AegisSecure: AI-Powered Threat Detection App

Software Design Document (SDD)

Group Number: 35
Mevada Soham Meghalkumar [202301484]
Gohil Suryadeepsinh Hardevsinh [202301463]
Rana Neelabh Vijaykumar [202301476]
Hrithik B Patel [202301441]
Vadsmiya Pransu Pradipkumar [202301445]
Dhruv Jigneshkumar Patel [202301095]
Bhagiya Jenish Rameshbhai [202301480]
Akshat Bhatt [202301460]
Vrajkumar Makwana [202301436]
Chavda Mihirsinh Labhubhai [202301479]

November 28, 2025

Figure 1: *
University Logo

IT314 - Software Engineering
Prof: Saurabh Tiwari
Mentor: Shyam Patel

Contents

1 Introduction

1.1 Purpose

The purpose of this Software Design Document (SDD) is to provide a detailed architectural and low-level design overview of the **AegisSecure** application. It describes the system's structure, components, interfaces, and data flow to ensure a clear understanding for developers and stakeholders.

1.2 Scope

This document covers the design of the following key components:

- **Frontend:** The Android mobile application built using Flutter/Kotlin.
- **Backend:** The server-side logic API built with FastAPI.
- **Database:** The data persistence layer using MongoDB.
- **AI Service:** The spam detection models (Ensemble: BERT, Bi-LSTM, Llama 4).
- **External Integrations:** Google OAuth, Gmail API, and Android OS services (Telephony/SMS).

2 Use Case Model

This section illustrates the functional requirements of AegisSecure from the user's perspective, identifying the key actors and their interactions with the system.

2.1 System Actors

- **Mobile User:** The primary actor who uses the app to analyze SMS and emails.
- **System Administrator:** Manages backend services and monitors system health.
- **AI Service:** External ML service that processes messages for threat detection.
- **Gmail API:** External service for email synchronization.
- **Android OS:** Provides SMS reading permissions and telephony services.

2.2 Use Case Diagram

The following diagram shows all major use cases and their relationships.

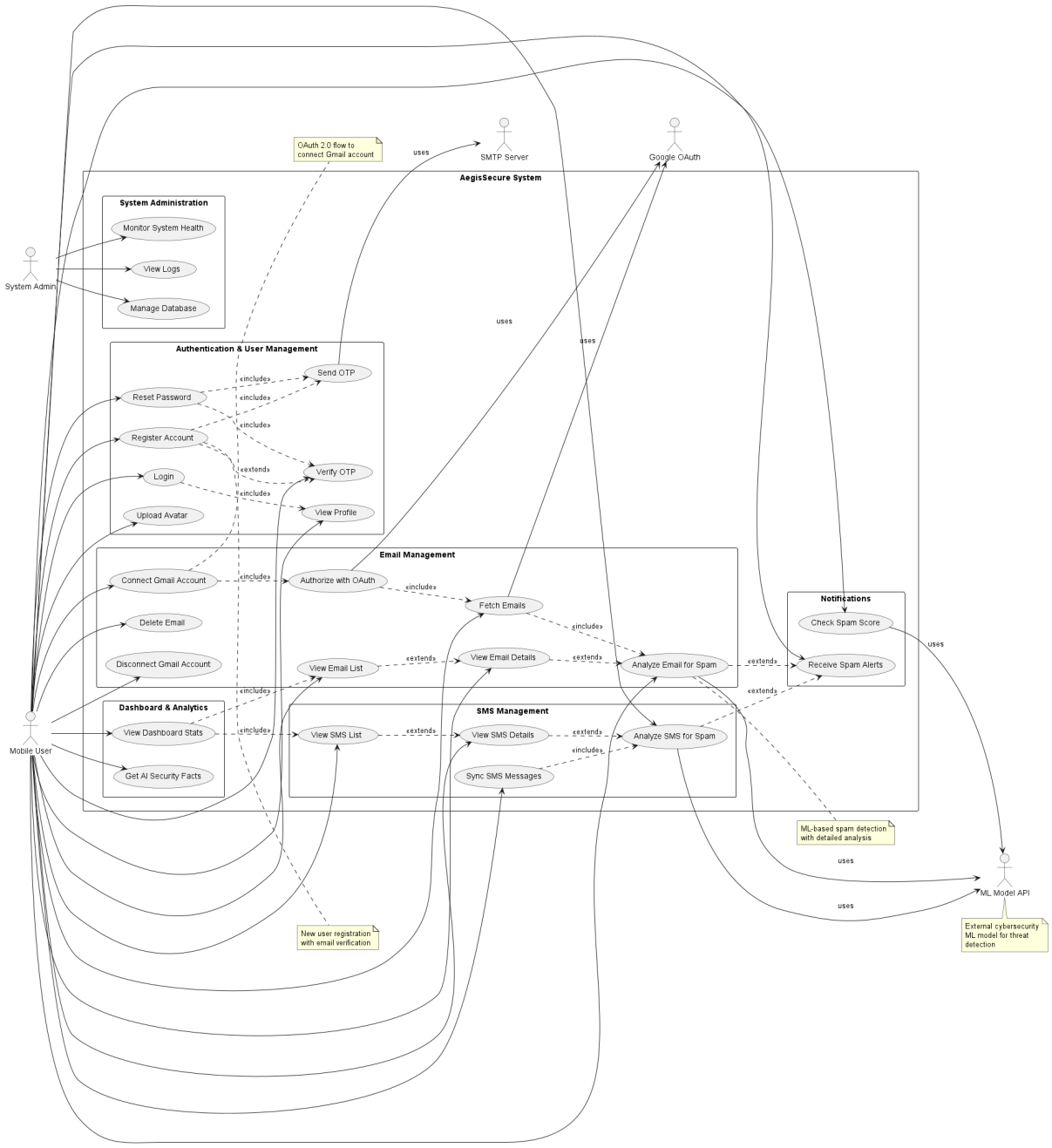

Figure 2: Use Case Diagram: AegisSecure System Use Cases

3 System Architecture

This section outlines the high-level architecture of AegisSecure. The design focuses on handling multiple users simultaneously (concurrency) and ensuring the app stays responsive (non-blocking I/O).

3.1 Overall Architecture

The system follows a Client-Server architecture. The Mobile/Web client sends requests to the FastAPI Backend. The backend handles security and data, while heavy tasks (like AI spam detection) run in the background to avoid delays.

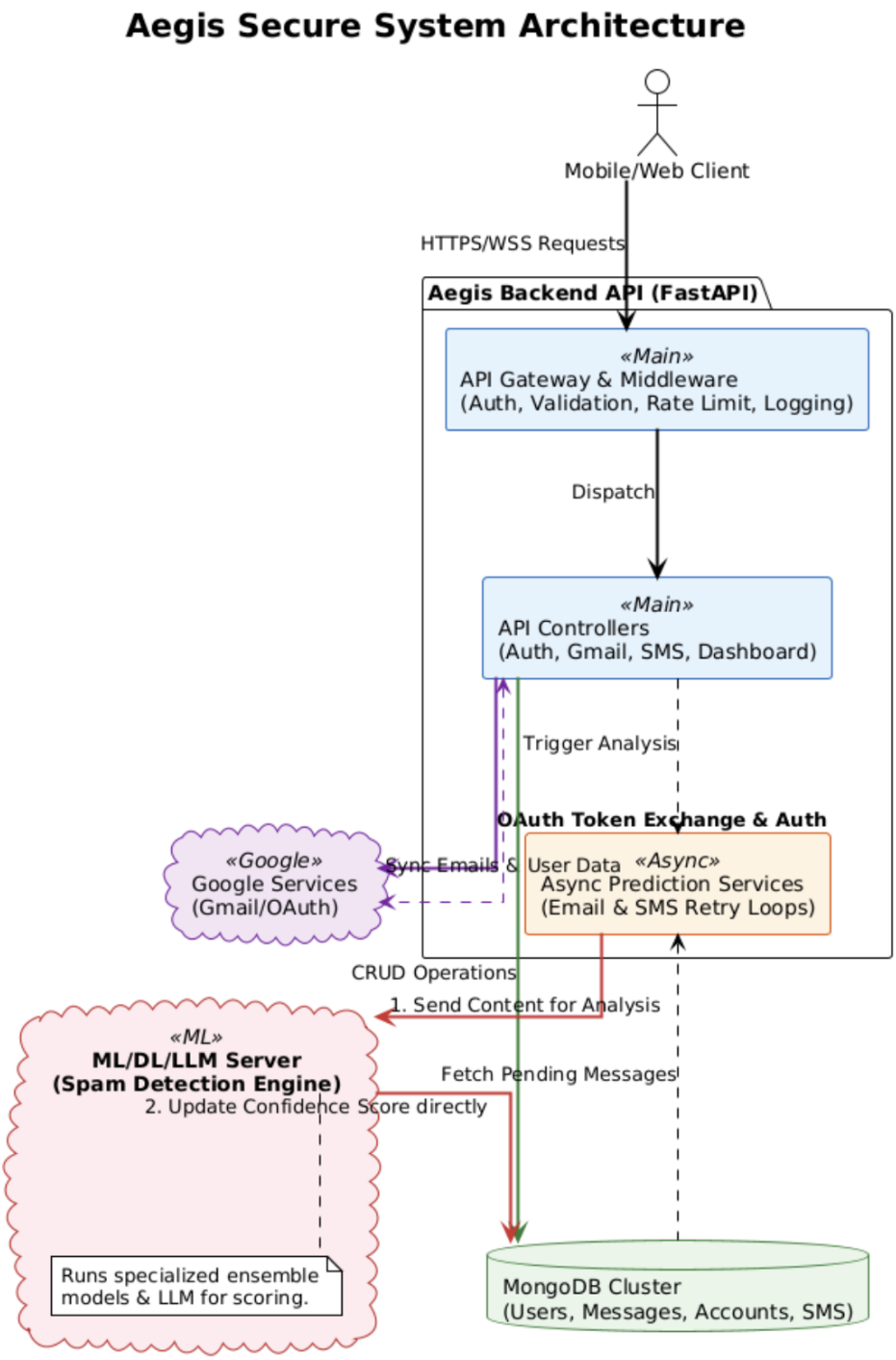

Figure 3: AegisSecure System Architecture

3.2 System Decomposition

The Aegis Backend is divided into specific layers to separate concerns. This structure ensures that security logic, database access, and AI analysis do not interfere with each other.

