Software Design Specification

[Smart Lens CCTV Surveillance System]

Project Code	*22-F-49*
Supervisor	*Madam Faryal Shamsi*
Co-Supervisor	*----------*
Project Team	*Ali Raza Memon (023-22-0200)* *Aadil Shah (023-22-0106)* *Waseem Mazari (023-22-0102)*
Submission Date	*04-11-25*

Supervisor Signature:----------

Table of contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose of Document

This Software Design Specification (SDS) provides a detailed architectural and component-level design for the "Smart Lens CCTV Surveillance System". Its primary purpose is to translate the comprehensive functional and non-functional requirements defined in the accompanying Software Requirements Specification (SRS) into a structured technical blueprint. This document guides the development, implementation, and testing teams, serving as a foundational agreement and technical reference for the project's construction.

1.2. Definitions, acronyms, and abbreviations

Term/Acronym	Definition
AI	Artificial Intelligence
CNN	Convolutional Neural Network
RTSP	Real-Time Streaming Protocol (used for video stream access)
FCM	Firebase Cloud Messaging
Threat Detection	The core process of the AI engine analyzing video to identify suspicious activities like theft, fire, or violence
Motion-Based Recording	The system’s storage-saving feature, where video is only recorded when motion or a suspicious event is detected
JWT	JSON Web Token used for secure session management and authentication
2FA	Two-Factor Authentication, a mandatory secondary security layer

2. Overall System Description

2.1. Project Background

Small and medium-scale retailers face severe security threats including theft, looting, fire, and violence. Existing conventional CCTV systems are passive, costly, and ineffective for real-time prevention as they generate massive volumes of unfiltered data. The Smart Lens project fills this market gap by providing an affordable, intelligent, and proactive AI-powered surveillance solution.

2.2. Project Objectives

- **Automate Monitoring:** Automate the manual and inefficient process of monitoring CCTV footage.
- **Instant Detection:** Provide instant detection of security threats (theft, fire, gun, knife, fighting) to enable prompt intervention.
- **Reduce Storage Costs:** Implement motion-based recording that only saves footage when activity is detected.
- **Empower Shopkeepers:** Empower users with instant, actionable alerts and video evidence via a mobile application.
- **Enhanced Safety:** Offer an option to send immediate alerts to local law enforcement or community groups.

3. System Architecture

The Smart Lens system employs a **Modular, Multi-tiered Vertical Architecture** designed for real-time processing and high security.

3.1. Architectural Components and Tiers

- **Layer I: Data Acquisition Layer (Input):**
  - **IP Cameras:** Source of live video feeds communicating via the RTSP protocol.

- **Layer II: Application Tier (FastAPI Server):**
  - **AI & Motion Pipeline:** Employs OpenCV for motion filtering and a YOLO/CNN Threat Classifier for event labeling.
  - **Core Logic Services:** Manages the **Session Manager (JWT)** for secure access, **Smart Storage Manager** for file handling, and the **Notification Dispatcher**.

- **Layer III: External Cloud Services:**
  - **Backblaze B2:** Secured cloud storage used exclusively for evidence-grade threat clips.
  - **Firebase (FCM):** Manages instant push notification delivery to mobile clients.
  - **SMTP Email API:** Facilitates the automated forwarding of evidence links to emergency contacts.

- **Layer IV: Persistence Layer:**
  - **Supabase (PostgreSQL):** Securely manages Alert Metadata, User & Session Data, and Audit Logs.
  - **Local HDD:** Stores high-volume motion-triggered recordings locally to minimize cloud overhead.

- **Layer V: Client Tier:**
  - **Mobile Application:** A Flutter UI for monitoring live feeds, receiving 2FA-secured alerts, and managing cameras.

