



**CEBU INSTITUTE OF TECHNOLOGY**  
**UNIVERSITY**

# **IT342-Section SYSTEMS INTEGRATION AND ARCHITECTURE 1**

---

## **FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)**

---

Project Title: InStock

Prepared By: Gregory Ivan Onyx M. Badinas

Date of Submission: 02/03/2026

Version: 1

# Table of Contents

1.	Introduction.....	3
1.1.	Purpose.....	3
1.2.	Scope.....	3
1.3.	Definitions, Acronyms, and Abbreviations.....	3
2.	Overall Description.....	3
2.1.	System Perspective.....	3
2.2.	User Classes and Characteristics.....	3
2.3.	Operating Environment.....	3
2.4.	Assumptions and Dependencies.....	3
3.	System Features and Functional Requirements.....	3
3.1.	Feature 1:.....	3
3.2.	Feature 2:.....	3
4.	Non-Functional Requirements.....	3
5.	System Models (Diagrams).....	4
5.1.	ERD.....	4
5.2.	Use Case Diagram.....	4
5.3.	Activity Diagram.....	4
5.4.	Class Diagram.....	4
5.5.	Sequence Diagram.....	4
6.	Appendices.....	4

## 1. Introduction

### 1.1. Purpose

The purpose of this document is to define the functional and non-functional requirements for the **Identity and Access Management (IAM) subsystem** of InStock. This subsystem provides the foundational security architecture required to manage user lifecycles across the Android mobile application, React web application, and Spring Boot backend.

### 1.2. Scope

The **InStock** system consists of three main components operating within a unified ecosystem:

1. **Mobile Client (Android/Kotlin):** A native application optimized for on-the-go pantry management and cooking assistance.
2. **Web Client (React):** A responsive web interface for recipe planning, profile management, and broader discovery.
3. **Backend (Spring Boot):** A centralized server providing RESTful APIs for authentication, data persistence, and the recipe matching logic.

### 1.3. Definitions, Acronyms, and Abbreviations

- **FRS:** Functional Requirements Specification.
- **API:** Application Programming Interface.
- **REST:** Representational State Transfer; the architectural style for the backend web services.
- **JWT:** JSON Web Token; used for stateless user authentication.
- **IAM:** Identity and Access Management

## 2. Overall Description

### 2.1. System Perspective

**InStock** operates as a distributed client-server system.

- **Presentation Layer (Frontend):** The Android and React clients capture user credentials and manage session states (Login/Logout).

- **Business Logic Layer (Backend):** The Spring Boot application acts as the "brain," enforcing security policies, password hashing, and role-based access control during registration and login.
- **Data Layer:** A centralized relational database (PostgreSQL/MySQL) provides data durability for user profiles and hashed credentials

## 2.2. User Classes and Characteristics

- **End User:** Individuals who register accounts to personalize their experience. They require a seamless, high-usability interface for authentication.
- **System Administrator:** Responsible for monitoring system health and managing user accounts through the backend administrative interface.

## 2.3. Operating Environment

- **Mobile Client:** Android devices (Android 7.0+).
- **Web Client:** Modern browsers utilizing **Secure & HttpOnly cookies** to protect session tokens from malicious JavaScript.
- **Backend Server:** Spring Boot environment managing RESTful API endpoints for authentication.
- **Network:** Requires an active connection to verify credentials against the server in real-time.

## 2.4. Assumptions and Dependencies

- **Connectivity:** Users must be online to register or log in; however, the system uses **JWT (JSON Web Tokens)** to allow for stateless session verification once the initial login is successful.
- **Token Integrity:** It is assumed that the client securely handles the **Logout** process by invalidating local tokens and clearing session metadata to prevent unauthorized access.

## 3. System Features and Functional Requirements

### 3.1. Feature 1:

Description: Enables new users to create a unique identity within the InStock ecosystem. This feature follows a "Backend-first" strategy, ensuring that all data validation and password hashing are centralized within the Spring Boot business layer to maintain consistency across the React and Android platforms.

Functional Requirements:

- **REQ-1.1:** The system shall provide a registration interface to capture user details, including Email, Password, and Full Name.
- **REQ-1.2:** The Spring Boot backend shall validate the uniqueness of the email address and enforce password complexity requirements before persistence.