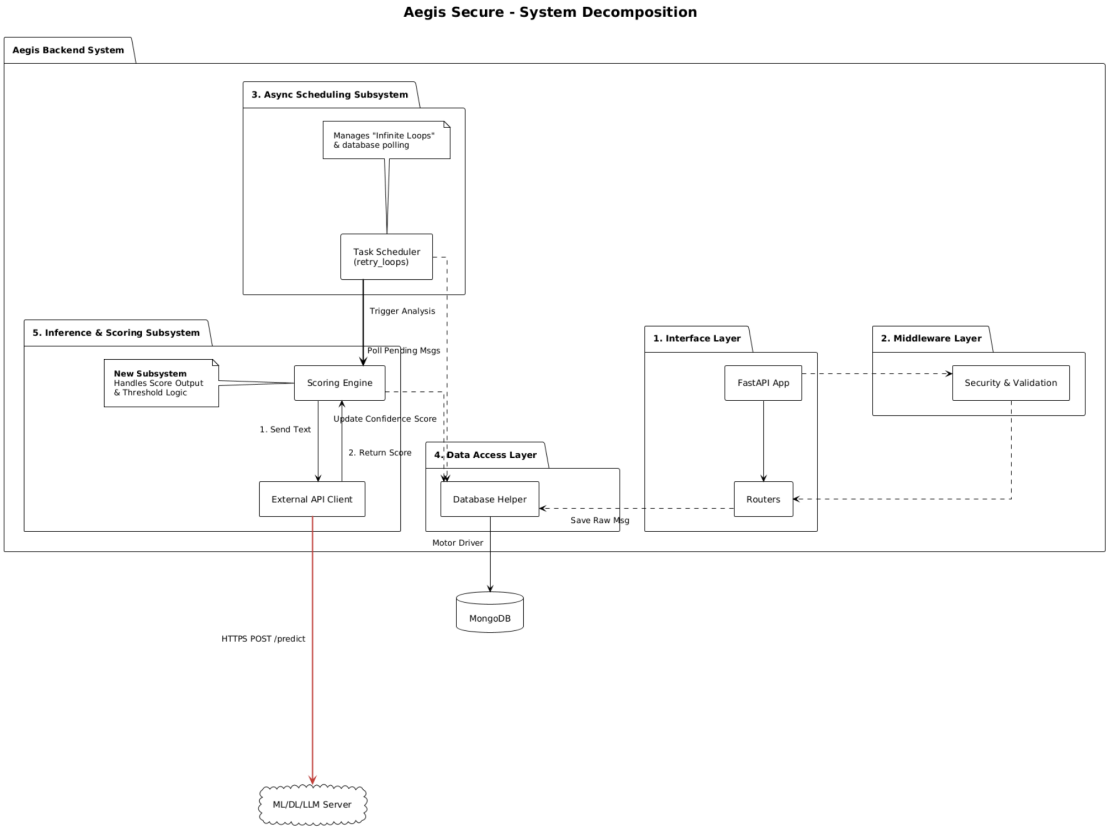

Figure 4: System Decomposition Layers

3.2.1 Subsystem Description

- **1. Interface Layer (Routers):** The entry point for the app. It routes requests (like Login or Sync SMS) to the correct controller.
- **2. Middleware Layer (Security):** Checks every request before it is processed. It handles Rate Limiting (to stop spam attacks) and Validation (to prevent hacking).
- **3. Async Scheduling Subsystem:** This layer manages background tasks. It uses "retry loops" to keep checking for new messages that need analysis.
- **4. Data Access Layer:** Handles all interactions with the MongoDB database. It uses a helper to automatically retry database connections if they fail.
- **5. Inference & Scoring Subsystem:** The "brain" of the system. It sends text to the AI server and calculates a final scam confidence score.

3.3 Component Architecture

The following diagram shows how major components interact and depend on each other within the system.

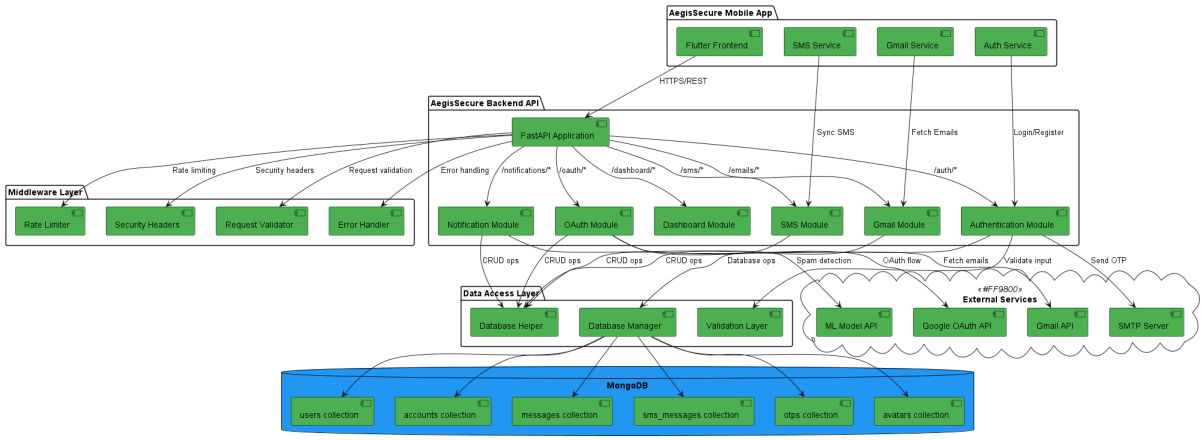

Figure 5: Component Diagram: System Component Interactions

3.4 Package Structure

The package diagram illustrates the logical organization of the codebase into modules and their dependencies.

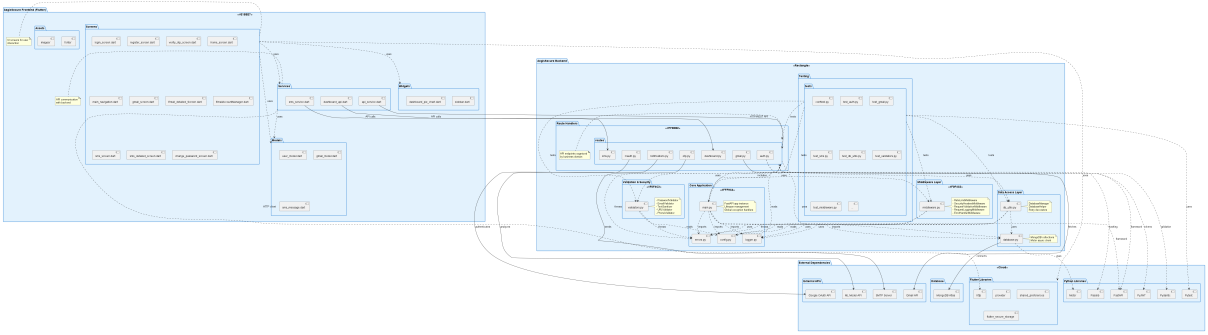

Figure 6: Package Diagram: Module Organization

4 Data Model

This section describes the database schema and data persistence layer for AegisSecure.

4.1 Entity Relationship Diagram

The following ERD shows the MongoDB collections, their fields, relationships, and cardinality across three databases: auth_db, Mails_db, and Sms_db.

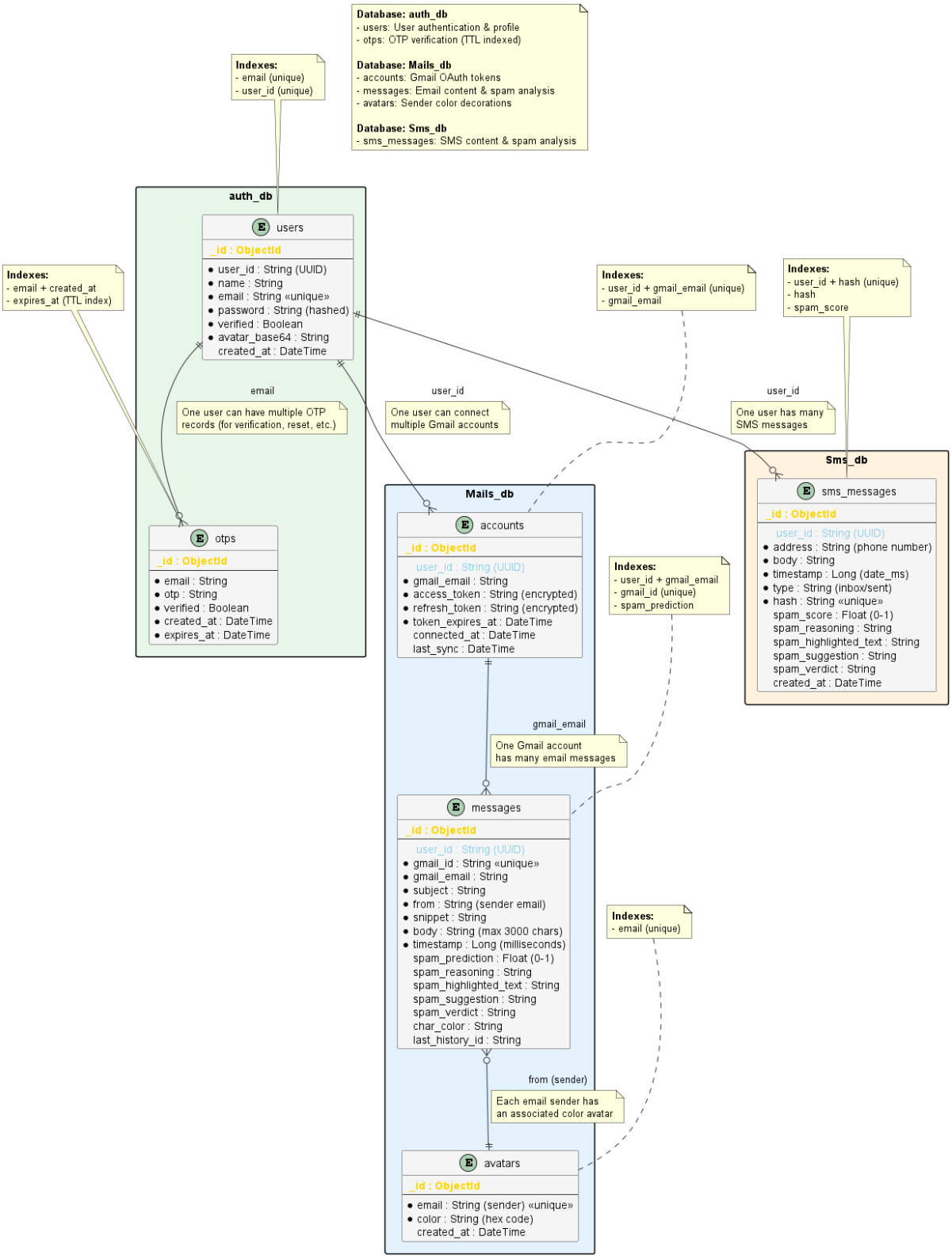

Figure 7: Entity Relationship Diagram: Database Schema

4.2 Database Collections

4.2.1 auth_db Database

- **users:** Stores user authentication credentials, profile information, and verification status
- **otps:** Stores one-time passwords with TTL (Time To Live) indexing for automatic expiration

4.2.2 Mails_db Database

- **accounts:** Stores Gmail OAuth tokens and account connection metadata
- **messages:** Stores email content, spam analysis results, and prediction scores
- **avatars:** Stores color mappings for email sender visual identification

4.2.3 Sms_db Database

- **sms_messages:** Stores SMS content, sender information, and spam analysis results

5 Object Design

In this phase, we bridged the gap between requirements and implementation by identifying two types of objects: Application Domain objects and Solution Domain objects.

5.1 Application Domain Objects

These objects represent real-world concepts relevant to the problem of spam detection.

- **User:** Represents the person using the app (attributes: email, name).
- **Message:** Represents the SMS or Email content being analyzed.
- **SpamReport:** The verdict produced by the system regarding a specific message.

5.2 Solution Domain Objects

These are technical objects created by developers to make the system work. They do not exist in the real world but are necessary for the software solution.

- **DatabaseHelper:** A wrapper class to manage MongoDB connections and retries.
- **RateLimiter:** A logic component that tracks IP addresses to prevent abuse.
- **JWTUtils:** A utility class for handling secure token encryption and decryption.
- **APIService (Frontend):** A client-side object that manages HTTP communication with the backend.

6 Class Diagrams

The following diagrams detail the specific classes, attributes, and methods for the Frontend, Backend, and AI Model components.

6.1 Complete Backend Class Model

Comprehensive class diagram showing all backend system classes, their relationships, attributes, and methods including configuration, error handling, database layer, validation, middleware, data models, and route handlers.