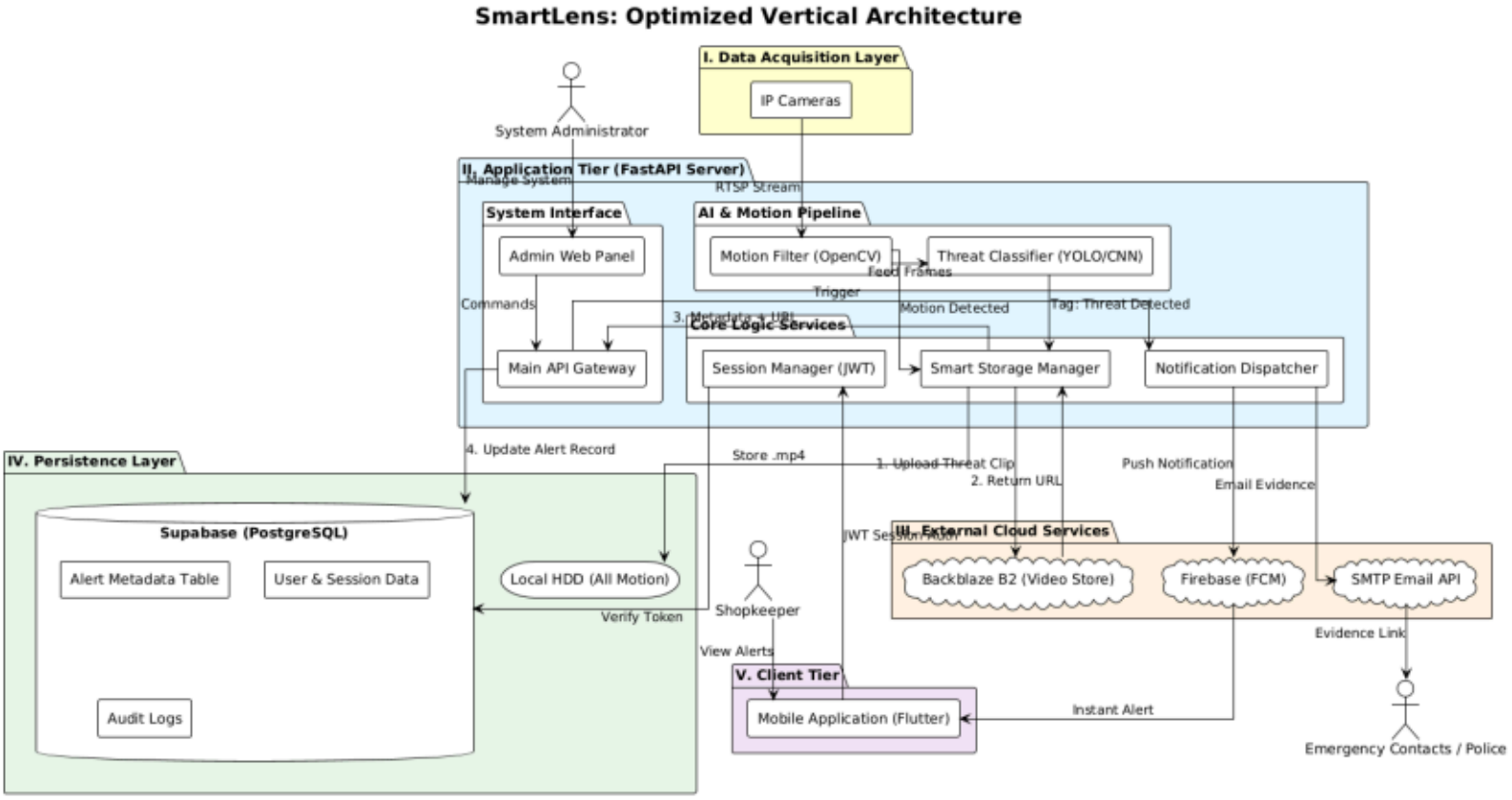


Figure 3.1—SmartLens : Multi-tier Vertical architecture

4. Domain Model

The Domain Model defines the conceptual framework, illustrating key entities and their structural relationships.

4.1. Domain Entities

Entity	Description	Key Attributes
User (Shopkeeper)	Primary stakeholder responsible for camera management and receiving alerts.	userID (PK), email, passwordHash, is_2fa_enabled
Session	Manages secure interaction periods via JWT.	sessionID (PK), jwtToken, expiry
Camera	Represents a physical IP camera registered within the system.	cameraID (PK), userID (FK), streamURL (RTSP), status
AI Model	Represents a specific version of the detection engine managed by administrators.	modelID (PK), version, lastTrained
Alert (Event Log)	A record of a detected and classified suspicious event.	alertID (PK), cameraID (FK), eventType, confidenceScore
Video Clip	The media file recorded upon motion/suspicion detection.	clipID (PK), alertID (FK), localPath, cloudURL
Audit Log	Security record tracking system maintenance and monitoring activities.	logID (PK), activityType, timestamp

4.2. Conceptual Relationships

- **User ⟵ maintains ⟶ Session:** A security-focused relationship ensuring every user

action is authenticated via JWT.

- **User ⟵ owns ⟶ Camera:** A one-to-many relationship where a shopkeeper manages multiple streams.
- **Camera ⟵ triggers ⟶ Alert:** A one-to-many relationship where detection events are logged per camera.
- **Alert ⟵ records ⟶ Video Clip:** A one-to-one relationship mapping event metadata to physical storage.

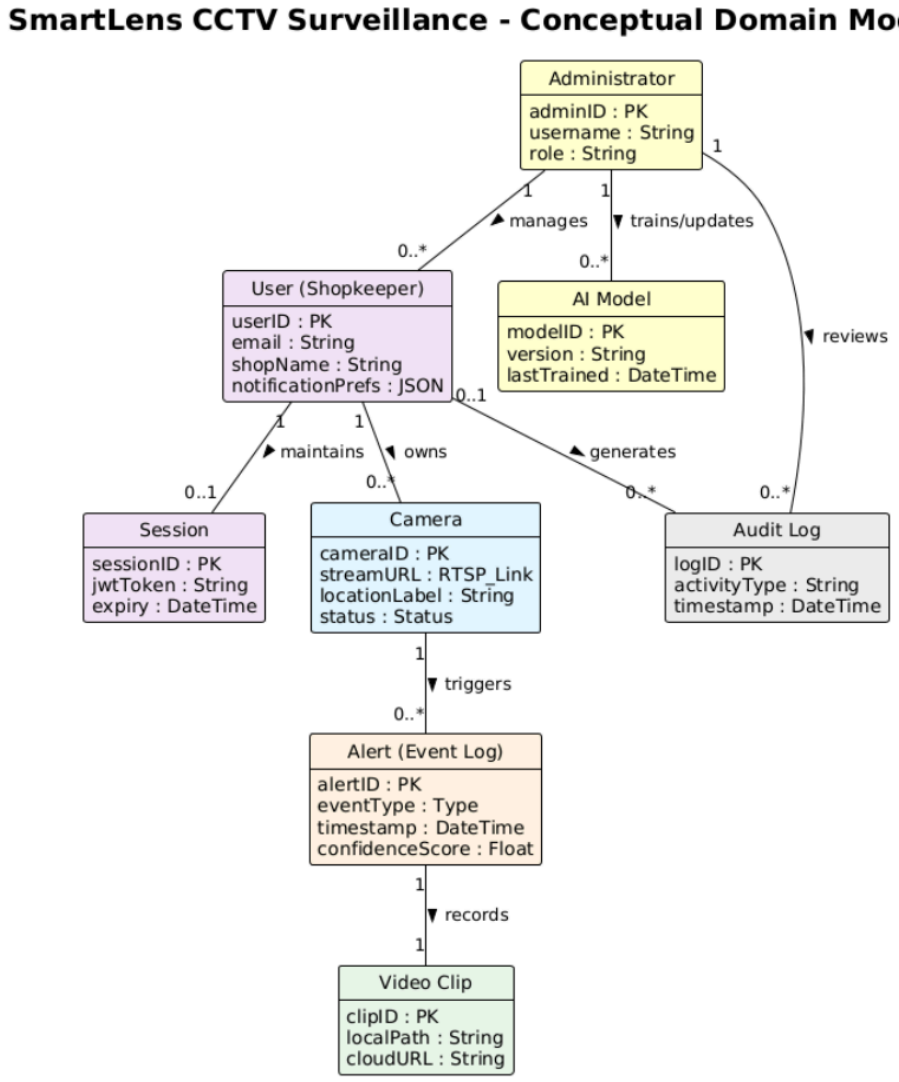


Figure 4.2—Conceptual Domain Model

5. Class Diagram

The Class Diagram details the structural design of the key software components, mapping the domain entities to implementable classes and their corresponding methods.

