Your Paper Title: Concise, Descriptive, Memorable

Nicolás Guevara Herrán
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: nguevarah@udistrital.edu.co

Samuel Antonio Sánchez
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: samasanchezp@udistrital.edu.co

***Abstract*—**

***Index Terms*—Project guidelines, IEEE LaTeX, engineering education, templates**

I. Introduction

II. Introduction

The cinema and theater sector increasingly relies on software systems to streamline daily operations such as cataloging films or plays, scheduling showtimes across multiple halls, and ensuring reliable access control for staff and customers. Audiences now expect frictionless digital experiences, while operators need tools that reduce scheduling errors, avoid underutilization of rooms, and keep operational data consistent at scale. Prior research on online ticketing and cinema management has underscored these needs by focusing on seat allocation, registration, and end-to-end booking flows in web and mobile environments.

This project presents a Cinema Management Application that automates the core back-office tasks of a cinema or theater complex. The system provides CRUD operations for films (title, genre, duration), enforcing the relationship in which one film can be scheduled in multiple functions. To align with modern engineering practices and to promote separation of concerns, the solution is implemented as two cooperating microservices exposed via REST: a Java-based authentication service built with Quarkus and backed by MySQL, complemented by Keycloak as the identity and access management provider, and a Python-based business service for catalog and scheduling CRUD backed by PostgreSQL. A lightweight web frontend consumes both APIs to support day-to-day administration.

Key challenges addressed in this work include preventing overlapping showtimes within the same hall, preserving data integrity across services and databases, and maintaining a clear audit trail for administrative actions. From an architectural viewpoint, the design emphasizes modularity, well-defined API contracts, and containerized deployment, which facilitates testing, continuous integration, and reproducible execution environments. Functionally, acceptance criteria derived from user stories guide the scope and serve as a basis for automated validation.

III. Methods and Materials

A. User Stories

User Stories: Movie Management

The following user stories define the functional requirements related to the management of movies or plays within the cinema system:

HU1 Movie/Play Registration
As an administrator, I want to register a movie or play with title, genre, and duration, so that it is available in the system.

HU2 Movie/Play Consultation
As a user, I want to view the list of available movies or plays, so that I can decide which show to attend.

HU3 Movie/Play Modification
As an administrator, I want to edit the information of a movie or play, so that the catalog remains up to date.

HU4 Movie/Play Deletion
As an administrator, I want to delete a movie or play that is no longer available, so that the database remains clean and consistent.

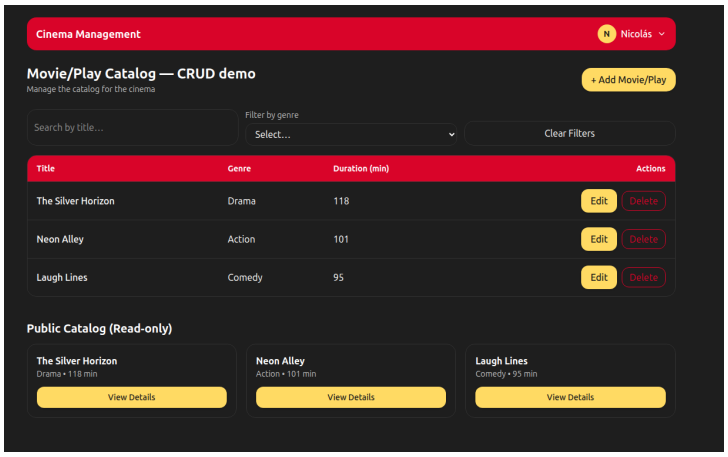

Fig. 1. Mockup for Movie Management View

B. Architecture Diagrams

Use Case Diagram

The use case diagram provides a high-level overview of the functional interactions between actors (users and administrators) and the system. It captures the scope of the project by showing which actions are supported, who performs them, and

how they are related. This representation helps to validate the user stories, to refine requirements, and to guide the design of subsequent sequence and class diagrams.

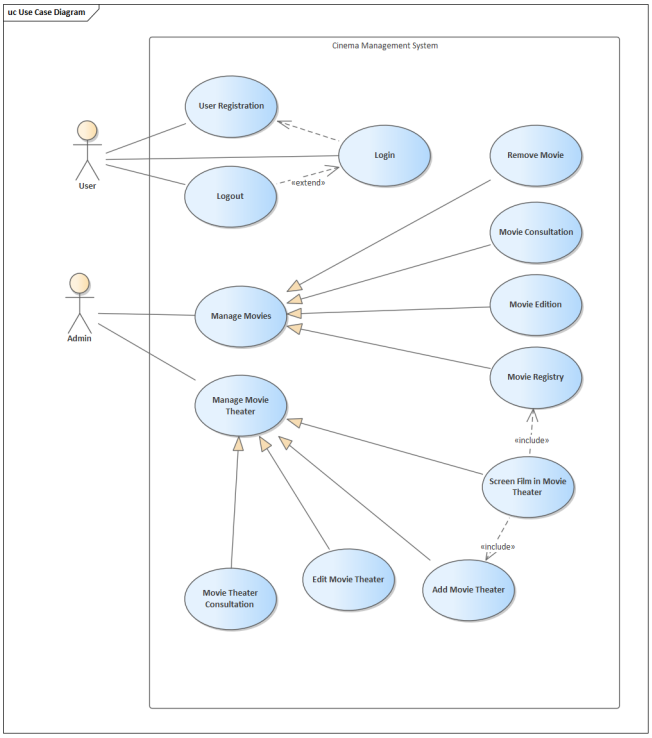

Fig. 2. Use Case Diagram of the Cinema Management System

The diagram shows two primary actors: the *User*, who can register, log in, log out, and consult available movies or functions, and the *Administrator*, who is responsible for managing the movie catalog and theater functions. The administrator can register, edit, and remove movies, as well as manage movie theaters, including adding, editing, and associating films with screenings. Relationships such as *include* and *extend* are used to indicate dependencies between cases: for example, *Screen Film in Movie Theater* includes *Movie Registry*, since a screening cannot be created without a registered movie, while *User Registration* extends *Login*, as a new user may log in immediately after registering.

IV. Results & Discussion

Tables, Figures, comparisons, statistical charts.

V. Conclusions

1-2 paragraphs. Summarize achievements, limitations, future work.

Acknowledgments

Optional: funding, collaborators.

References