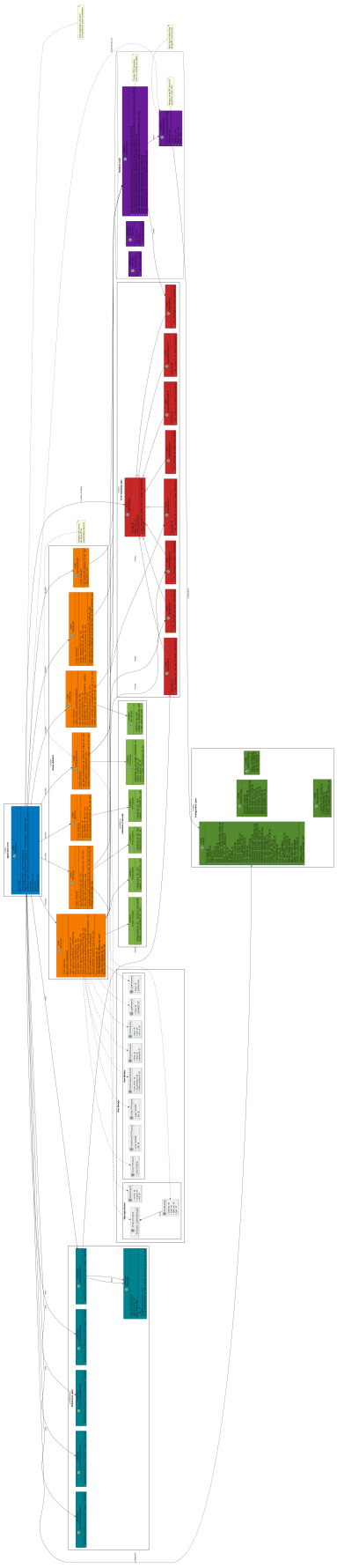

Figure 8: Complete Backend Class Diagram: System-Wide Class Hierarchy

6.2 Frontend Class Hierarchy

Detailed view of the Flutter/Mobile application structure, including screens, services, and models.

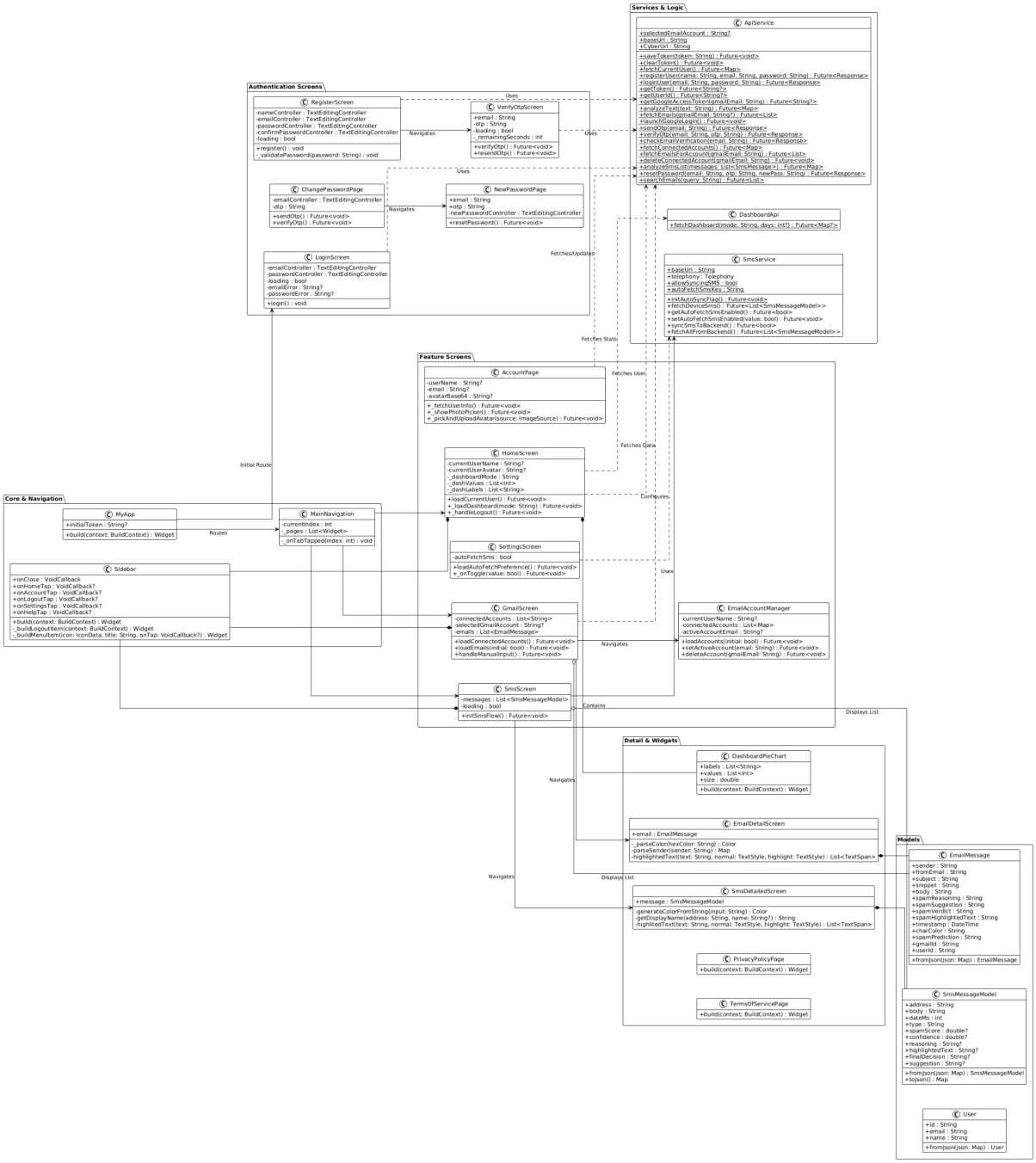

Figure 9: Frontend Class Diagram

6.3 Backend Class Hierarchy

Detailed view of the FastAPI services, Data Transfer Objects (DTOs), and database models.

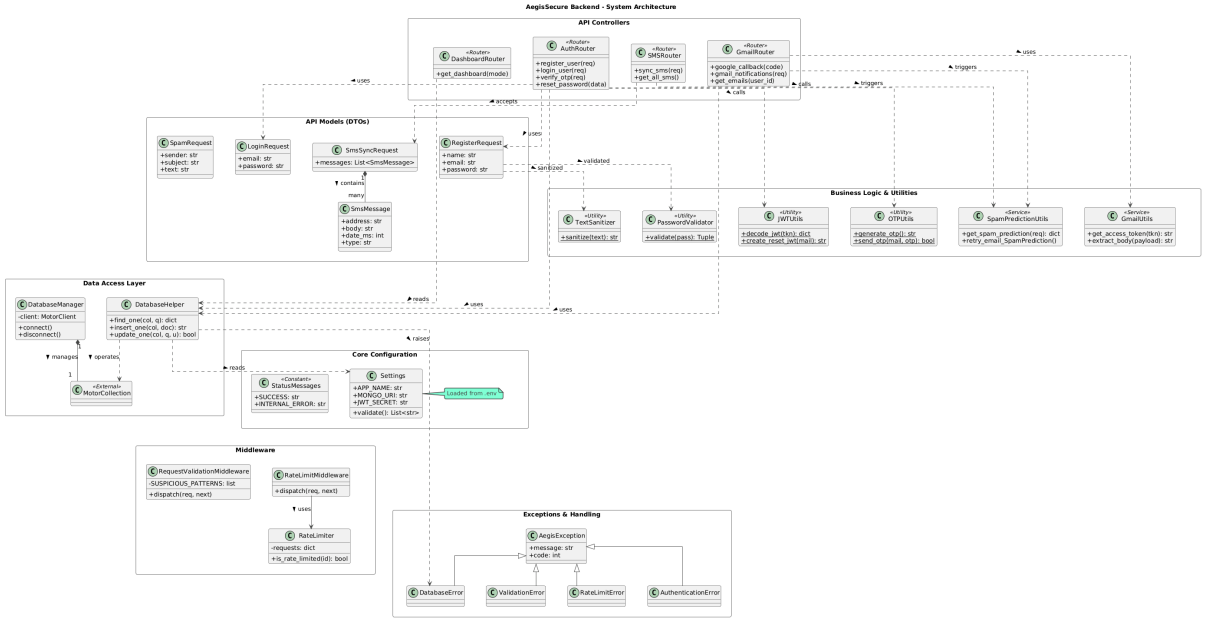

Figure 10: Backend Class Diagram

6.4 AI Model Class Hierarchy

Detailed view of the Machine Learning orchestration and Deep Learning models.

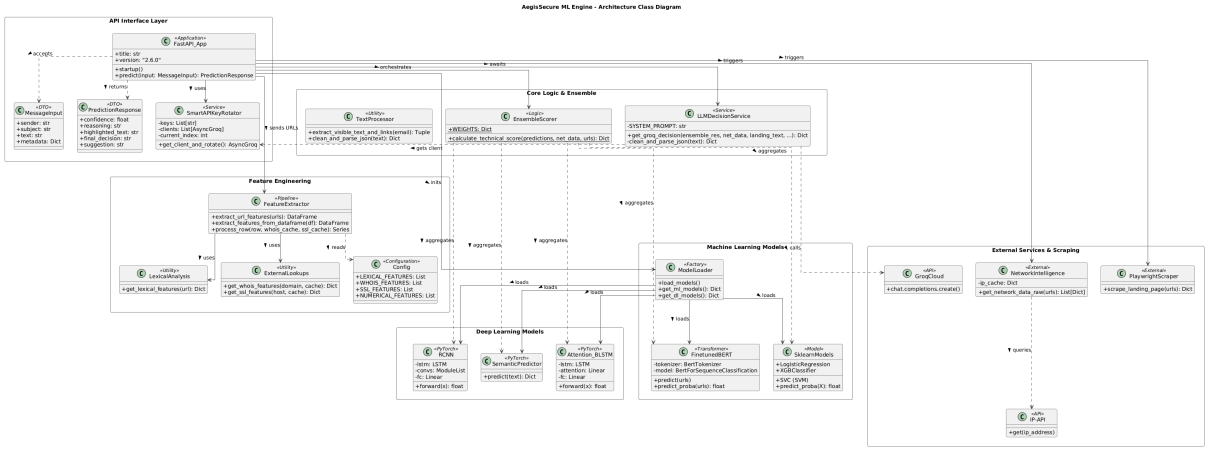

Figure 11: AI Model Class Diagram

7 Dynamic Modeling (Sequence Diagrams)

This section details the runtime interactions between system objects for key use cases.

7.1 Authentication Flow

The following sequence diagram illustrates the complete authentication process including user registration, login, OTP verification, password reset, and JWT token generation.

