<Assignment Name>

Analysis and Design Document

Student: Neagoi Mihai

**Group: 30431**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The application is a ticket selling system for shows at a festival. Cashiers can view available shows, sell tickets and view the tickets sold for a certain show. Admin users can add, update and delete shows and cashiers. Both users can authenticate.

# Functional Requirements

*Functional Requirements:*

* *Users can view a list of available shows*
* *Users can book tickets for a particular show*
* *Users can view their booking history*
* *Admin users can add new shows to the system*
* *Admin users can update show details*
* *Admin users can delete shows*
* *Admin users can add new tickets for a show*
* *Admin users can update ticket details*
* *Admin users can delete tickets*
* *The system should validate input data and display appropriate error messages*
* *The system should maintain the integrity of the data*

# Non-functional Requirements

* *The system should be fast and responsive*
* *The system should be secure and require authentication*

2. Use-Case Model

*Use-Case description format: authenticate user*

*Use case: authentication*

*Level: user-goal level*

*Primary actor: user*

*Main success scenario:*

1. *Make a form-data body for the request with email and password*
2. *Send POST request to /login endpoint*
3. *Store received cookie*

*Use-Case description format: delete a ticket*

*Use case: delete ticket*

*Level: user-goal level*

*Primary actor: cashier*

*Main success scenario:*

1. *Fill header with Cookie: \_session\_id:<received cookie upon login>*
2. *Send DELETE request to /ticket/ <ticket\_id> endpoint*

*Use-Case description format: edit a show*

*Use case: edit show*

*Level: user-goal level*

*Primary actor: admin*

*Main success scenario:*

1. *Fill body in a form-data format -> title: <Title> genre: <Genre> dateTime: <Data and time> remaining\_tickets: <number of remaining tickets>*
2. *Fill header with Cookie: \_session\_id:<received cookie upon login>*
3. *Send PUT request to /show/ <show\_id> endpoint*

3. System Architectural Design

**3.1 Architectural Pattern Description**

*The layered architectural pattern for Spring Boot is a design that separates an application into layers where each layer is responsible for a specific set of concerns. Typically, the layers are the presentation layer, business layer, persistence layer, and database layer. This pattern provides a clean separation of concerns and promotes modularity and maintainability in the application.*

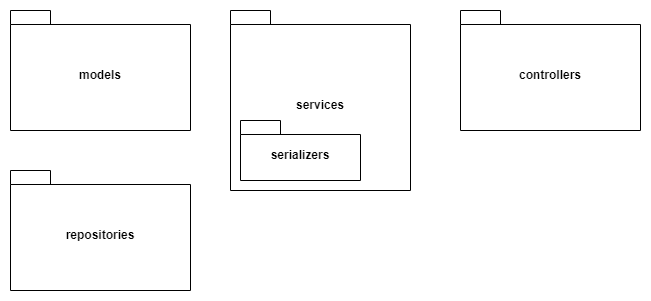
**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

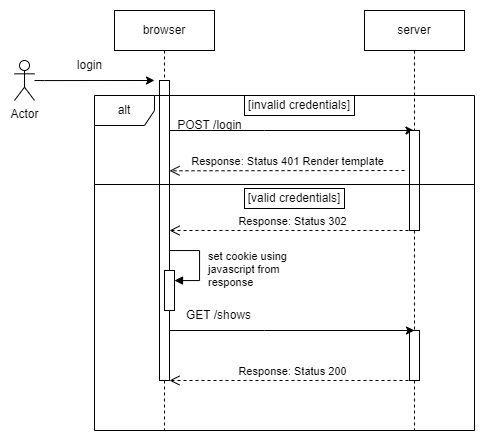
Layered Achitecture:



Package diagram:



4. UML Sequence Diagrams

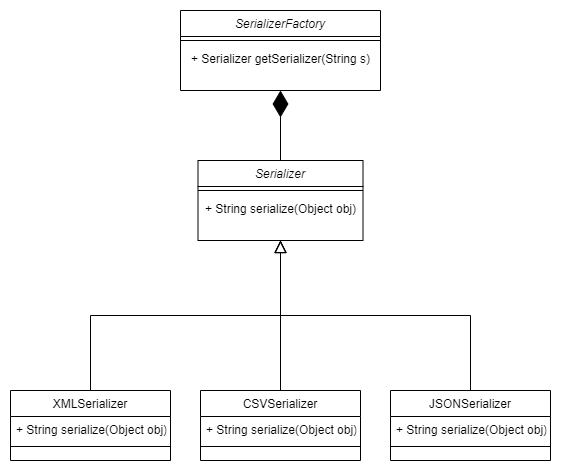


5. Class Design

**5.1 Design Patterns Description**

*The factory pattern is a creational design pattern that provides an interface for creating objects in a superclass and allows subclasses to alter the type of objects that are instantiated.*

**5.2 UML Class Diagram**



6. Data Model

*The system implements two data models using JPA: Show and Ticket. Show includes fields for title, start date and time, genre and remaining tickets. Ticket includes fields for ID, unit price, and number of places, and a many-to-one association with Show. The user data model contains the email address, password and role.*

7. System Testing

*For unit testing I used JUNIT testing framework.*

8. Bibliography

* [**https://refactoring.guru/**](https://refactoring.guru/)
* [**https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/**](https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/)
* [**https://junit.org/junit5/docs/current/user-guide/**](https://junit.org/junit5/docs/current/user-guide/)