5.1 Textual Description of Key Classes

The following classes represent the unified implementation layer of the Smart Lens system:

Class Name	Key Attributes	Key Methods (Derived from FR)
UserAccount	userID, email, passwordHash	register() (FR-1.1), login() (FR-1.2), updateProfile() (FR-1.4), manageNotificationPrefs() (FR-1.6)
CameraManager	cameraID, streamURL, locationLabel	addCamera() (FR-2.1),editMetadata() (FR-2.2), removeCamera() (FR-2.3), validateConnection() (FR-2.4)
AI_Engine	modelVersion, confidenceThresholds	analyzeFeed(stream) (FR-3.2), detectSuspiciousActivity() (FR-3.3), updateModel(file) (FR-6.3)
Storage_Manager	storageLimit, currentUsage	recordEventClip() (FR-5.1), deleteClip() (FR-5.3), manageStorage() (FR-5.5)
Alert_System	alertID, eventType, clipURL	generateAlert() (FR-4.1), viewClip() (FR-4.3), forwardAlert(recipient) (FR-4.4)
Mobile_App_UI	currentCameraStream, notificationStatus	displayLiveFeed() (FR-3.1), displayAlert(), searchEvents(filter) (FR-9.1)

5.2 Key Class Relationships

The updated class relationships emphasize a secure, top-down flow across the system layers as illustrated in the unified implementation design:

- **Identity to Services:** The UserAccount authenticates via the SessionManager to receive a JWT token, which is required for all subsequent secure interactions and session establishment.
- **User to Hardware:** UserAccount maintains a direct 1-to-many relationship with the CameraManager, enabling shopkeepers to maintain centralized control over multiple registered devices.
- **AI Orchestration:** The AI_Engine depends on the CameraManager to provide the necessary RTSP stream details; it subsequently triggers both the Storage_Manager for motion-based recording and the BackendAPI_Gateway for metadata logging upon threat detection.
- **Hybrid Storage Flow:** The Storage_Manager provides secure cloudURL links to the BackendAPI_Gateway only after threat-detected clips are successfully mirrored from local buffers to cloud storage.
- **Alert Dispatch:** The BackendAPI_Gateway initiates the alert sequence through the NotificationDispatcher, which utilizes Firebase Cloud Messaging (FCM) to push instant notifications to the Mobile_App_UI.
- **Administrative Control:** The AdministratorAccount utilizes a dependency relationship with the AI_Engine to facilitate system maintenance, push updated model weights, and monitor performance metrics

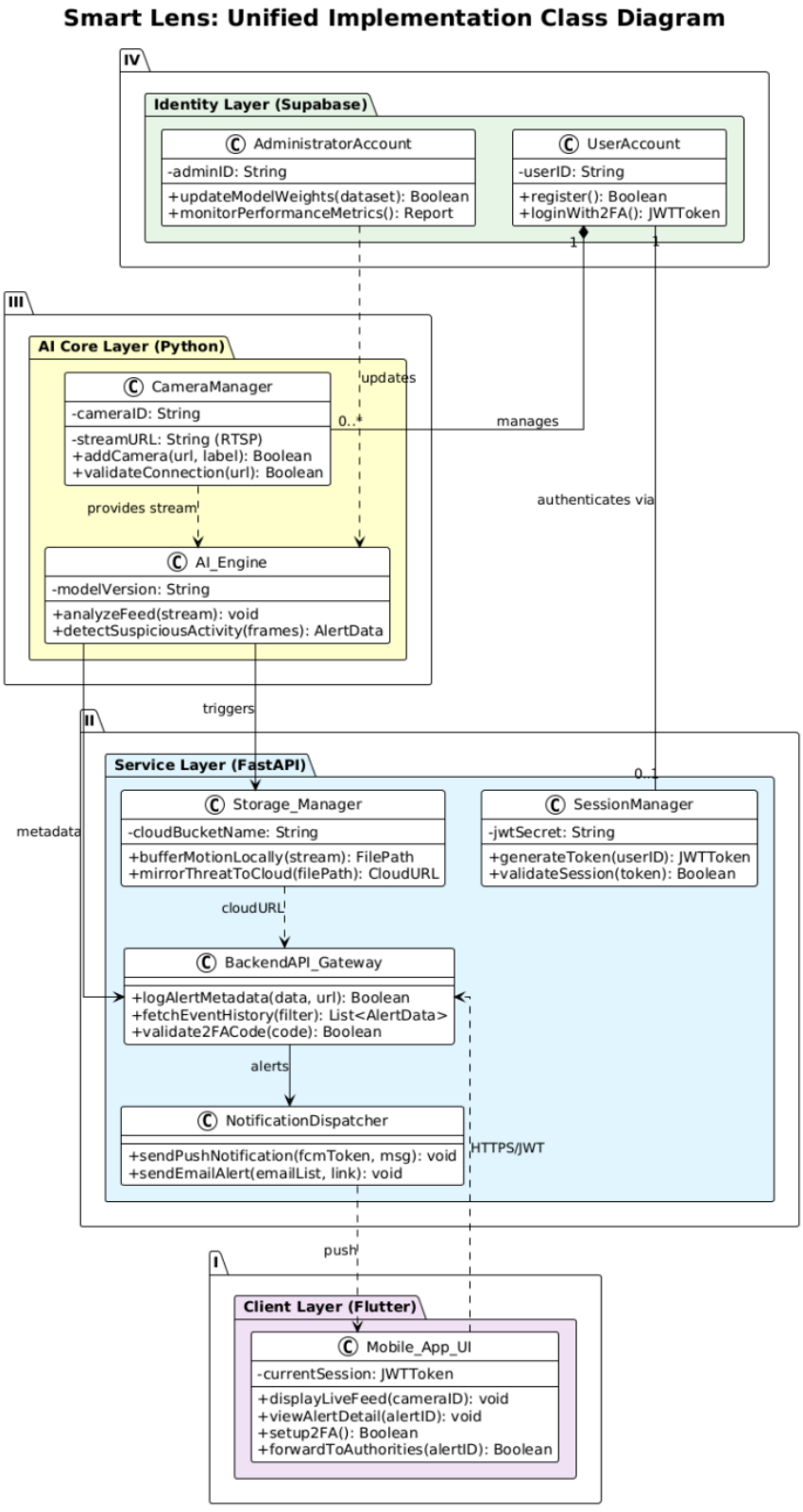


Figure 5.1—Class Diagram

6. Database Diagram & 7. Entity Relationship Diagram (ERD)

The system utilizes a relational schema to ensure data integrity and the fulfillment of storage optimization goals.

6.1. Database Schema (Logical ERD)

Table Name	Attributes (Data Type & Constraints)	Relationships and Constraints
Users	user_id (UUID PK), email (Unique), password_hash, is_2fa_enabled	Base identity table for shopkeepers.
Sessions	session_id (PK), user_id (FK), jwt_token, expires_at	**FK:** user_id → Users. Manages secure sessions.
Cameras	camera_id (PK), user_id (FK), rtsp_url, status	**FK:** user_id → Users (ON DELETE CASCADE).
Alerts	alert_id (PK), camera_id (FK), event_type, is_threat	**FK:** camera_id → Cameras. Logs all detections.
Video_Clips	clip_id (PK), alert_id (FK), local_file_path, cloud_url	**FK:** alert_id → Alerts. Stores hybrid paths.
Audit_Logs	log_id (PK), user_id (FK, Nullable), activity_type	**FK:** user_id → Users. Security event tracking.