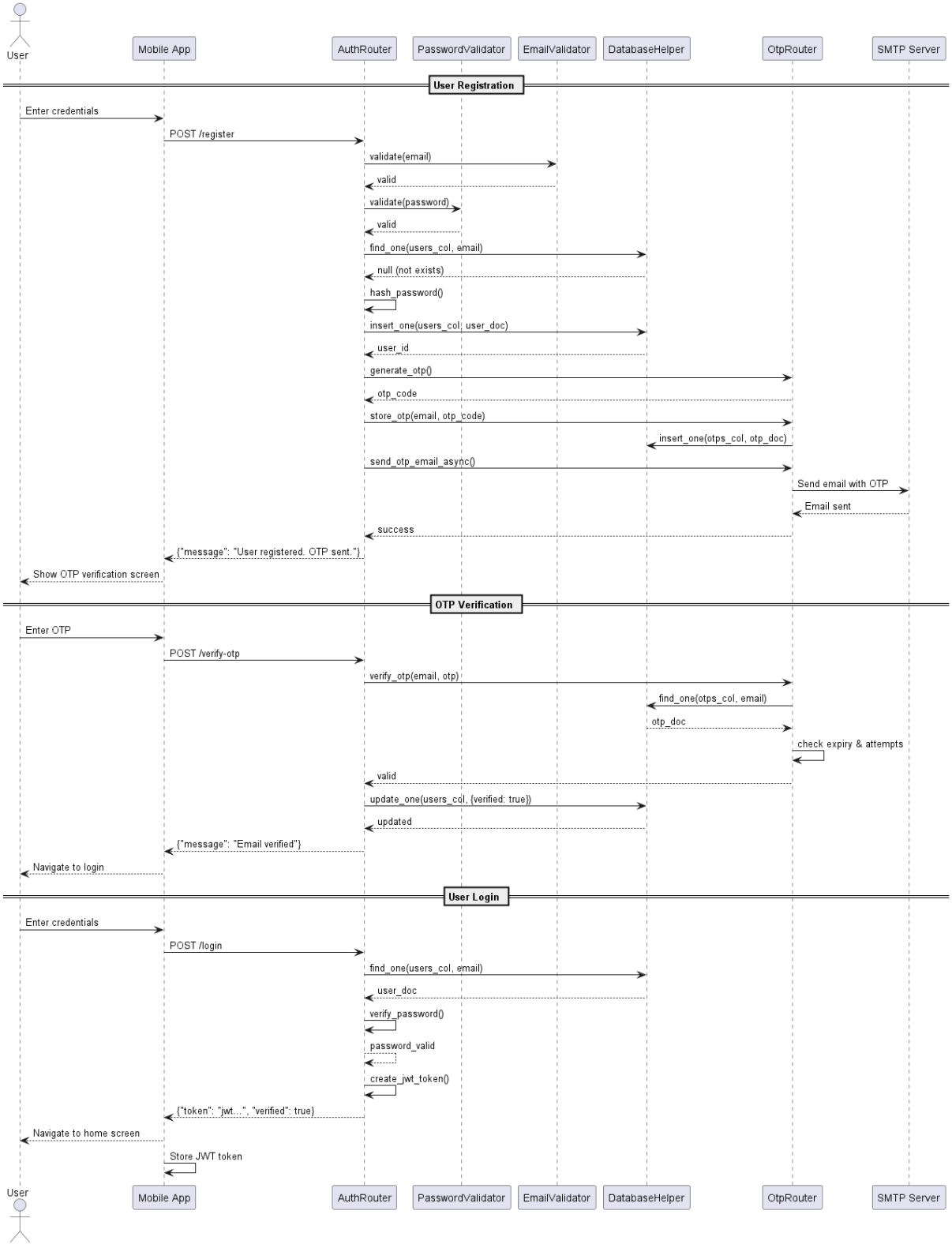

Figure 12: Sequence Diagram: Authentication and User Management

7.2 Detailed Authentication Flows

7.2.1 Sign In & Sign Up Flow

Illustrates the process of a user registering or logging into the system using Email/Password.

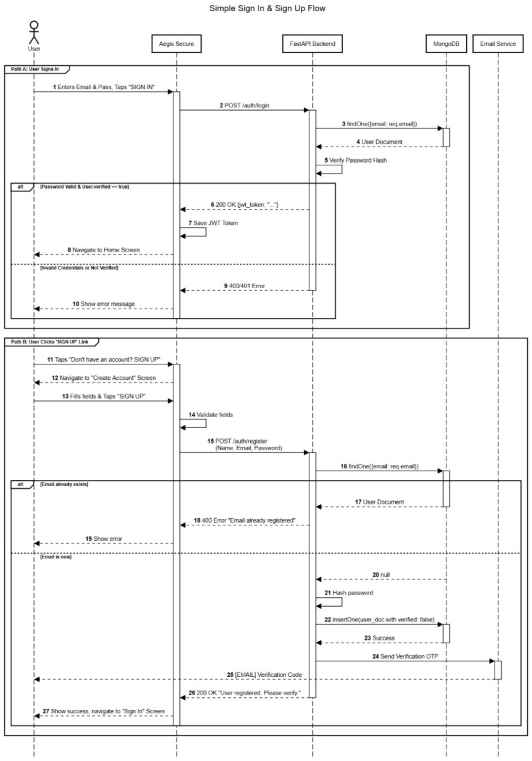

Figure 13: Sequence Diagram: Simple Sign In & Sign Up

7.2.2 Change Password Flow

Shows the secure flow for resetting or changing a user password via OTP verification.

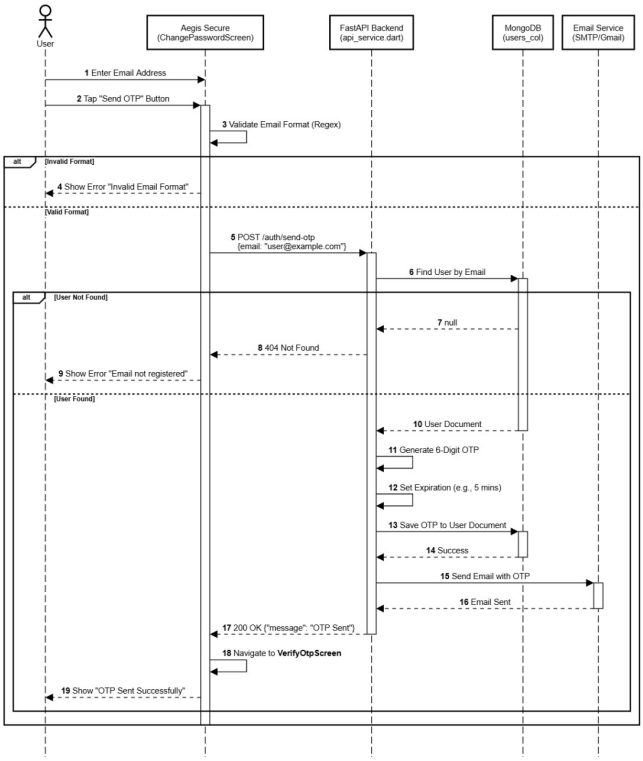

Figure 14: Sequence Diagram: Change Password Flow

7.3 SMS Analysis Flow

This diagram shows the end-to-end process of SMS synchronization, spam detection, and result notification.

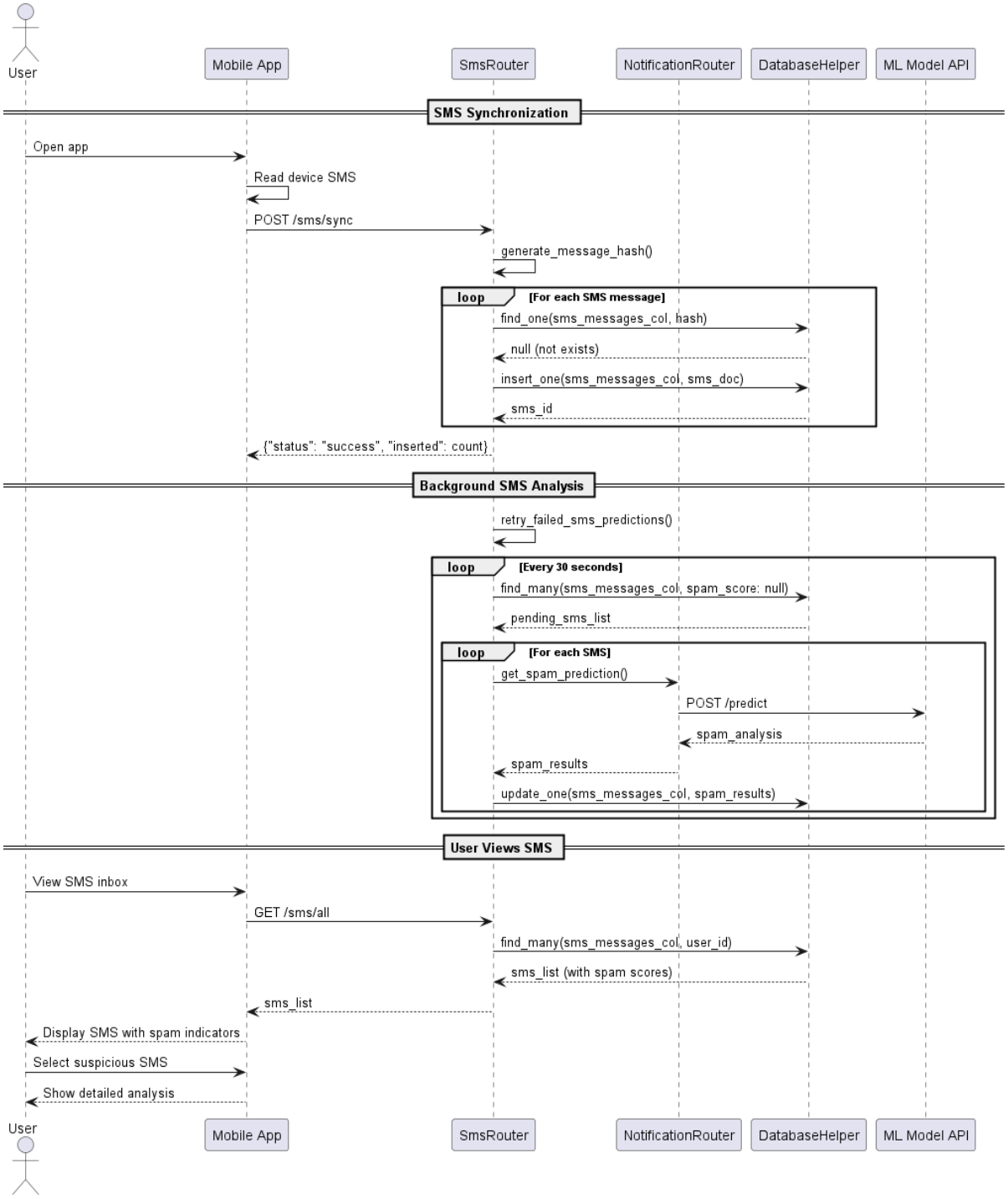

Figure 15: Sequence Diagram: SMS Analysis Pipeline

7.4 Data Synchronization

7.4.1 SMS Sync Flow

Details how the app requests permissions, reads local SMS, and syncs them to the backend DB.

