**Lab1: Requirements Elicitation & Modeling**

Table of Contents

[1. Cover page 1](#_Toc215912683)

[2. Abstract/Summary 2](#_Toc215912684)

[3. Lab Specific Section: I. Requirements Elicitation & Modeling 2](#_Toc215912685)

[3.1 Software Requirements Specifications (SRS) 2](#_Toc215912686)

[3.2 Modeling Artifact: UML Use Case Diagram 4](#_Toc215912687)

[4. Architectural Design (Problem Analysis for Lab 2) 6](#_Toc215912688)

[5. Conclusion & Reflection 6](#_Toc215912689)

# 1. Cover page

· **Title**: Elicitation and Modeling Requirements for the NoZie Online Streaming Application

· **Course Info:** Software Architecture

· **Student Details:**

|  |  |
| --- | --- |
| Họ Tên | MSSV |
| Nguyễn Duy Minh | 23010359 |
| Nghiêm Đức Việt | 23010636 |
| Nguyễn Văn Nhật | 23010887 |
| Đặng Nhất Nhất | 23010345 |

· **Date**: December 6, 2025

# **2. Abstract/Summary**

The NoZie project aims to develop a robust, scalable movie streaming platform. This initial lab focused on requirements elicitation and modeling. We documented essential Functional Requirements (FRs), Non-Functional Requirements (NFRs), and Architecturally Significant Requirements (ASRs) in a structured format. The core system behavior was formally modeled using a UML Use Case Diagram, clearly defining system boundaries, external actors, and key behavioral relationships.

This foundation establishes the scope and the critical architectural drivers that will be addressed in subsequent labs, particularly the design of the Layered Architecture in the upcoming Lab 2.

# **3. Lab Specific Section: I. Requirements Elicitation & Modeling**

This section documents the deliverables for Lab 1, adhering to the specified reporting requirements.

## **3.1 Software Requirements Specifications (SRS)**

The following tables document the elicited requirements, which collectively form the basis for the Movie Streaming Platform design.

**3.1.1 Functional Requirements (FRs)**

These requirements define the specific behaviors and functions the system must provide.

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Priority** |
| **FR-01** | The system shall allow a **User** to **browse and search movies** by title, genre, actor, or release year. | High |
| **FR-02** | The system shall allow a **User** to **stream movies online** with adaptive video quality (e.g., 480p, 720p, 1080p). | Critical |
| **FR-03** | The system shall allow a **User** to **download movies for offline viewing**, subject to license and storage permissions. | Medium |
| **FR-04** | The system shall support **User Account Management**, including registration, authentication, and password recovery. | High |
| **FR-05** | The system shall enable users to **add movies to Favorites/Watchlist**, and view **watch history** for personalized recommendations. | High |
| **FR-06** | The system shall allow an **Administrator** to **upload, update, or remove** **movie metadata and media files** from the content library. | High |

b)Non-Functional Requirements (NFRs)

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Attribute** | **Description** | **Impact** |
| **NFR-01** | **Performance** (Latency) | The system must return search results for 90% of queries within **2.0 seconds** under normal load. | High |
| **NFR-02** | **Security** (Integrity) | All sensitive data, particularly payment information, must be encrypted in transit (HTTPS) and at rest (AES-256). Payment processing must adhere to PCI-DSS compliance via Stripe. | Critical |
| **NFR-03** | **Reliability** (Availability) | The system must maintain a minimum of **99.9% uptime** during operational hours to ensure uninterrupted streaming availability. | Critical |
| **NFR-04** | **Usability** | The UI must be responsive and optimized for mobile form factors (iOS/Android), adhering to Material Design/Cupertino guidelines. | High |
| **NFR-05** | **Scalability** | The backend must support horizontal scaling to handle traffic surges of up to **10,000 concurrent active users** during peak events. | High |
| **NFR-06** | **Performance** (Streaming) | The video player must support **Adaptive Bitrate Streaming (ABS)** to minimize buffering under fluctuating network conditions. | Critical |
| **NFR-07** | **Security** (Authentication) | Session management must utilize secure, token-based authentication (JWT via Firebase). Protected API routes must validate tokens on every request. | Critical |
| **NFR-08** | **Maintainability** | The architecture must facilitate the integration of new third-party services (e.g., alternative Payment Gateways) without necessitating refactoring of core business logic. | Medium |
| **NFR-09** | **Performance** (Synchronization) | User data (Wishlist, History) must synchronize across devices in near real-time using Firestore listeners. | High |
| **NFR-10** | **Usability** (Offline) | The system should support offline access for downloaded content (DRM-protected local storage). | Low |

