# ****Movie Catalog System - Understanding Document****

## ****1. Introduction****

This document provides a comprehensive overview of the Movie Catalog System, outlining its purpose, key components, and functionality. The system is designed to manage a movie catalog where movies can be stored, updated, rated, and associated with directors. The system also provides the ability to retrieve movies based on different criteria.

## ****2. System Overview****

The Movie Catalog System consists of three main entities: **Movies**, **Directors**, and **Ratings**. The system allows users to:

Create, update, and delete movies and directors.

Rate movies and update those ratings.

Retrieve movies based on various filters (e.g., by director, rating, etc.).

Support pagination for movie lists.

The system's backend is implemented using Java and Spring Boot, utilizing RESTful services to handle the various operations on the entities. The data is transferred through Data Transfer Objects (DTOs) such as MovieDTO, DirectorDTO, and RatingDTO.

## ****3. Key Components and Responsibilities****

### ****3.1 Movie Management****

### ****3.2 Director Management****

### ****3.3 Rating Management****

## ****5. Data Flow****

The data flow within the system occurs through interaction with the following components:

**Controller Layer**: Receives requests from the client, processes them, and calls the appropriate service method.

**Service Layer**: Contains the core business logic, calling the necessary methods in the repository or database to perform operations.

**Repository Layer**: Interacts with the database to persist and retrieve entities (such as Movie, Director, and Rating).

## ****6. Error Handling****

The system handles various errors, including:

**Invalid Input**: When an invalid MovieDTO, DirectorDTO, or RatingDTO is provided, a 400 Bad Request error is returned.

**Not Found**: When a movie, director, or rating is requested by ID and not found, a 404 Not Found error is returned.

**Server Errors**: If an unexpected error occurs on the server, a 500 Internal Server Error is returned with custom/generic message so system does not leak any stack trace.

## ****8. Future Enhancements****

To keep the development simple at this stage, we have skipped implementing security. However, in the future, we can introduce role-based access control (RBAC) or JWT-based security to protect sensitive endpoints and manage user access appropriately.

## ****9. Conclusion****

The Movie Catalog System is designed to efficiently manage movies, directors, and ratings, providing a seamless experience for users to interact with the catalog. With the service layer abstracting the core business logic, the system is easily maintainable and extendable.