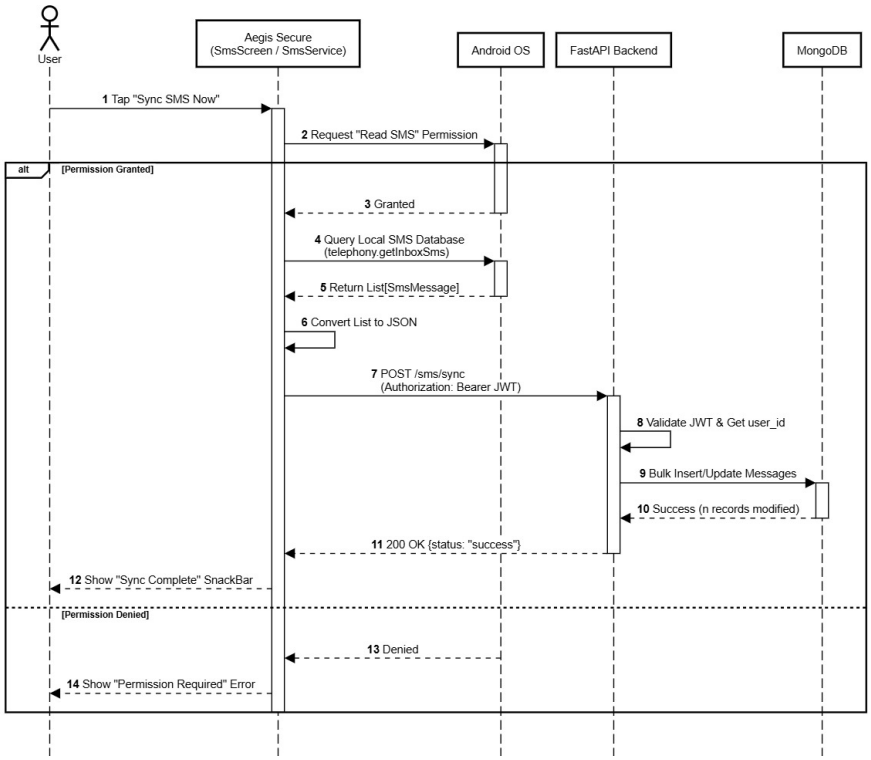

Figure 16: Sequence Diagram: SMS Synchronization

7.4.2 Gmail Fetch Flow

Shows the interaction between the app, backend, and Gmail API to fetch user emails.

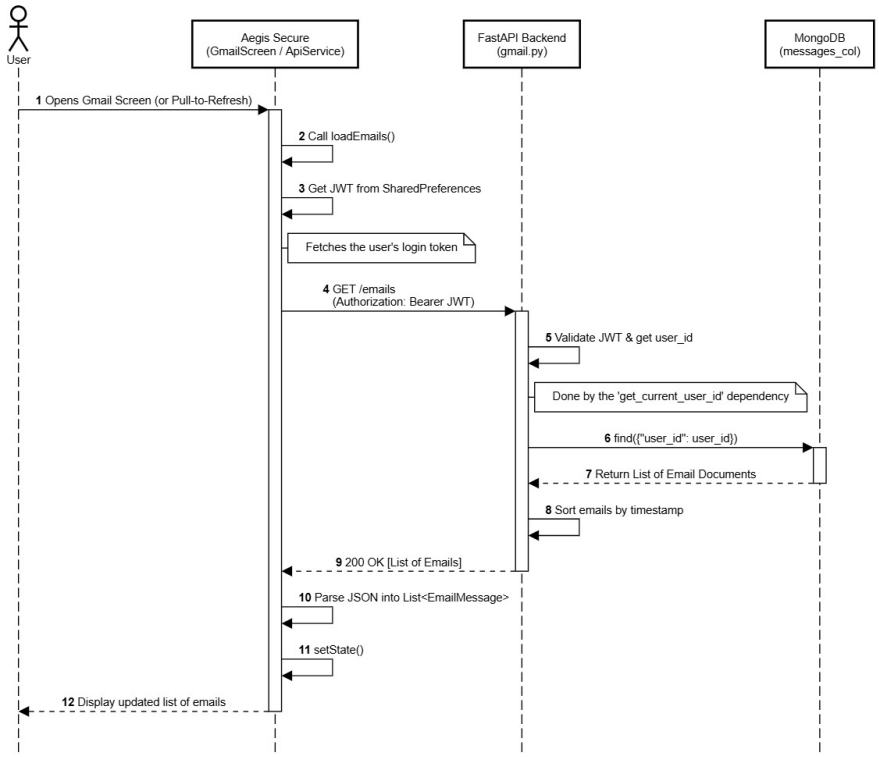

Figure 17: Sequence Diagram: Gmail Fetching

7.4.3 Google OAuth Flow

Illustrates the secure token exchange process for connecting a Google Account.

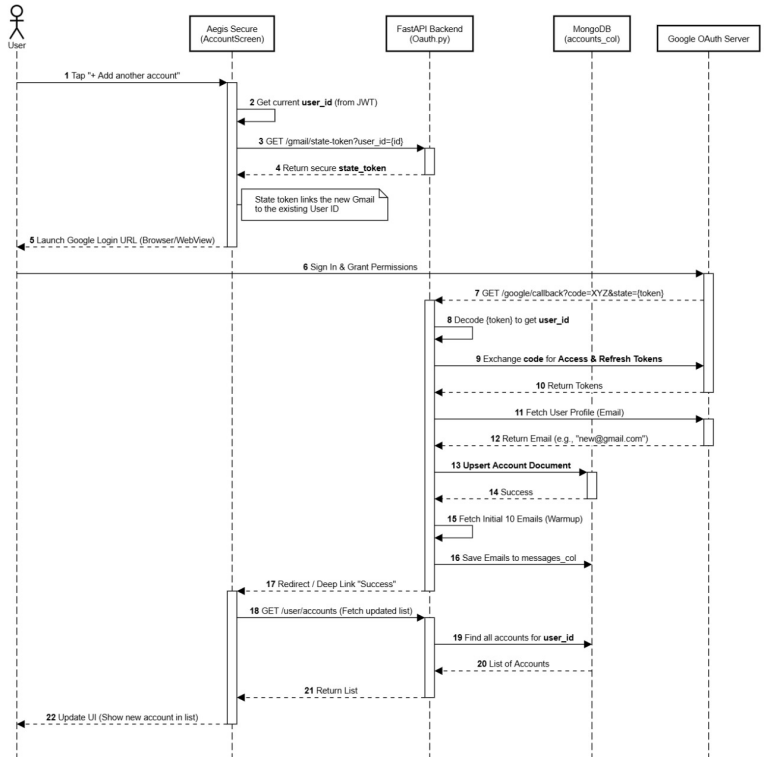

Figure 18: Sequence Diagram: Google OAuth Integration

7.5 Email Analysis Flow

This diagram illustrates the Gmail OAuth integration, email fetching, and spam analysis workflow.

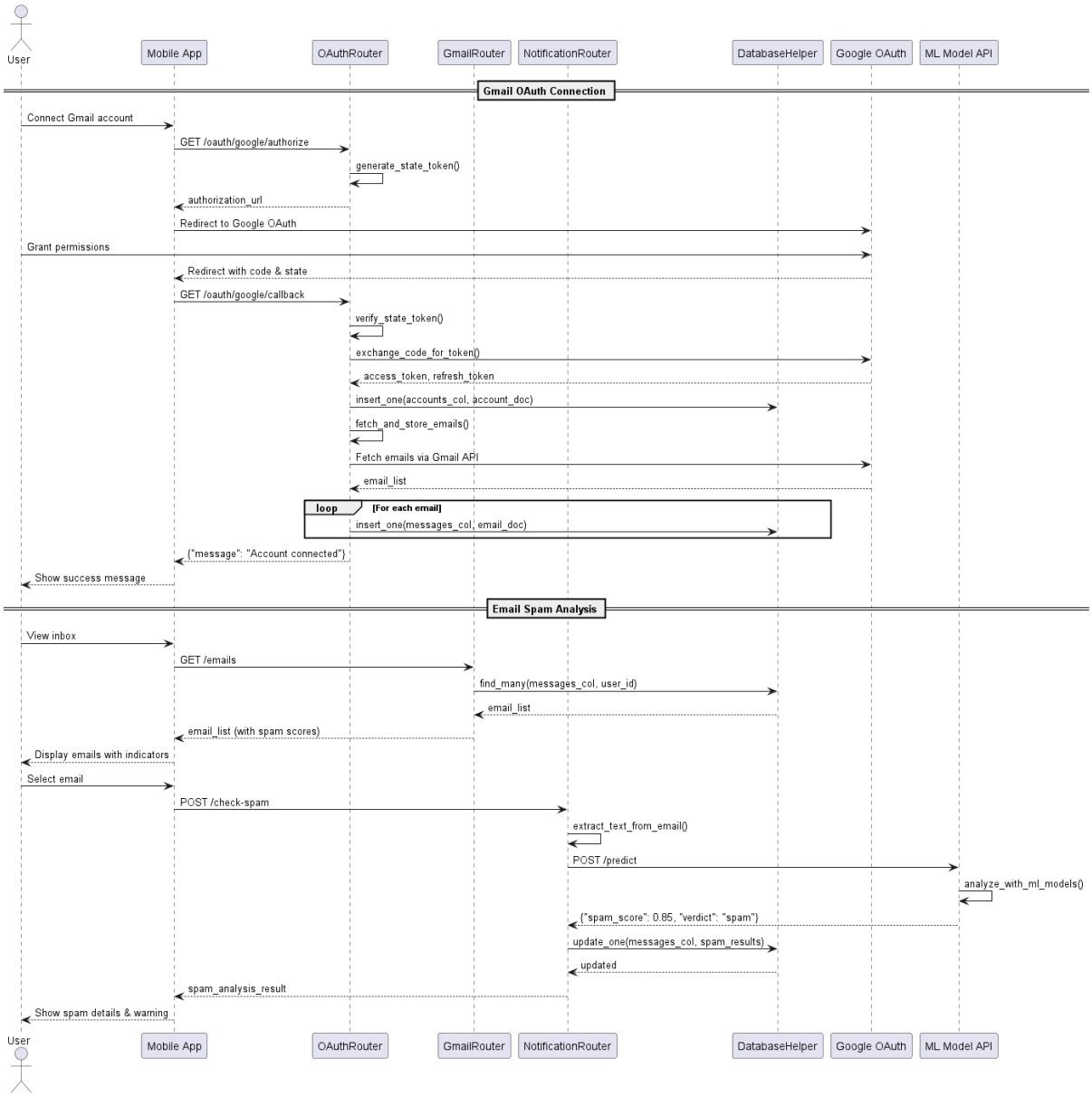

Figure 19: Sequence Diagram: Email Analysis Pipeline

7.6 Threat Detection Pipeline

7.6.1 Manual Text Analysis

Describes the flow when a user manually pastes text for immediate analysis.

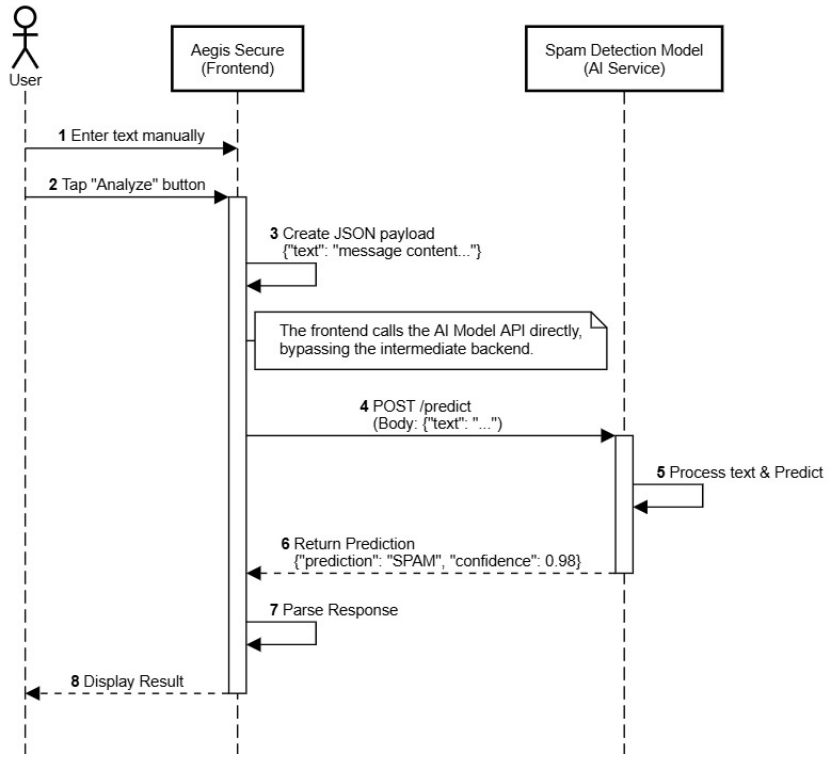

Figure 20: Sequence Diagram: Manual Text Analysis

7.6.2 Async Analysis Pipeline (Pub/Sub)

Shows the backend architecture for processing incoming messages asynchronously using workers.