6.2. Key Design Decisions

- **Advanced Security:** Passwords and sensitive data are stored as salted and hashed values to prevent unauthorized access. The inclusion of a Sessions table ensures that all API requests are validated against an active JWT, fulfilling the committee's requirement for robust session management.

- **Two-Factor Authentication (2FA):** The Users table now includes an `is_2fa_enabled` flag, supporting the mandatory secondary security layer during the login process.

- **Hybrid Storage Implementation:** Instead of a single `clip_url`, the Video_Clips table maintains both a `local_file_path` for high-volume motion data and a `cloud_url` for threat-specific evidence mirrored to Backblaze B2. This optimizes cloud costs while ensuring evidence availability.

    - **Governance and Auditability:** The Audit_Logs and AI_Models tables provide a transparent record of system modifications, ensuring that administrative actions (like model retraining) and user actions (like deleting clips) are fully traceable.

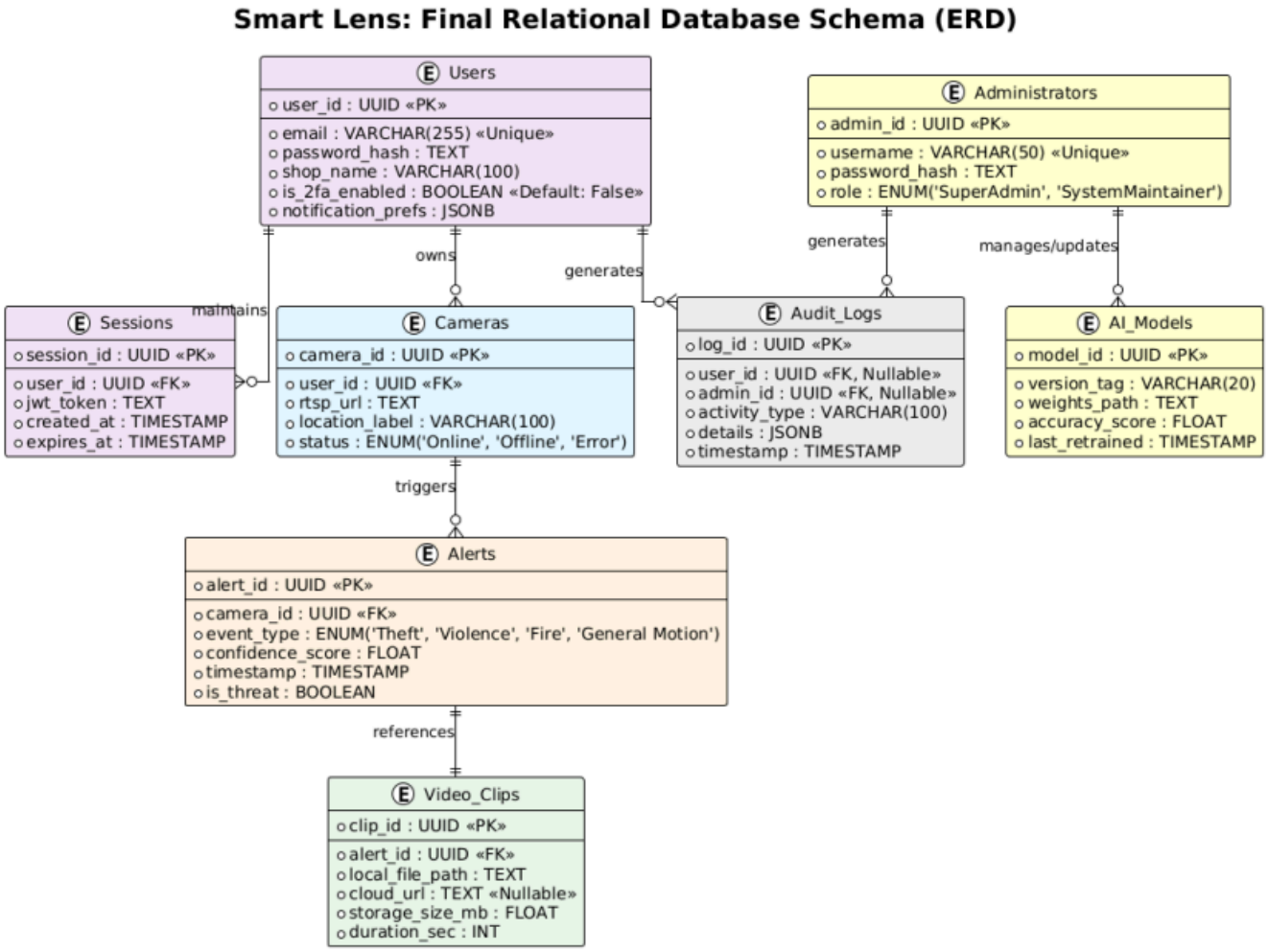


Figure 6.1—Entity Relationship Diagram

8. Sequence Diagrams

To ensure a comprehensive technical defense, the Smart Lens system provides multiple sequence diagrams covering core surveillance logic, administrative maintenance, and mandatory security protocols. These diagrams illustrate how data flows between the actors and the layered architectural components.

8.1. Threat Detection & Alert Flow

This flow details the real-time AI pipeline and the hybrid storage logic used to minimize data costs by separating motion-triggered local recording from threat-triggered cloud uploads.

- **IP Camera ⟶ AI Engine:** Sends the live RTSP stream for continuous frame analysis (FR-3.2).
- **AI Engine ⟶ AI Engine:** Processes frames via YOLO/CNN to detect motion; once detected, it triggers the **Storage Manager** to begin the local recording buffer (FR-6.4).
- **AI Engine ⟶ AI Engine:** Simultaneously calculates a Confidence Score to classify suspicious activity (Theft, Fire, etc.).
- **AI Engine ⟶ Storage Manager:** If the score exceeds the predefined threshold (Confidence > Threshold), the AI Engine triggers a "Threat Detected" event (FR-6.5).
- **Storage Manager ⟶ Storage Manager:** Buffers the motion-based clip to the local HDD to optimize storage consumption (FR-5.1)
- **Storage Manager ⟶ Backend Gateway:** Syncs metadata and mirrors only threat-detected clips to cloud storage (Backblaze B2), providing a secure Cloud URL.
- **Backend Gateway ⟶ Mobile App:** Dispatches a push notification via Firebase (FCM) to notify the shopkeeper instantly (FR-4.1, NFR-1.1).
- **Mobile App ⟶ Backend Gateway:** Validates the user session (JWT) before requesting the secured evidence link (FR-7.2).
- **Backend Gateway ⟶ Mobile App:** Provides the evidence link and streams the video to the client for secure playback.