a) Architecturally Significant Requirements (ASRs)

|  |  |  |  |
| --- | --- | --- | --- |
| **ASR ID** | **Quality Attribute** | **Requirement Statement** | **Architectural Rationale** |
| **ASR-1** | **Scalability** | The architecture must support rapid horizontal scaling to accommodate sudden load spikes (1k to 10k users) during premieres. | Dictates a **Stateless Architecture** for backend services to facilitate load balancing. It also necessitates the use of a **Content Delivery Network (CDN)** for media offloading and efficient caching strategies at the Data layer. |
| **ASR-2** | **Security** | Access to protected resources (Purchases, Profile) must be strictly controlled via token validation; Payment logic must be isolated. | Requires a centralized **Security/Authorization Component** (e.g., AuthGuard) within the Business Logic layer to intercept and validate requests. Payment integration must be encapsulated to prevent leakage of sensitive data. |
| **ASR-3** | **Modifiability** | Replacing third-party integrations (Stripe, Firebase, Video Provider) must not impact the core Domain Logic or UI. | Enforces **Separation of Concerns** via a Layered Architecture. Service interfaces (e.g., IPaymentService) must be defined to decouple implementation details from the business logic, enabling independent evolution of components. |

## **3.2 Modeling Artifact: UML Use Case Diagram**

The UML Use Case Diagram below models the functional scope of the Movie Streaming Platform, delineating the **System Boundary**, **Actors** (User, Payment Gateway), and **Core Use Cases**.

* **System Boundary:** The rectangle represents the scope of the "Movie Streaming Platform" application.
* **Actors:**
  + **User:** The primary actor interacting with the application.
  + **Payment Gateway (Stripe):** A secondary system actor responsible for processing transactions.
* **Critical Relationships:**
  + **<<include>>**: The "Process Payment" use case is mandatory for "Purchase Movie."
  + **<<extend>>**: The "Stream Movie" use case is an optional extension of "Purchase Movie" (available only post-purchase).

A diagram of a movie streaming platform

AI-generated content may be incorrect.

# **4. Architectural Design (Problem Analysis for Lab 2)**

The requirements elicited in Lab 1 have a direct impact on the design challenge of Lab 2: **Layered Architecture Design**.

**4.1 The Problem Statement**

The objective is to design an architecture that satisfies core requirements while strictly enforcing **Separation of Concerns**. The **Layered Architecture** pattern is selected as the primary structural approach to address the Modifiability (ASR-3) and Security (ASR-2) requirements. This pattern will segregate the system into distinct layers: Presentation (Flutter UI), Business Logic (Services), Persistence (Repositories), and Data (Firestore/API).

**4.2 Impact of ASRs on Layered Architecture**

* **ASR-3 (Modifiability) Layered Abstraction:** ASR-3 necessitates that changes to external services (e.g., switching from Stripe to PayPal) must not propagate to the UI or Core Logic. The Layered Architecture addresses this by:
  + **Business Logic Layer:** Encapsulates integration logic (e.g., StripeService).
  + **Persistence Layer:** Abstracts data access (e.g., MovieRepository), allowing the underlying database (Firestore) to change without affecting the Business Logic, provided the repository interfaces remain consistent.
* **ASR-2 (Security) Business Logic Guardrails:** ASR-2 requires centralized authorization. In the Layered Architecture, an AuthGuard component within the Business Logic layer acts as a gatekeeper. It validates authentication tokens before any request reaches the core services (e.g., PurchaseService), ensuring consistent security policies across all entry points (UI or API).
* **ASR-1 (Scalability) Statelessness:** To support 10,000 concurrent users, the architecture must employ **Stateless Services** (relying on Firebase Auth tokens rather than server-side sessions) and optimize Data Layer interactions via caching and efficient indexing strategies.

# **5. Conclusion & Reflection**

The requirements elicitation phase for the Movie Streaming Platform has successfully defined the project scope through detailed Functional, Non-Functional, and Architecturally Significant Requirements. The UML Use Case Diagram provides a clear visual model of user interactions and the critical purchase-to-streaming workflow.

The identified ASRs—specifically those regarding **Modifiability**, **Security**, and **Scalability**—directly necessitate the adoption of a structured **Layered Architecture** for the subsequent design phase (Lab 2). This approach ensures the system can sustainably manage the complexity of payment integration, secure media delivery, and cross-platform data synchronization.