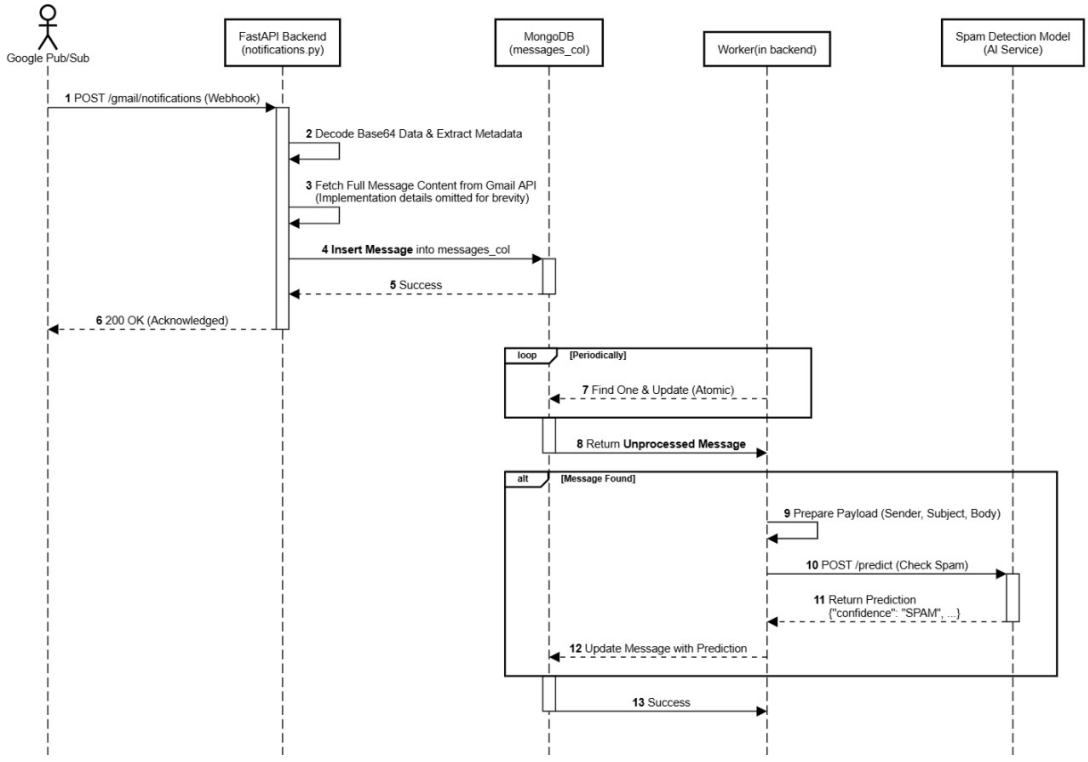

Figure 21: Sequence Diagram: Backend Async Analysis Worker

7.7 User Interaction

7.7.1 Local Search Flow

Details the internal logic for filtering and searching through the list of messages on the client side.

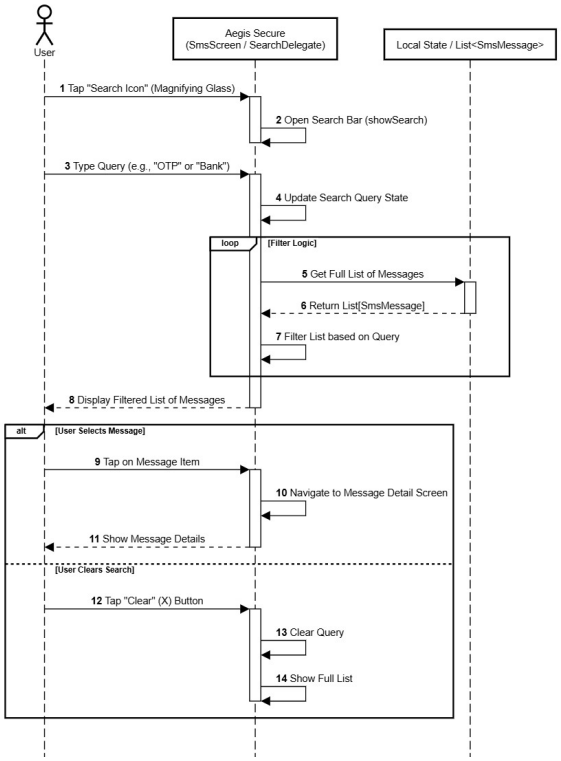

Figure 22: Sequence Diagram: Local SMS Search

7.7.2 Permission Handling

Shows the flow for requesting Android runtime permissions for SMS access.

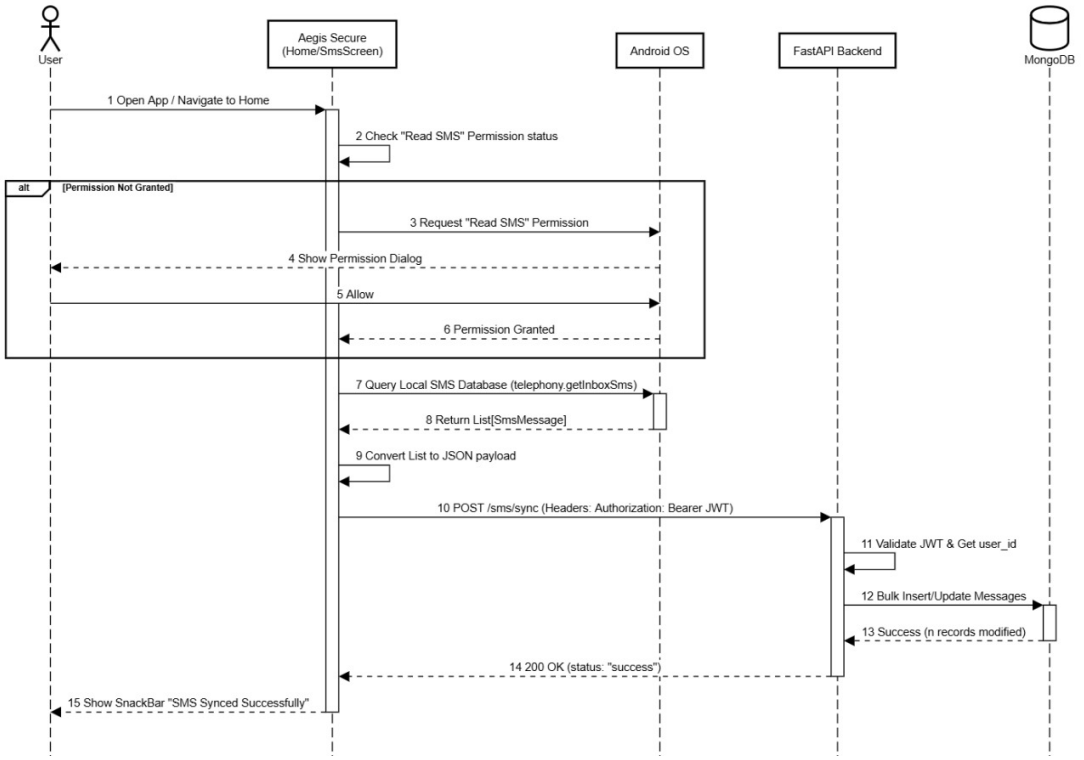

Figure 23: Sequence Diagram: App Permission Handling

8 Behavioral Modeling

8.1 Activity Diagrams

Activity diagrams show the workflow logic and decision points within key system processes.

8.1.1 Email Analysis Workflow

This diagram illustrates the complete workflow for analyzing emails from Gmail synchronization to threat detection and user notification.

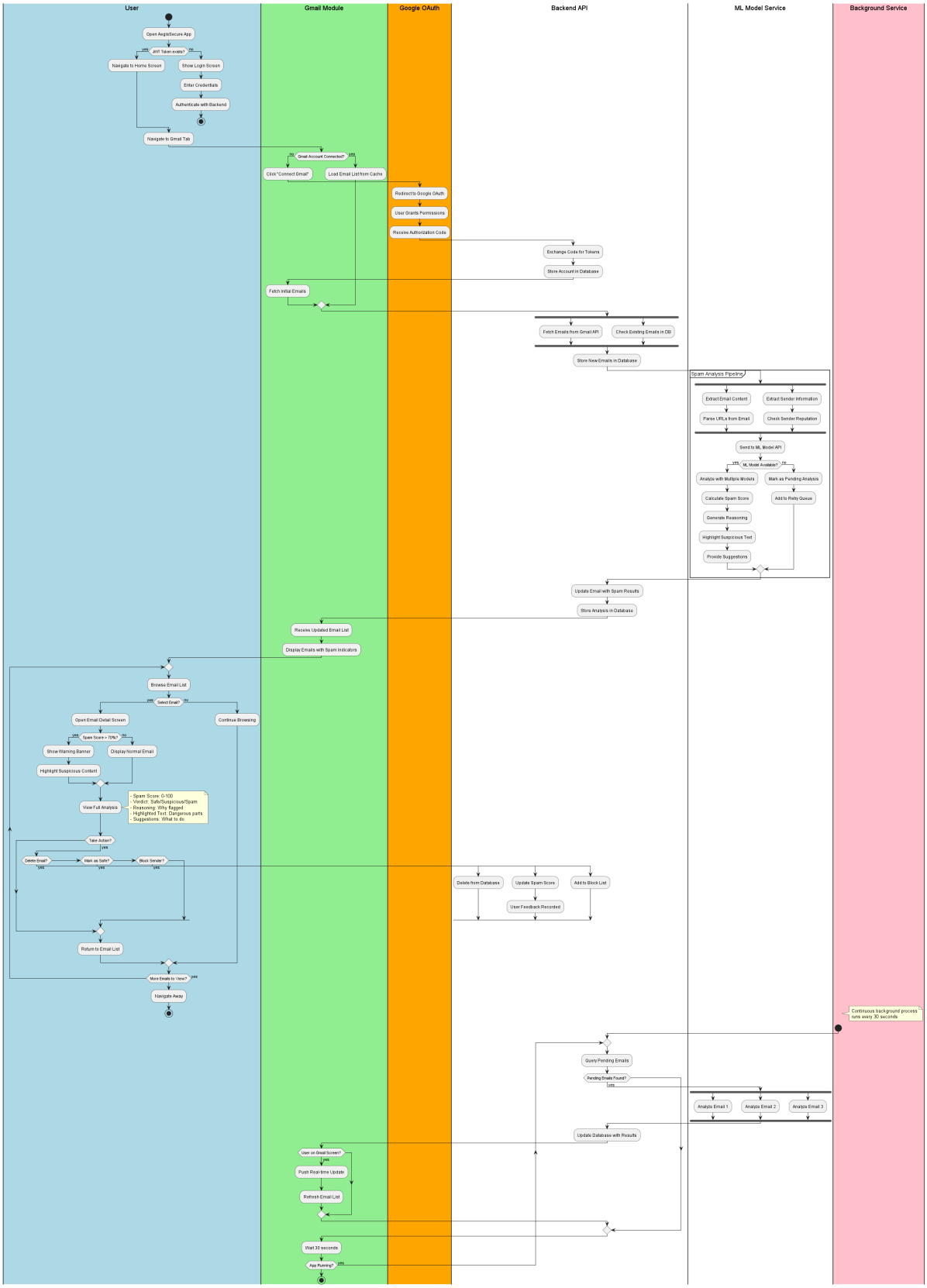

Figure 24: Activity Diagram: Email Analysis Workflow

8.1.2 SMS Analysis Workflow

This diagram shows the end-to-end process of SMS analysis including permission handling, data extraction, and spam classification.