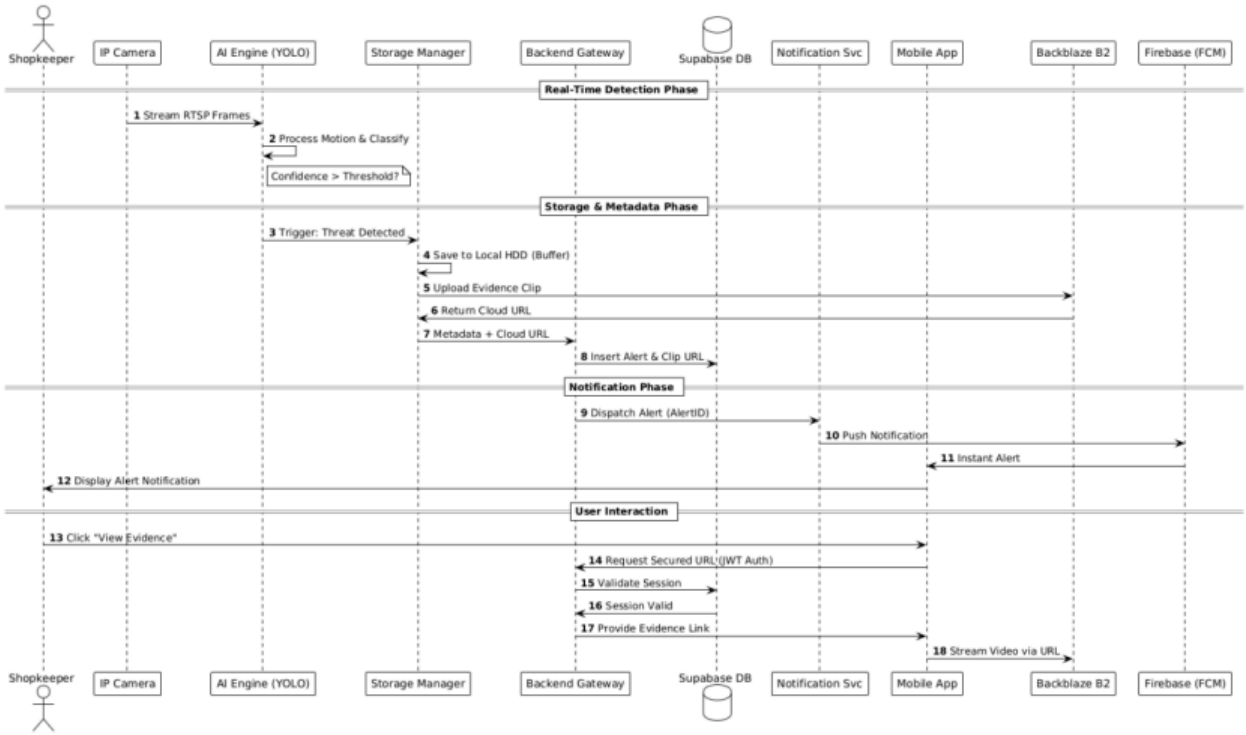


Figure 8.1—Suspicious Activity Detection and Instant Alert Generation

8.2. Two-Factor Authentication (2FA) & Session Flow

Per committee requirements, this diagram illustrates the detailed secondary authentication process and session establishment logic.

- **User → Mobile App:** The Shopkeeper enters primary email and password credentials.
- **Backend Gateway → Database:** Validates credentials; if successful, the system identifies that 2FA is required for this user account.
- **Backend Gateway → External API:** Generates and dispatches a unique 6-digit OTP code via Email or SMS.
- **User → Mobile App:** The user receives the OTP and enters the code into the verification interface.
- **Backend Gateway → Session Manager:** Validates the OTP code and requests the generation of a new JWT session.
- **Session Manager → Mobile App:** Stores the session ID and expiry in the database and

returns the JWT token to the app for future authenticated requests.

