Cinema Booking System

Introduction

This is a simple cinema booking system with separate front-end and back-end components. Built for learning purposes.

Technologies

Backend

- Java 11
- Spring Boot 2 for RESTful API
- MyBatis for database access
- Shiro, JWT for authentication and authorization
- MySQL for database
- Maven for dependency management
- Lombok for boilerplate code reduction

Frontend

- JavaScript
- Vue.js 2
- Element UI for UI components
- Axios for HTTP requests
- Vue Router for routing
- Vuex for state management
- npm for dependency management

Development Environment

- IntelliJ IDEA
- WebStorm
- Github Copilot
- Navicat
- Postman
- Git
- Docker & Docker Compose
- Node.js
- macOS

Features

- Admin Panel:
    - Authentication: login, logout based on JWT
    - Manage cinemas: crud operations

    - Manage movies: crud operations
    - Manage users: crud operations
    - Manage halls: crud operations (halls are the rooms in cinemas where movies are shown)
    - Manage sessions: crud operations (sessions are the showtimes of movies in cinemas)
    - Manage bookings/tickets: view, cancel
    - Manage roles and permissions: assign roles to users, add permissions to roles
    - Manage file uploads: upload images for movies, cinemas, halls, ...
- User Web:
    - Authentication: login, register, logout based on JWT
    - User profile: view, update
    - Movies: view list of movies, view movie details
    - Cinemas: view list of cinemas, view cinema details
    - Sessions: view list of sessions, view session details
    - Bookings: view the seat map, book tickets, cancel bookings
    - Payment: pay for bookings (not implemented yet)
    - History: view booking history of the user

Project Architecture

This project is a maven multi-module project, with the following modules:

- For backend:
    - `cinema-admin`: Backend api endpoints for cinema system including controllers.
    - `cinema-common`: Common classes and utilities shared by other modules like constants, exceptions, file uploads, ...
    - `cinema-framework`: Core classes and interfaces for cinema system including config and security.
    - `cinema-system`: Main business logic for cinema system including services, mappers and entities.
- For frontend:
    - `cinema-ui`: Admin Panel for admin users to manage cinema system.
    - `cinema-user`: Web for normal users to book tickets.

Run the Project

Prerequisites

First, clone the project:

```
git clone https://github.com/vanhung4499/cinema
```

Make sure you have the following tools installed:

- Java 11
- Maven

- Node.js 14 (use can use `nvm` or `fnm` to manage multiple versions of Node.js)
- Docker & Docker Compose

Backend

1. Start MySQL database:

I use docker compose to create a MySQL container by running the following command:

```
docker-compose up -d
```

You can run locally MySQL database instead, just make sure to update the `application-druid.yml` file in `cinema-admin` module.

2. Import the database schema and data:

You can import the database schema and data from the `cinema.sql` file in the `sql` directory of the project.

```
mysql -u root -p < sql/cinema.sql
```

3. Run the backend:

The easiest way to run the backend is to run the `CinemaAdminApplication` class in the `cinema-admin` module in IntelliJ IDEA. Otherwise, you can use maven to run the project:

```

# Clean and build all modules first
mvn clean install

# Run the admin module
cd cinema-admin
mvn spring-boot:run
```

Frontend

> The Node.js version I use is 14. You must use the same version to avoid any issues.

There are two frontend applications: `cinema-ui` and `cinema-user`. You can run them separately. The running flow is the same for both applications.

```

# Go to the frontend directory
cd cinema-ui

# Install dependencies
npm install
```

```
# Run the application
npm run serve
```

Do the same for the `cinema-user` module.

After running the frontend, you can access:

- The admin panel at `http://localhost:8888`.
- The user web at `http://localhost:8081`.

Demo

Admin Panel

- Please watch the video here.

User Web

- Please watch the video here.

> Please choose 1080p for better quality.

Deployment

TODO: Add deployment instructions.

TODO

- [ ] Handle concurrency problems when 2 users book the same seat at the same time.
- [ ] Add payment feature.
- [ ] Refactor the code to make it cleaner.
- [ ] Add deployment instructions.
- [ ] Add unit tests.

License

This project is licensed under the MIT License.

References

- Spring Boot
- Vue.js
- Element UI
- MyBatis
- Shiro
- JWT
- Lombok
- Docker
- Docker Compose

- MySQL
- Maven