- **REQ-1.3:** The system shall utilize industry-standard hashing (e.g., BCrypt) to secure passwords at rest in the relational database.
- **REQ-1.4:** Upon successful registration, the system shall provide a clear, actionable success message to the user or an error message if the registration fails.

### **3.2. Feature 2:**

Description: The mechanism for verifying user identity and establishing a secure session. To prevent authorization code injection, the system implements the Authorization Code with PKCE flow, which is mandatory for modern mobile and SPA security.

Functional Requirements:

- **REQ-2.1:** The system shall authenticate users via a REST API and issue a **JSON Web Token (JWT)** for stateless session management.
- **REQ-2.2:** The backend shall issue short-lived Access Tokens and long-lived Refresh Tokens.
- **REQ-2.3:** The Android client shall store tokens in the **Android Keystore**, providing encryption-at-rest for sensitive credentials.
- **REQ-2.4:** The React client shall store tokens in **Secure & HttpOnly cookies** to prevent access by malicious JavaScript (XSS).

### **3.3. Feature 3:**

Description: Ensures the secure termination of the user session across distributed clients. This process prioritizes data integrity by clearing sensitive session metadata.

Functional Requirements:

- **REQ-3.1:** The system shall provide a "Logout" function that clears the local JWT and session data from the client's secure storage.
- **REQ-3.2:** The Spring Boot backend shall provide an endpoint to invalidate or blacklist the current Refresh Token to prevent further access.
- **REQ-3.3:** The system shall immediately redirect the user to the login interface upon successful session termination.
- **REQ-3.4:** The system shall "Fail Gracefully," ensuring local tokens are cleared even if a network error occurs during the logout request.

## **4. Non-Functional Requirements**

### **4.1. Security**

- Authentication: All user endpoints must be secured using OAuth 2.0 or JWT (JSON Web Tokens). Passwords must be hashed using BCrypt before storage.

- Data Privacy: User data, specifically email and dietary restrictions, must be encrypted at rest and in transit (HTTPS/TLS 1.3).

#### 4.2. Performance

- Latency: The Recipe Generation API call should return results in under 30 seconds for a standard query of <20 ingredients.
- Scalability: The Spring Boot backend must be stateless to support horizontal scaling across multiple instances if user load increases.

#### 4.3. Reliability & Availability

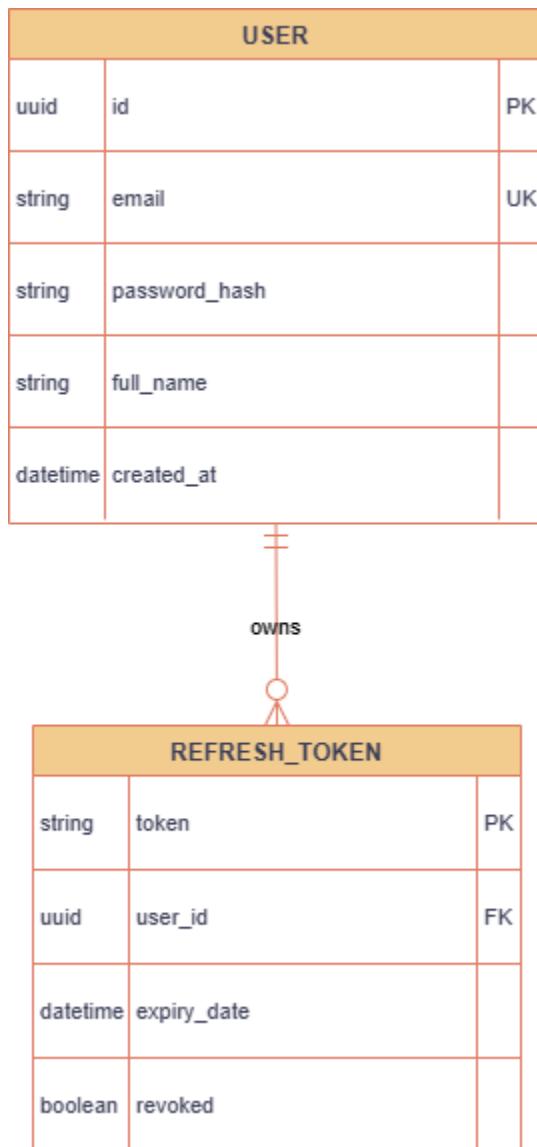
- **Centralized Error Handling:** Use middleware to categorize errors by severity. For example, a "Database Timeout" during registration is a **P0 (Critical)** failure, while an "Invalid Email Format" is a **P2 (Minor)** user-input error.
- **User-Centric Feedback:** Rather than exposing technical stack traces or generic 500 codes, the system must provide actionable, plain-language messages (e.g., "Account already exists" or "Check your internet connection").
- **Availability:** The authentication subsystem must prioritize **Partition Tolerance and Availability** (referencing the CAP Theorem), ensuring users can still access the system even if certain non-critical background services are down.