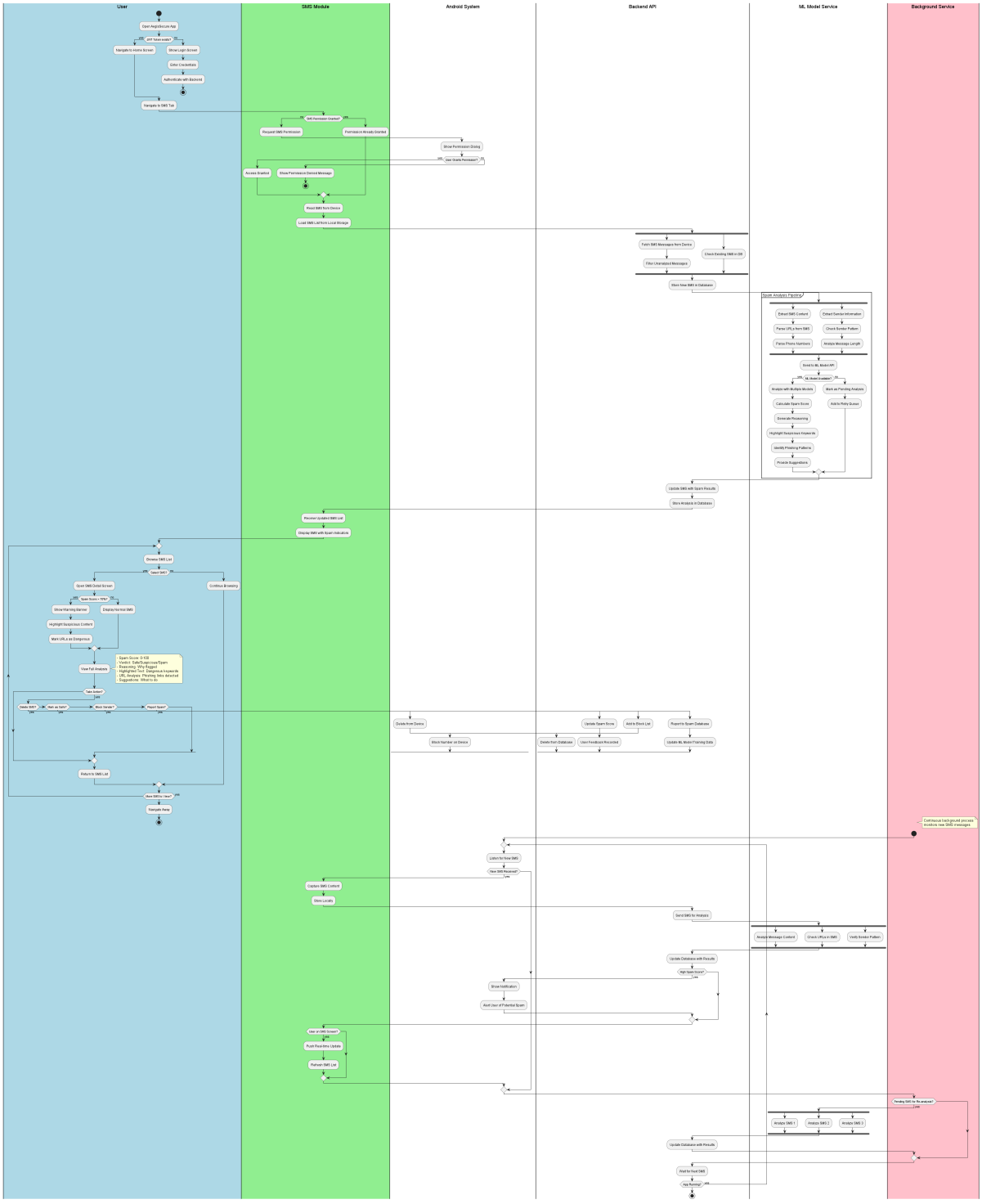

Figure 25: Activity Diagram: SMS Analysis Workflow

8.2 State Diagrams

State diagrams model the lifecycle and state transitions of key system entities.

8.2.1 User Account Lifecycle

This diagram shows the various states a user account transitions through from registration to active usage.

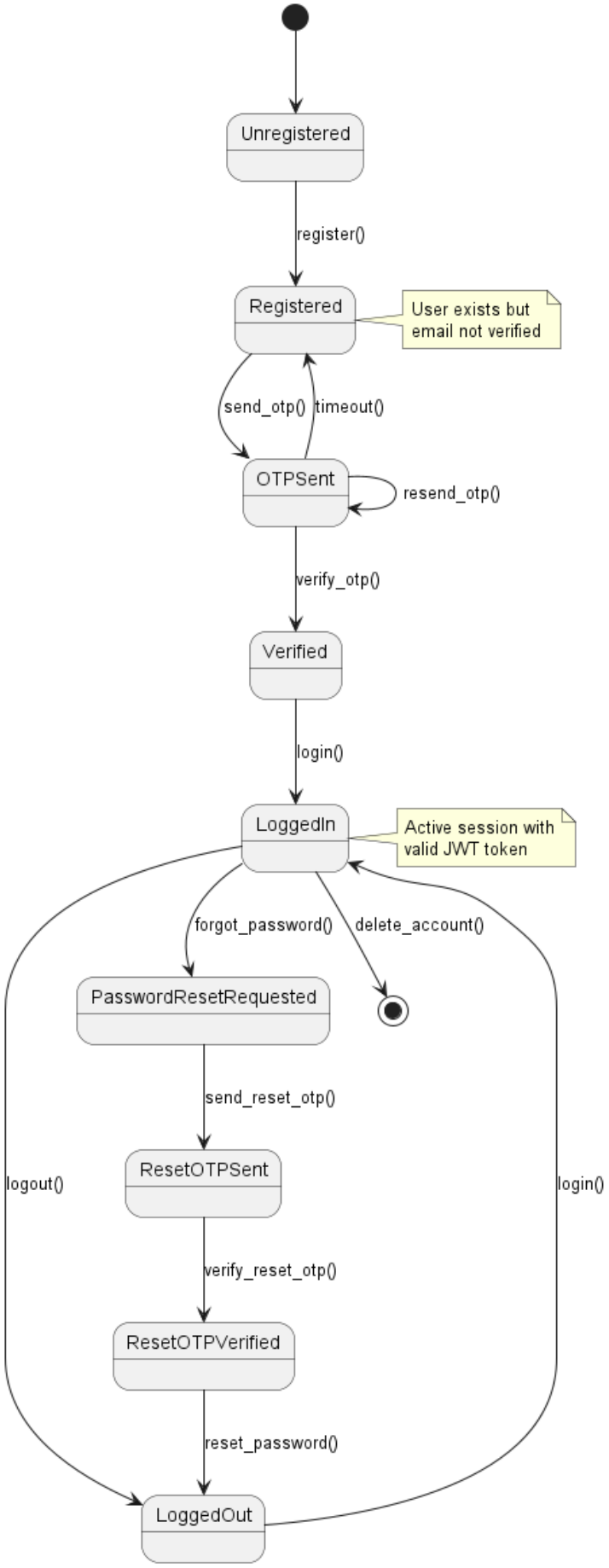

Figure 26: State Diagram: User Account Lifecycle

8.2.2 Email Analysis States

This diagram models the state transitions during email processing and analysis.

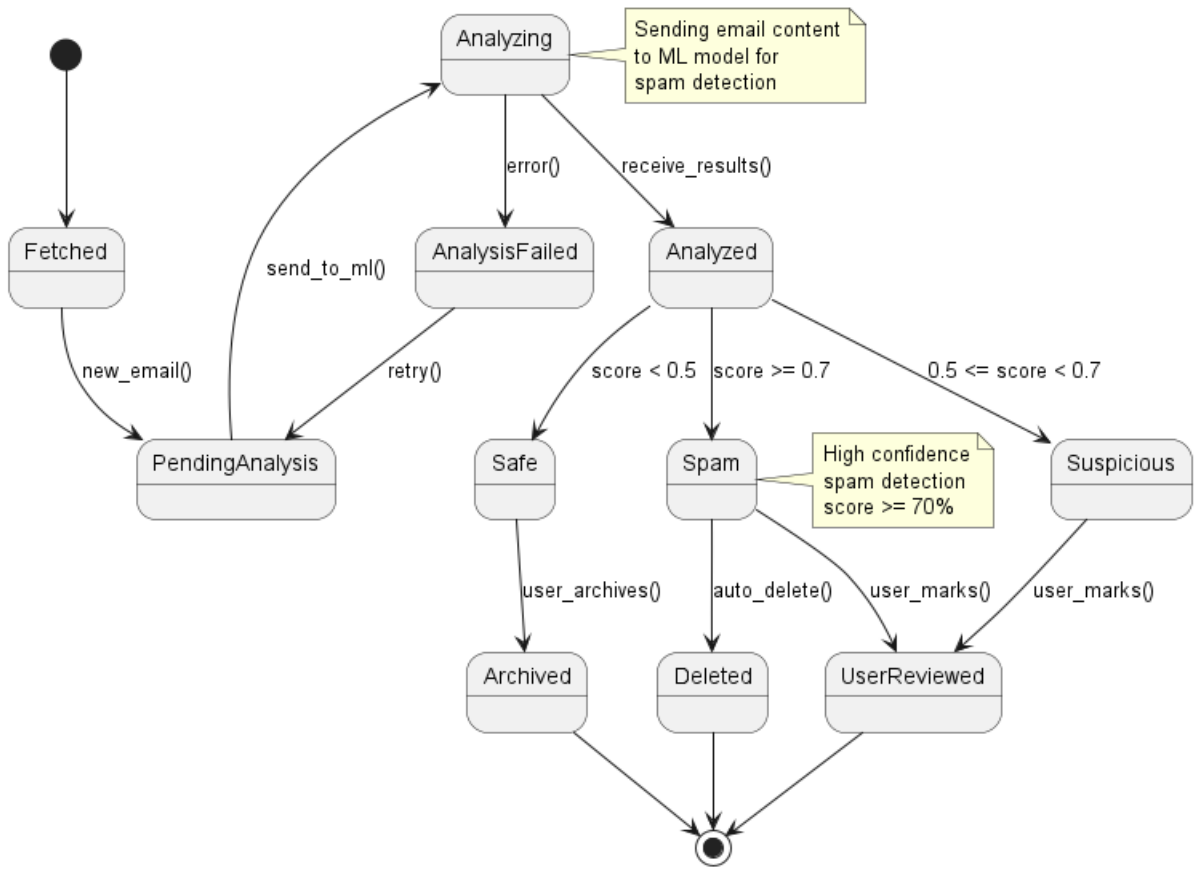

Figure 27: State Diagram: Email Analysis States

8.2.3 SMS Analysis States

This diagram shows the various states an SMS message goes through during the analysis pipeline.

