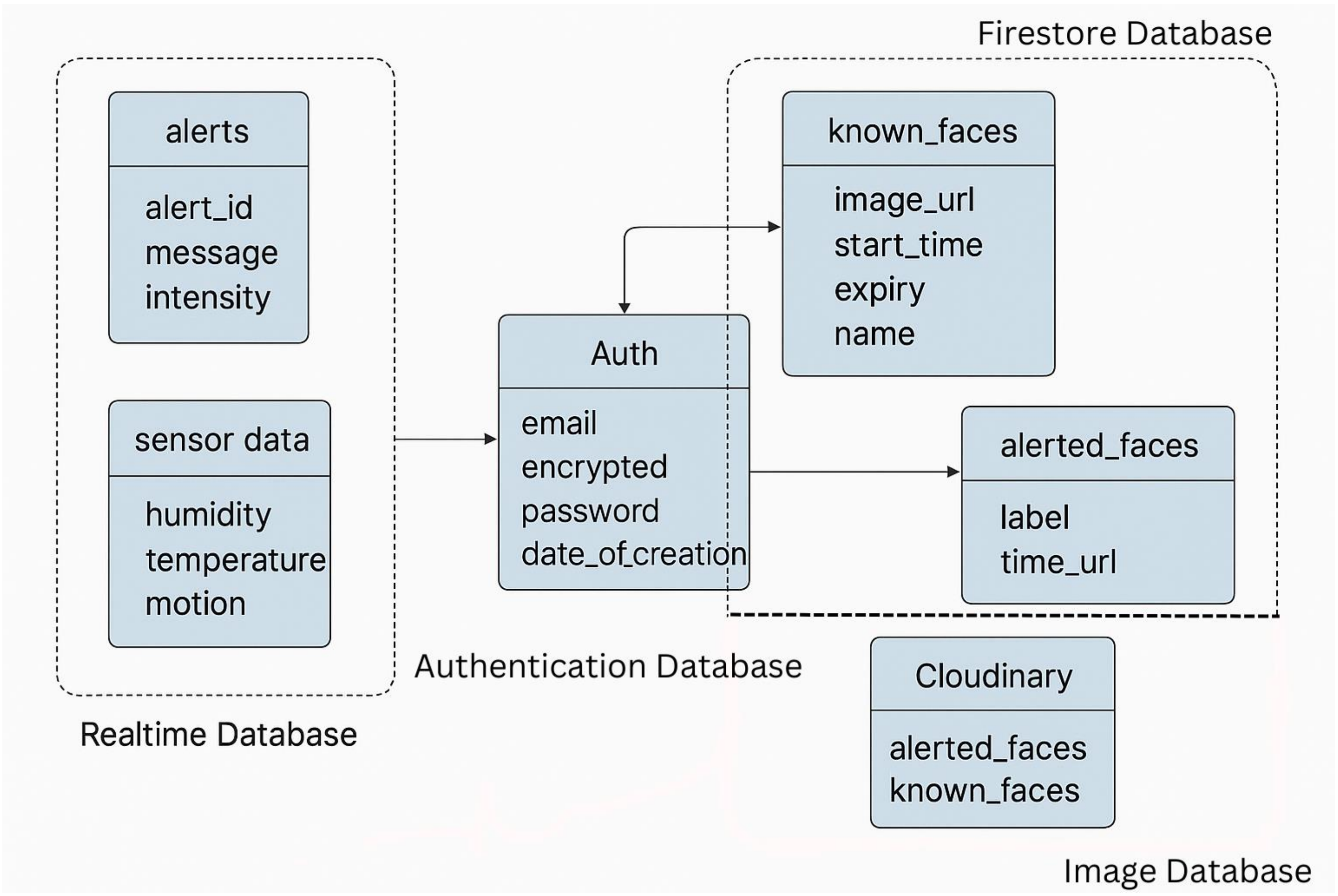
6. Other Requirements

Any other requirements that are stumbled upon will be duly listed here.

Appendix A: Glossary

- ***AI (Artificial Intelligence)*** *– The simulation of human intelligence in machines, used in this system for face recognition.*
- ***IoT (Internet of Things)*** *– A network of interconnected devices that communicate and exchange data, such as smart cameras and sensors.*
- ***Flask*** *– A lightweight Python web framework used to build the backend of the system.*
- ***ESP-WHO*** *– A C-based library used for image and video processing made specifically for ESP-32.*
- ***Face Recognition*** *– The process of identifying and verifying individuals based on facial features.*
- ***Cloudinary Database*** *– A structured storage system where detected face logs, user data, and security events are stored.*

Appendix B: Analysis Models


Firestore Database
alerts
alert_id
message
intensity
known_faces
image_url
start_time
expiry
name
Auth
email
encrypted
password
date_of_creation
sensor data
humidity
temperature
motion
alerted_faces
label
time_url
Authentication Database
Cloudinary
alerted_faces
known_faces
Realtime Database
Image Database