#### 4.4. Usability

- Feedback: The system must provide visual feedback (loading spinners, toast notifications) for Registration, Logging in, and Logout.

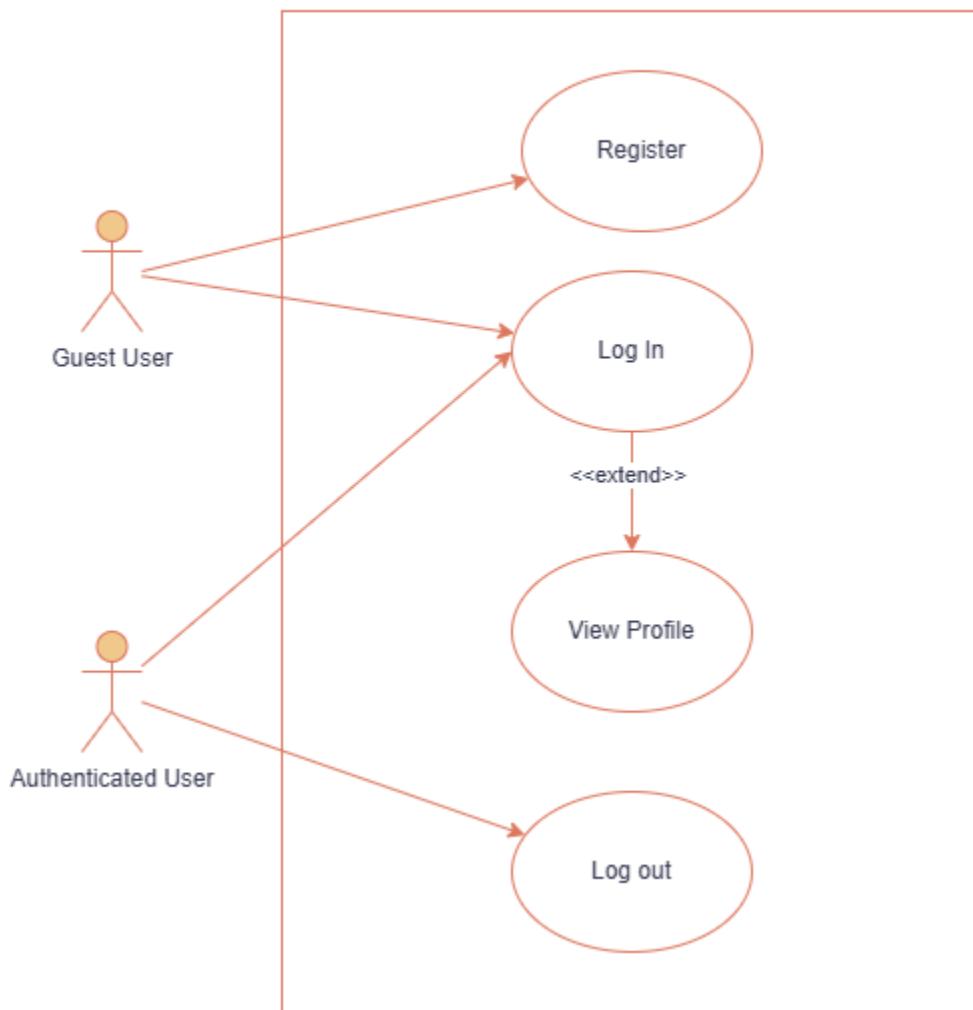
## 5. System Models (Diagrams)