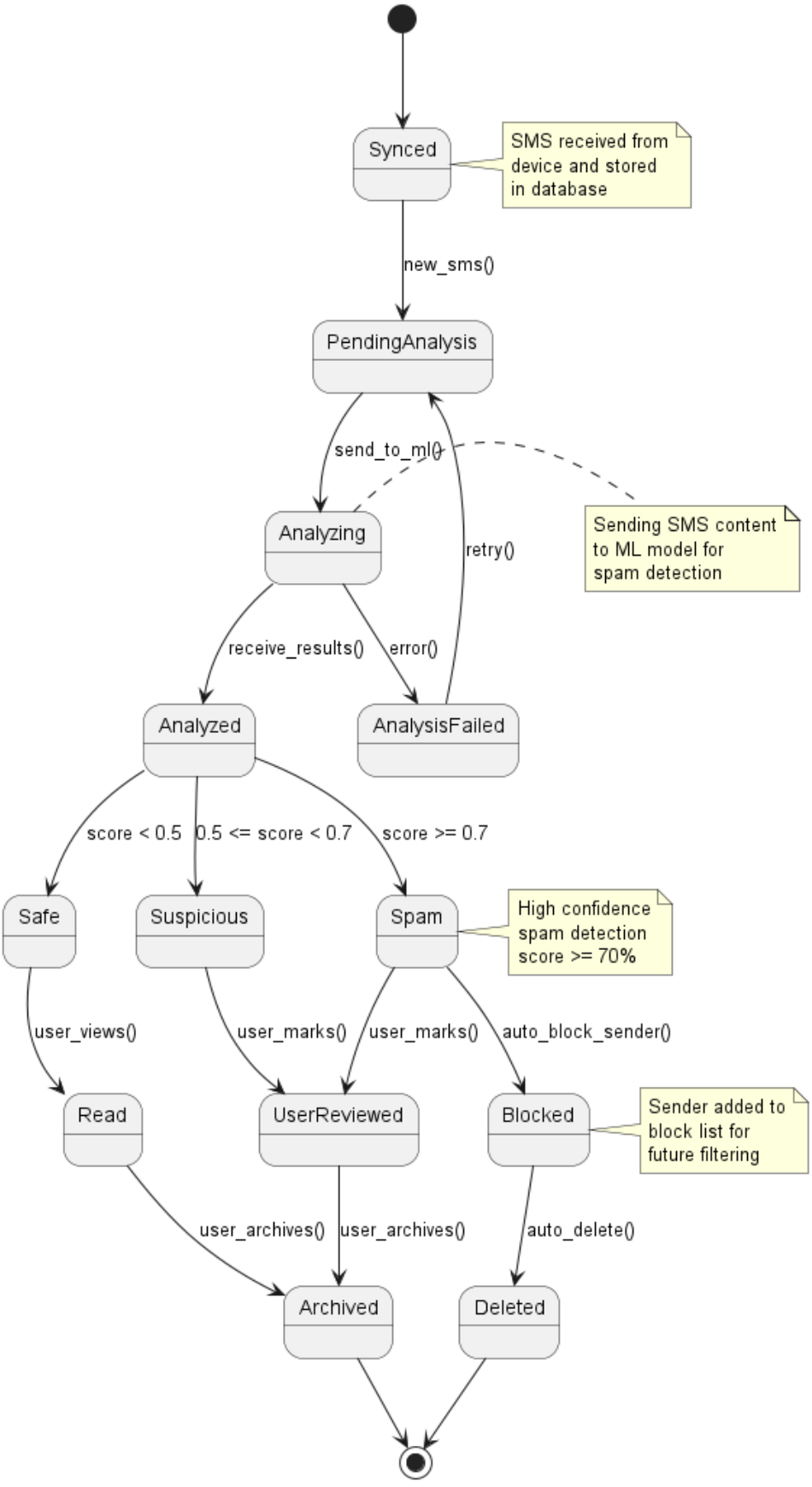

Figure 28: State Diagram: SMS Analysis States

9 Navigation Flow

This section illustrates the user interface navigation structure and screen transitions within the mobile application.

9.1 Application Navigation Diagram

The navigation flow diagram shows how users move between different screens and features in the AegisSecure mobile app.

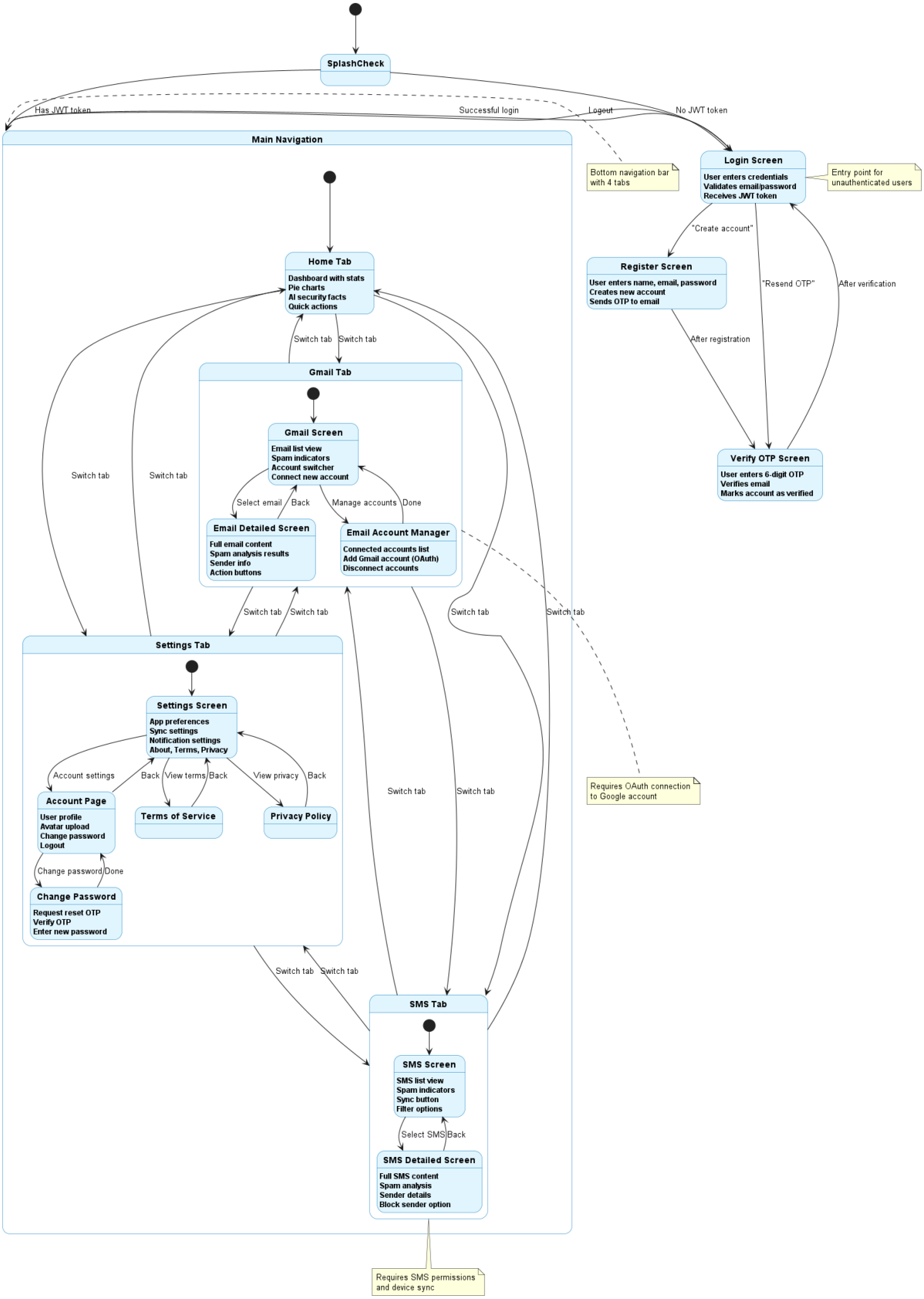

Figure 29: Navigation Flow Diagram: Screen Transitions and User Journey

This diagram maps out:

- **Authentication Screens:** Login, Registration, OTP Verification, Password Reset
- **Main Dashboard:** Central hub with threat analytics and quick actions
- **SMS Module:** SMS list view, detailed analysis screen, search functionality
- **Email Module:** Gmail account connection, email list, detailed threat reports
- **Settings:** Account management, notification preferences, security settings

10 Deployment Architecture

This section describes how the AegisSecure system components are deployed across different infrastructure nodes and their communication protocols.

10.1 Deployment Diagram

The deployment diagram illustrates the physical architecture showing servers, mobile devices, databases, and external services.

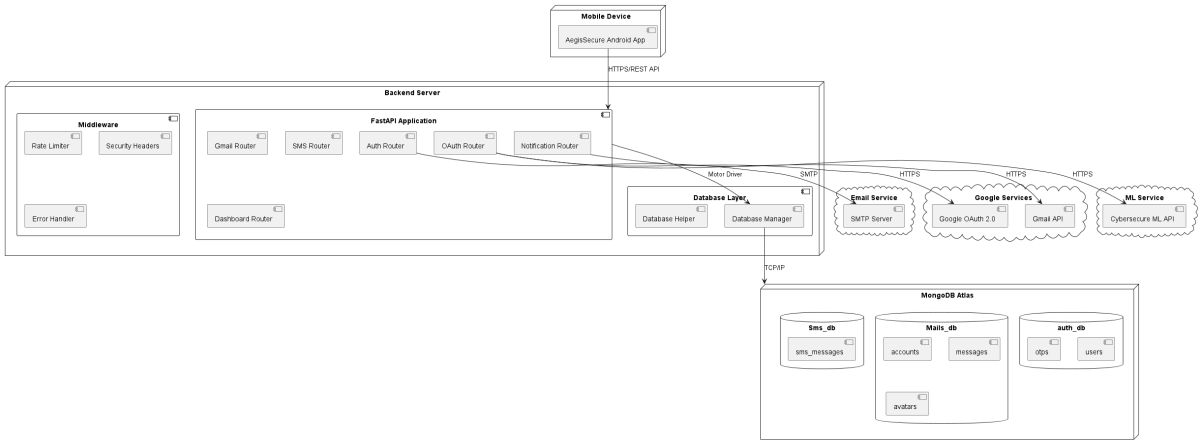

Figure 30: Deployment Diagram: Infrastructure and Component Distribution

10.2 Deployment Configuration

- **Mobile Client Node:** Android device running Flutter application
- **API Server Node:** FastAPI backend hosted on cloud infrastructure (AWS/Azure/GCP)
- **Database Node:** MongoDB Atlas cluster for data persistence
- **AI Service Node:** ML model serving infrastructure with GPU support
- **External Services:** Gmail API, SMS Gateway, OAuth providers

Communication protocols include HTTPS for API calls, WebSocket for real-time notifications, and secure OAuth2.0 for authentication.

11 Conclusion

This Design Document outlines the comprehensive architecture of the AegisSecure system. By adhering to the Client-Server model and utilizing asynchronous processing for heavy AI workloads, the system is designed to be scalable, responsive, and secure.

The document has covered:

- **Use Case Model:** Identifying actors and functional requirements
- **System Architecture:** High-level structure, component organization, and package structure
- **Object Design:** Application and solution domain objects
- **Class Diagrams:** Complete backend hierarchy and specialized frontend/AI models
- **Dynamic Modeling:** Runtime interactions through comprehensive sequence diagrams
- **Behavioral Modeling:** Process workflows via activity diagrams and state transitions
- **Navigation Flow:** User interface structure and screen transitions
- **Deployment Architecture:** Physical infrastructure and component distribution

The models provided serve as the blueprint for the implementation phase, ensuring that all team members have a unified understanding of the system's design and architecture.