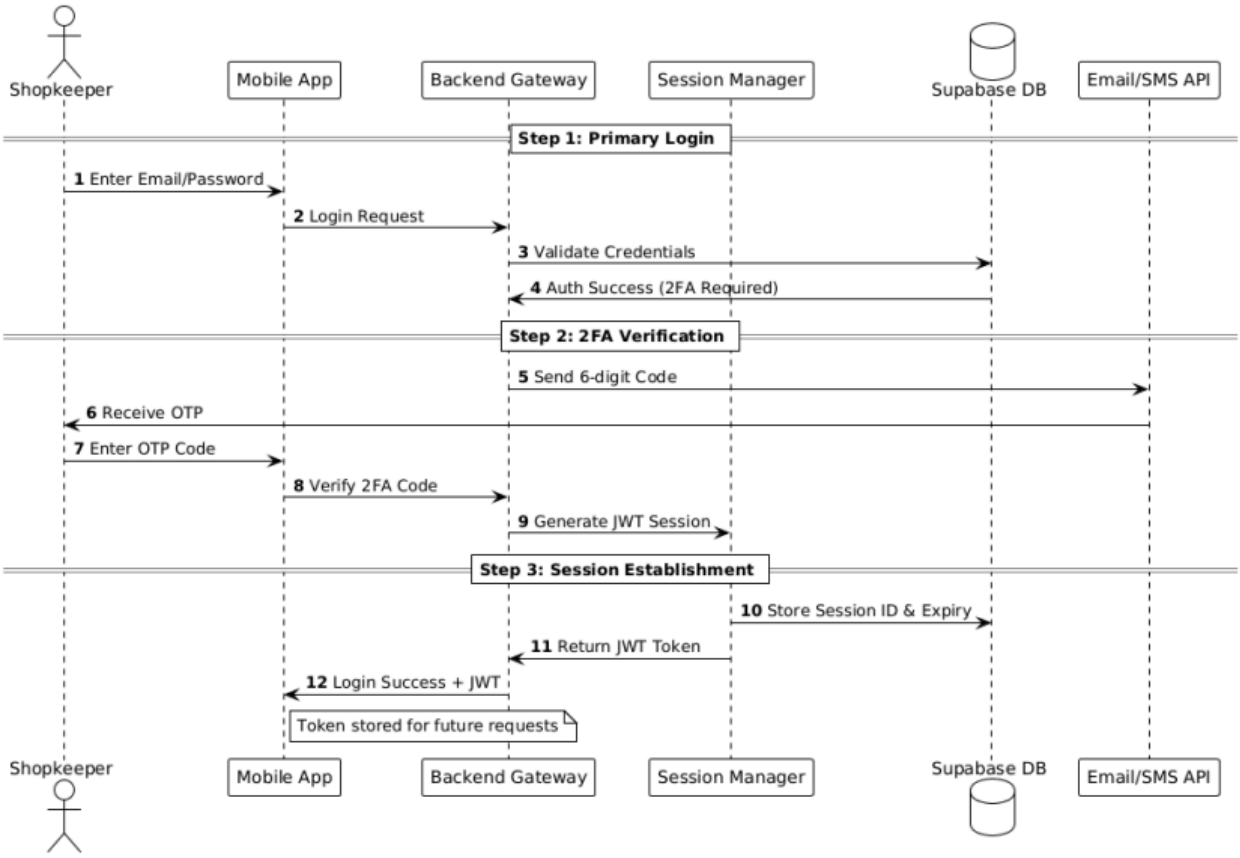


Figure 8.2—2FA & Session Management Sequence Diagram

8.3. Administrator AI Model Update Flow

This flow illustrates the system maintenance lifecycle, highlighting how administrators manage AI versioning and retraining.

- **Administrator → Admin Dashboard:** The System Administrator logs in to the web panel to access system health and performance metrics.
- **Backend Gateway → AI Engine:** Following a request for an update, the Gateway dispatches a trigger to the AI Engine to reload model weights or initiate retraining (FR-8.5).
- **AI Engine → Model Repository:** The Engine pulls the latest verified weight file (.pt or .weights) from the repository.

- **AI Engine → Backend Gateway:** The Engine runs validation tests and confirms a successful update with a new Version ID.

- **Backend Gateway → Database:** Updates model metadata (Version, Accuracy) and creates a mandatory entry in the **Audit Log** for security tracking.

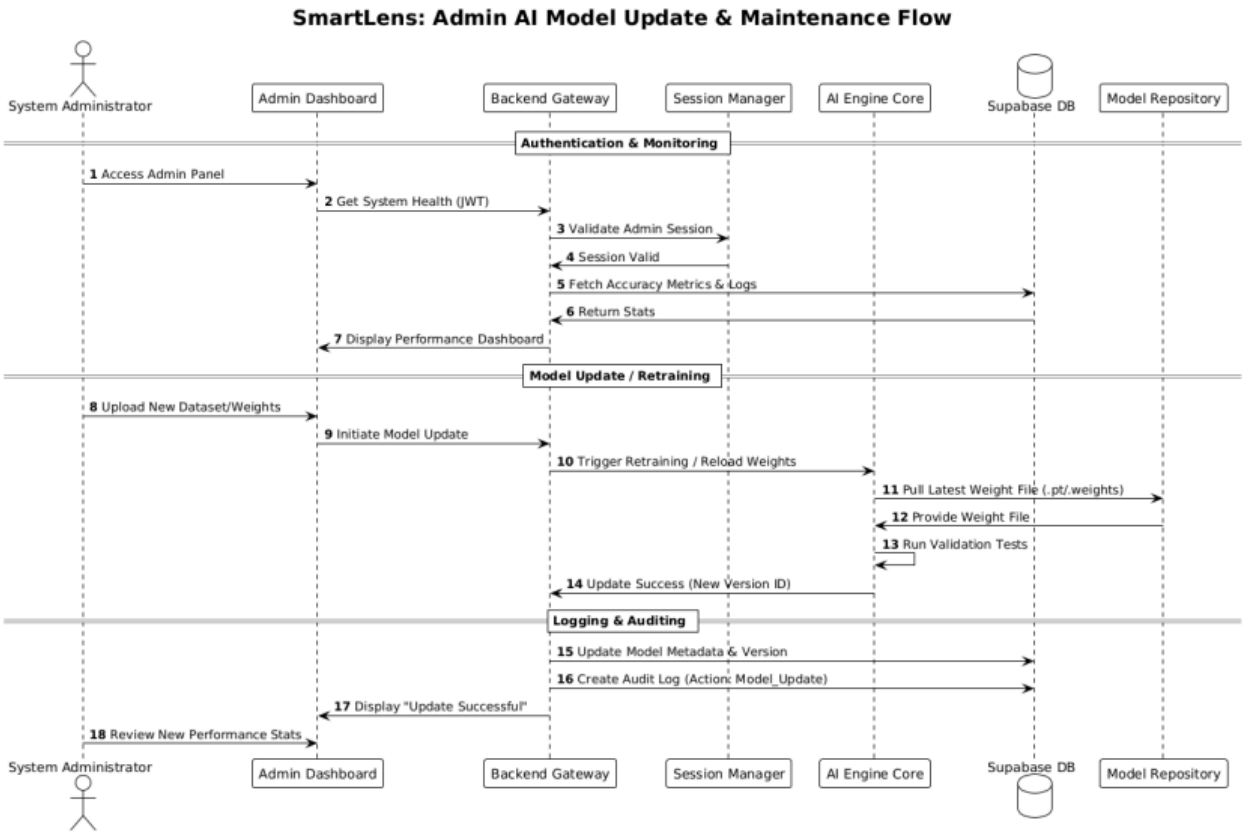


Figure 8.3—Administrator AI Model Update Sequence

9. System Interface Design

The interface design focuses on the primary user, the **Shopkeeper**, emphasizing usability (NFR-5.1) , clarity, and immediate actionability (NFR-5.2). This section has been updated to align with the **Modular, Multi-tiered Vertical Architecture** , ensuring that mandatory security states—such as session validation and Two-Factor Authentication (2FA)—are represented in the interface logic.

9.1. Mobile Application Interface (Main Screens)

The following table details the primary screens within the Flutter-based mobile application and the specific requirements they fulfill:

Screen/Interface	Functionality	Key Requirements Met (FR/NFR)
Authentication & 2FA	Mandatory gateway for secure access. It validates primary credentials (email/password) followed by a 6-digit OTP verification.	FR-1.2, FR-7.4, NFR-2.1
Dashboard (Live Monitoring)	Provides a tiled view of Multi-Camera Live Feeds with low-latency streaming and a prominent "Alerts" badge for navigation to recent events.	FR-3.1, NFR-10.3, NFR-1.2
Alert History & Review	Displays a chronological, filterable log of all detected events, showing Event Type, Timestamp, and specific Camera Location.	FR-5.4, FR-4.2
Event Detail View	The core action screen that plays the recorded video clip. It provides prominent buttons to "Forward Alert" to third parties or "Delete Clip".	FR-4.3, FR-4.4, FR-5.3

Camera Management	An intuitive setup interface for registering new cameras via RTSP URL and managing existing device metadata.	FR-2.1, FR-2.3
Alert Recipient Management	Interface for managing the list of up to 5 trusted contacts who receive automated notifications upon user consent.	FR-4.4, Obj-5

9.2. Notification Interface

The notification system is designed to provide maximum information with minimum user cognitive load.