### 5.1. ERD

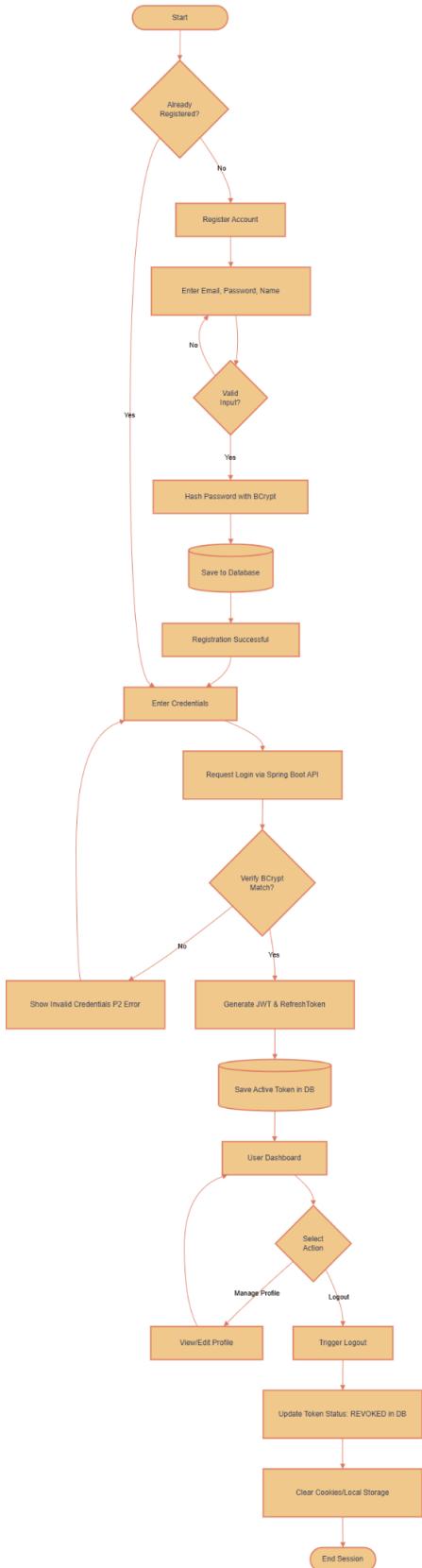


## 5.2. Use Case Diagram

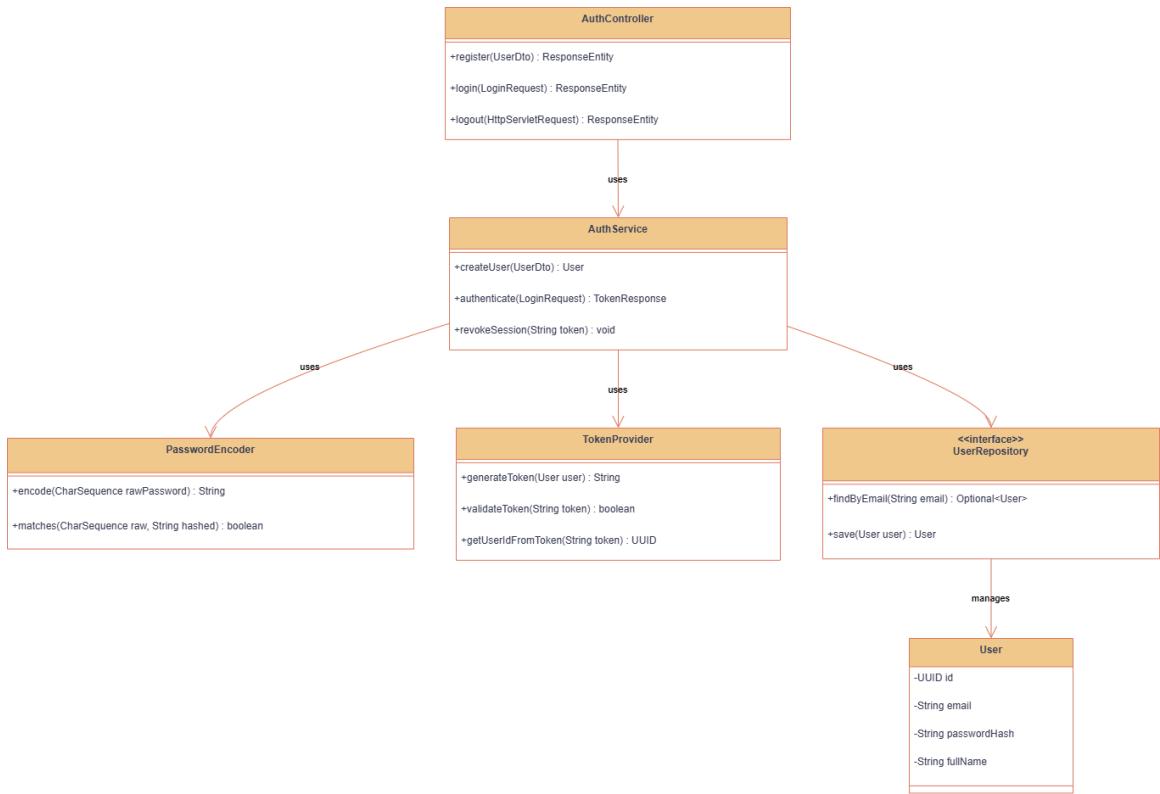
### System Registration and Authentication



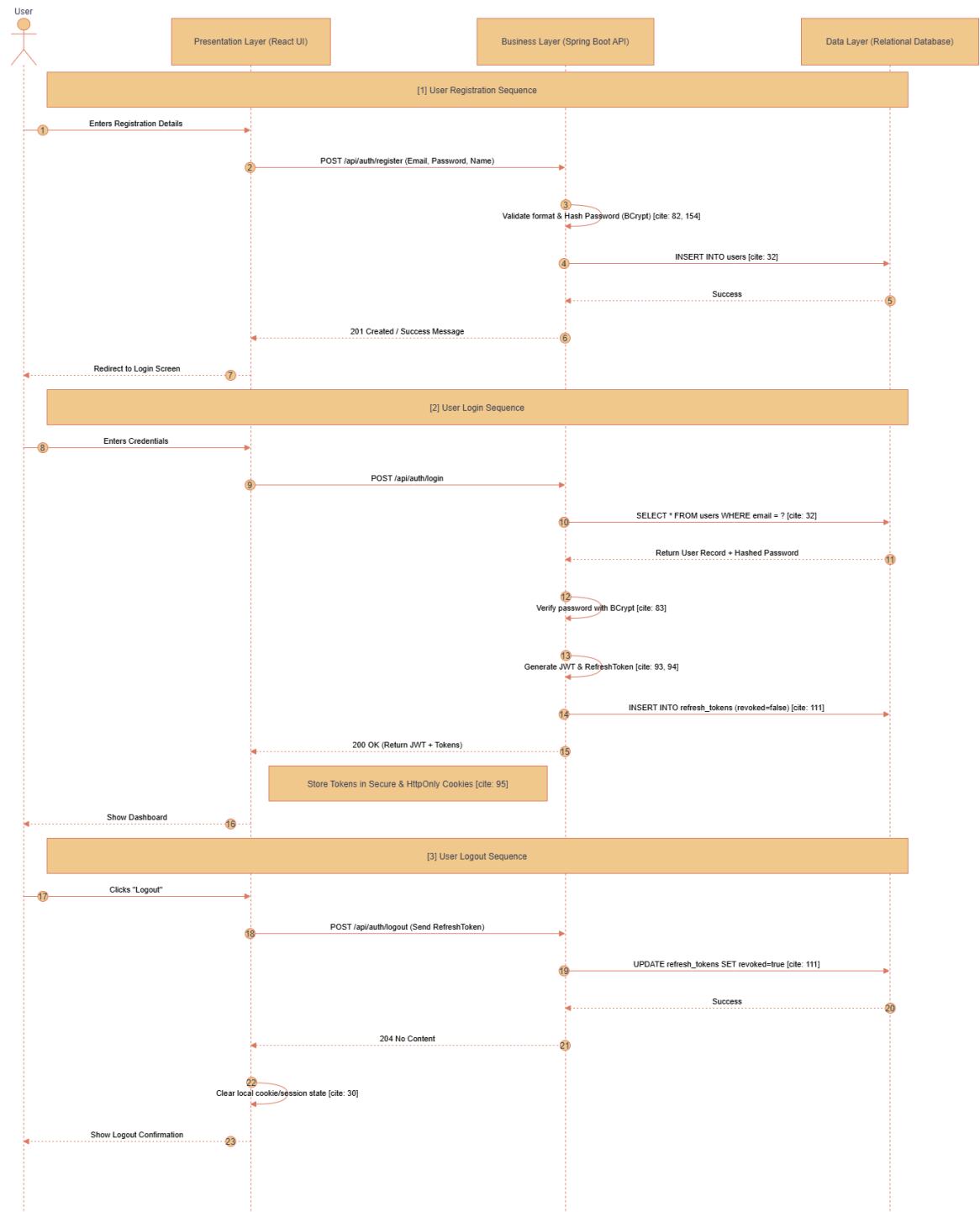
### 5.3. Activity Diagram



## 5.4. Class Diagram



## 5.5. Sequence Diagram



## 6. Appendices

- [1] System Integration & Architecture Course, "System integration and architecture," *Scribd*, Feb. 01, 2026. [Online]. Available:  
<https://fr.scribd.com/document/559863705/System-Integration-and-Architecture-P1>.
- [2] "Chapter 1 - Introduction to system integration and architecture," *SlideShare*, Feb. 01, 2026. [Online]. Available:  
<https://www.slideshare.net/slideshow/chapter-1-introduction-to-system-integration-and-architecturepdf/263057894>.
- [3] "What is enterprise application integration? - EAI explained," *AWS*, Feb. 01, 2026. [Online]. Available: <https://aws.amazon.com/what-is/enterprise-application-integration/>.
- [4] IBM, "What is enterprise application integration?," *IBM Think*, Feb. 01, 2026. [Online]. Available: <https://www.ibm.com/think/topics/enterprise-application-integration>.
- [5] NareshIT, "How full stack Java projects work: Frontend, backend & database," *NareshIT Blogs*, Feb. 01, 2026. [Online]. Available:  
<https://nareshit.com/blogs/inside-a-full-stack-java-project-how-frontend-backend-and-database-work-together>.
- [6] Simform, "Mobile application architecture: Layers, types, principles, factors," *Simform Blog*, Feb. 01, 2026. [Online]. Available:  