Element	Description	Key Requirements Met (FR/NFR)
Push Notification	Concise and actionable alert titled "Smart Lens Security Alert!". The body includes the specific Event Type, Location, and Timestamp.	NFR-1.1, NFR-5.2
Notification Action	Tapping the notification bypasses the main dashboard and transitions the state machine directly to the **Event Detail View**.	NFR-5.2

9.3. Interface Design Flow Logic

The mobile interface is governed by a secure state-machine logic to ensure data protection and low-latency response.

- **Session Validation:** Upon application launch, the system automatically enters a **CheckSession** state to verify the existing JWT token stored locally. If the token is valid, the user is granted immediate access to the Dashboard.
- **2FA Gateway:** If no valid session exists, the user is routed through **PrimaryAuth** and a mandatory **Verify2FA** state. Access to live feeds is strictly prohibited until the 6-digit

OTP is validated by the Backend Gateway.

- **Contextual Redirection:** To minimize response time during a security event, tapping an instant notification triggers a direct transition to the **EventDetail** view, bypassing intermediate navigation steps.
- **Forwarding Action:** Within the EventDetail state, users can transition to the **ForwardAlert** state, which utilizes the **Notification Dispatcher** to send evidence links to selected emergency contacts.

The final visual implementation of this design is presented in the high-fidelity mockups located in **Appendix A: Mobile Application UI Mockups**.

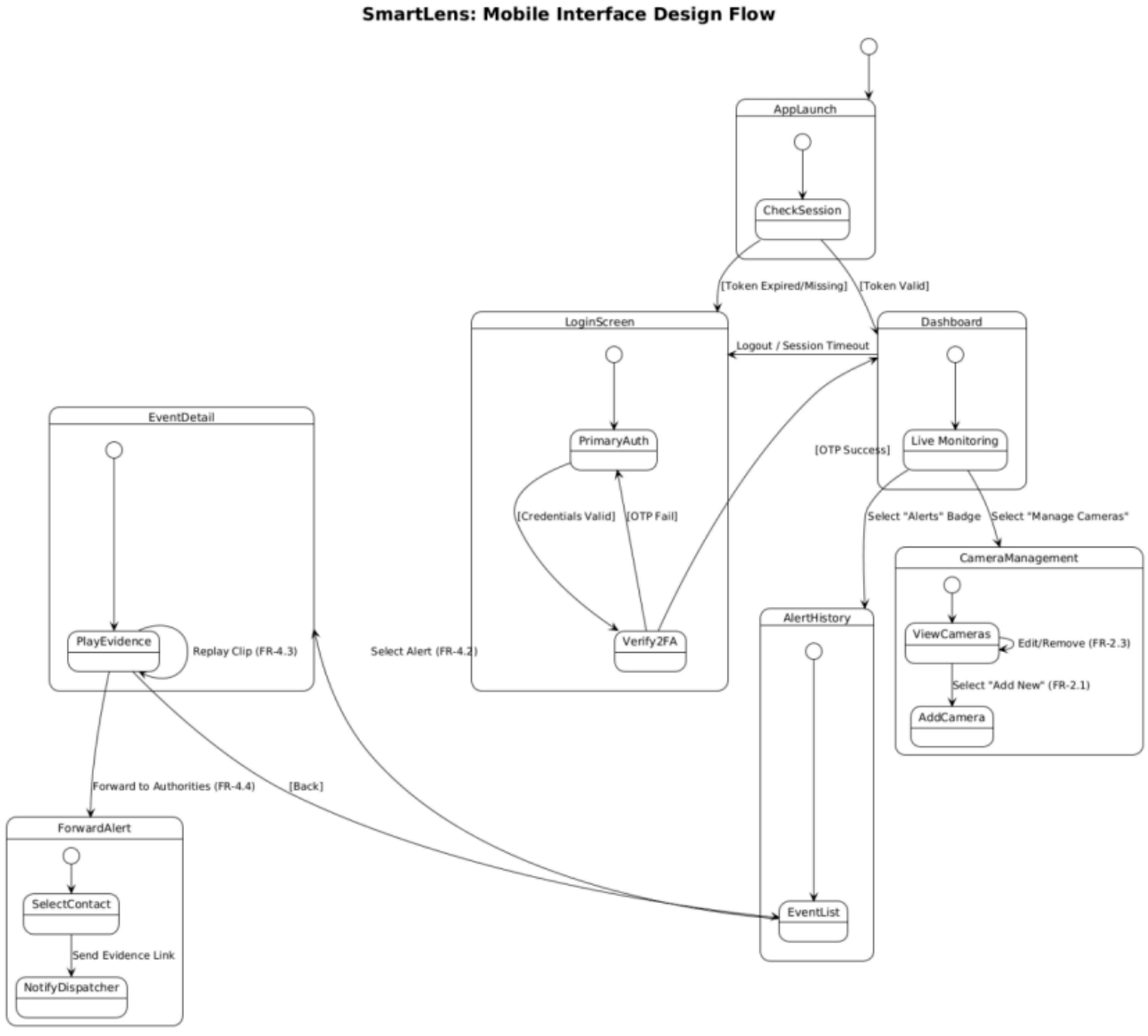
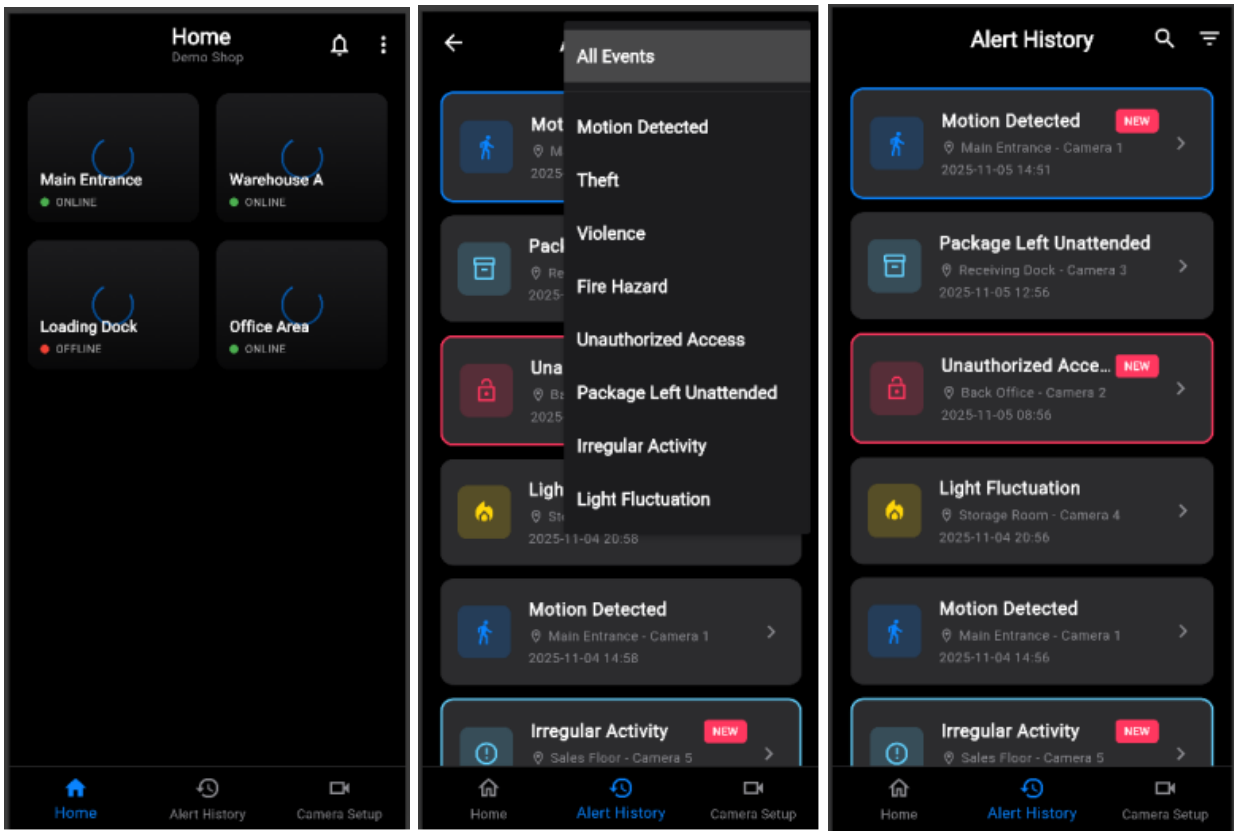