<https://www.simform.com/blog/mobile-application-architecture/>.
- [7] Techstack, "Exploring modern web app architectures: Trends and best practices for 2026," *Techstack Blog*, Feb. 01, 2026. [Online]. Available:  
<https://tech-stack.com/blog/modern-application-development/>.
- [8] AWS, "What is middleware?," *AWS*, Feb. 01, 2026. [Online]. Available:  
<https://aws.amazon.com/what-is/middleware/>.
- [9] OWASP, "Mobile app authentication architectures," *OWASP Mobile Application Security*, Feb. 01, 2026. [Online]. Available:  
<https://mas.owasp.org/MASTG/0x04e-Testing-Authentication-and-Session-Management/>.
- [10] Droidcon, "The complete guide to offline-first architecture in Android," *Droidcon*, Dec. 16, 2025. [Online]. Available:  
<https://www.droidcon.com/2025/12/16/the-complete-guide-to-offline-first-architecture-in-android/>.
- [11] GeeksforGeeks, "Sequence diagrams - Unified Modeling Language (UML)," *GeeksforGeeks*, Feb. 01, 2026. [Online]. Available:

<https://www.geeksforgeeks.org/system-design/unified-modeling-language-uml-sequence-diagrams/>.

[12] OneUptime, "How to document REST APIs with OpenAPI," *OneUptime Blog*, Jan. 27, 2026. [Online]. Available:

<https://oneuptime.com/blog/post/2026-01-27-openapi-documentation/view>.