Figure 9.1—Mobile Interface Design

10. Test Cases

Test Case ID	Requirement	Scenario	Expected Result
TC-NFR-1.1	FR-4.1, NFR-1.1	Measure time from hazard simulation to push notification.	Received within 30 seconds.
TC-NFR-4.1	FR-6.5, NFR-4.1	Monitor normal activity for 2 hours to check for false alerts.	Zero alerts generated.
TC-FR-5.1	FR-5.1	Run camera on static, empty scene for 1 hour.	No clips saved.
TC-FR-7.4	FR-7.4, NFR-2.2	Inspect API calls between app and server for encryption.	All traffic uses HTTPS.
TC-FR-3.3.1	FR-3.3.1	Simulate a theft and verify AI classification.	Correct tag and high confidence.
TC-FR-2.4	FR-2.4	Attempt to add an invalid RTSP URL.	Error displayed; no DB record.

11. Appendix A: UI Mockups



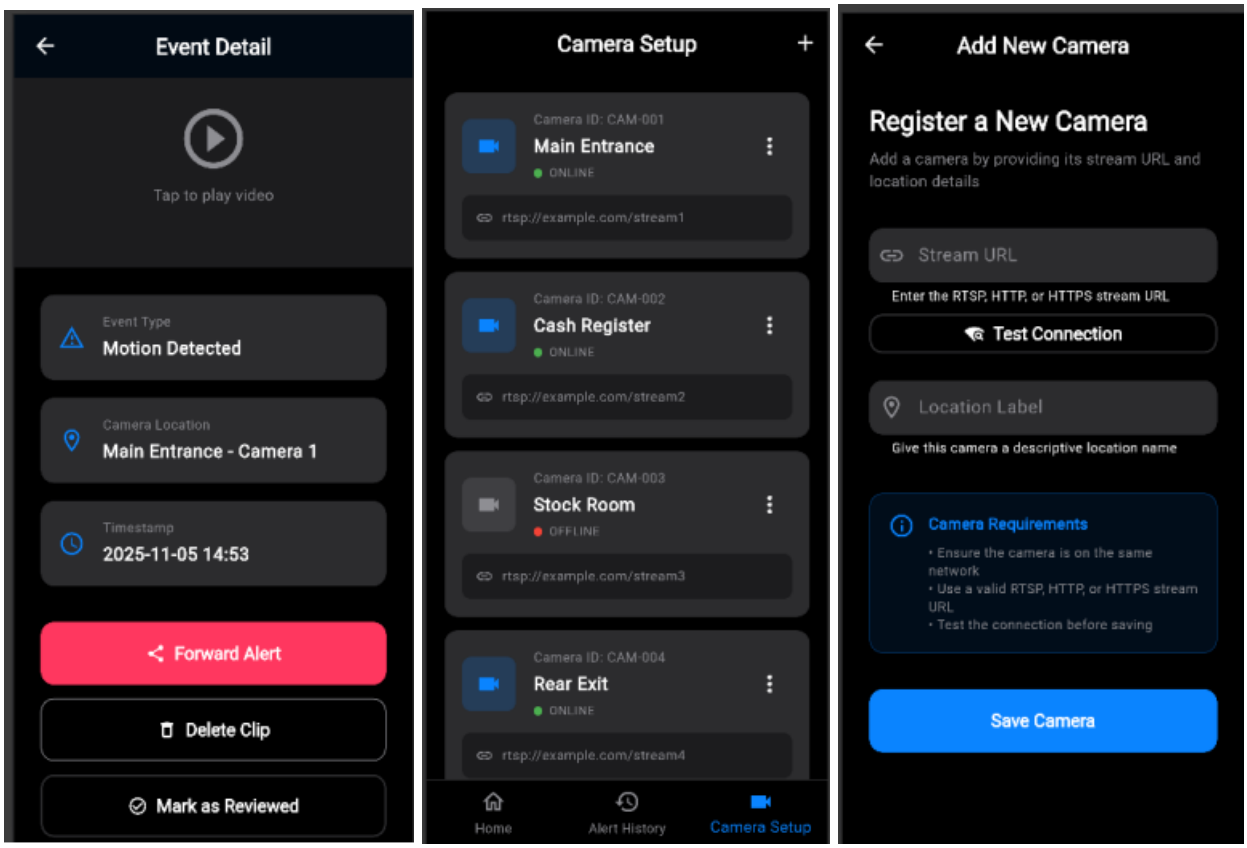


Figure 11.1—Smart Lens Mobile App UI Mockups