Specification Document - NMS Cinema

Author: Trey Turley

1. Introduction

NMS Cinemas is a chain of single screen theatres that screen movie shows of different genres and languages at very genuine prices. They are looking to have a web application developed for online movie ticket booking.

1.1 Purpose

An online ticket booking portal will be created to support NMS Cinema. This will give customers the option of booking tickets online and will broaden NMS Cinemas reach and help them expand their business.

1.2 Intended Audience

The primary user of the application will be people (customers) looking to purchase movie tickets for shows at NMS Cinemas. There will also be a management side of the application where employees of NMS Cinemas can login and manage the available shows.

1.3 Intended Use

This application is intended to be used by customers that are looking to purchase movie tickets.

1.4 Scope

This application will allow users to view, sort, and search for movie tickets. Some of the searching and sorting options include filtering by genres and languages. They will be able to select multiple movies to purchase tickets for. There will be a cart where the user can update ticket quantities or remove tickets they do not want. When the customer is happy with their selection they will be taken through a basic checkout flow with a dummy payment gateway. There will be an user profile page to track the user's purchases.

On the admin side of the application, there will be management portal where the movies can be added or modified as needed as well as adding or removing the available genres.

1.5 GitHub

The source code for the application can be found on github at
https://github.com/TreyT-FSD/Capstone.

1.6 Amazon EC2

The application is being hosted on AWS and can be reached at http://18.117.89.47/.

The default admin credentials are:
Username: admin
Password: 12345

A sample user account can be accessed with the credentials below. This user has already made a couple purchases.
Email: test@test.com
Password: 12345

2. Overall Description

2.1 User Needs

The section below describes the two main user groups that will be using the application.

2.1.1 Customer

The customer is anyone looking to purchase movie tickets. They will be able to register and login to the site to track their purchases. They will be able to browse the shows by genre and search for shows as well. There will be a cart showing what tickets they have selected and a basic checkout process for making purchases.

2.1.2 Admin

The admin will be able to manage the genres and shows that are currently available. They will also be able to set whether a show is available or not.

2.2 Assumptions

2.2.1 Payment gateway

The checkout process will use a dummy payment gateway. The user will enter some basic details and it will be assumed that upon completing the payment steps that their payment processes successfully.

2.3 Dependencies

As this is a full stack application there will be several dependencies.

2.3.1 Front-End

The front-end will be made using the Angular framework and will utilize bootstrap to make it a rich and responsive web application.

2.3.2 Backend

Spring Boot Rest APIs will be used on the backend for handling user, admin, order, genre, and movie details.

2.3.3 Database

A MySql DB will be used to persist user, order, and movie details.

2.3.4 Devops and Testing

The applications and dependencies will be containerized using Docker and Jenkins will be used to automate the CICD process. Source code will be available on github.

2.3 Application Appearance

This will be a web application developed using Angular and bootstrap. It will use a basic layout with options for filtering, searching, and sorting the available movies.

3. System Features and Requirements

3.1 Functional Requirements

3.1.1 Movie Browsing

The user will be able to browse the available movies. Options for searching, filtering, and sorting the movies will be provided.

3.1.2 User Sign-Up and Login

Users will be able to register with the site to track their purchases. A login portal will be available for returning users.

3.1.3 Cart and Checkout

Users will be able to select tickets and add them to a cart. Once items are added to the cart the user can proceed through a dummy checkout process. An order summary is available at the completion of checkout.

3.1.4 Admin Login

The admin will be able to login to the website to manage the movies and genres. Default login credentials are:
Username: admin
Password: 12345

3.1.5 Admin Movie Management

The admin will be able to add/update movie genres and add/update movies. Movie details include movie title, description, runtime, showtime, language, number of tickets, and whether the movie is currently active and visible to the users.

3.2 External Interface Requirements

No external APIs are utilized.

3.3 System Features

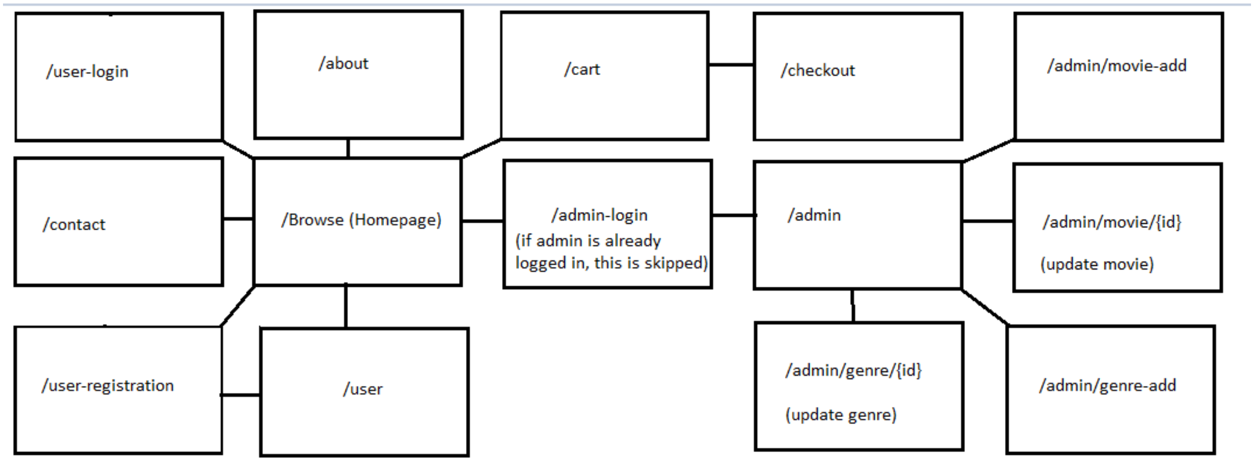
The website can be found at http://18.117.89.47/.
The SpringBoot Api is hosted at http://18.117.89.47:8085/

3.4 Non-Functional Requirements

No specifications were provided for any non-functional requirements such as SLAs for user interactions or response times. However, an effort will be made to use efficient algorithms where needed.

3.5 System Interaction Flow Chart

Users can access the application at http://18.117.89.47/. This will take the user to the main page and from there they can access the user and admin portals as outlined in the flowchart below.



4. Application Details

4.1 Algorithms

Two sorting algorithms are used on the homepage to allow the user to sort the movies. The first option is to sort the movies alphabetically by Title. The second option is to sort the movies by genre. The genres are also sorted in alphabetical order.

5. Sprint Planning

5.1 Sprint 1 (1 Week)

5.1.1 Story 1: Admin Portal

As an admin, I need to be able to login to manage the movies and genres.

5.1.1.1 Task 1: Implement admin login component

5.1.1.2 Task 2: Implement add and update movie components

5.1.1.3 Task 3: Implement add and update genre components

5.2 Sprint 2 (1 Week)

5.2.1 Story 2: User Registration/Login

As an user, I need to be able to register with the website so that my purchases are tracked. I should be able to login anytime I return to the site.

5.2.1.1 Task 4: Implement user registration component

5.2.1.2 Task 5: Implement user login component

5.2.1.3 Task 6: Implement user profile component

5.3 Sprint 3 (1 Week)

5.3.1 Story 3: User cart

As an user, I need to be able to add movie tickets to a cart before going to checkout.

5.3.1.1 Task 7: Implement cart component

5.3.2 Story 4: User checkout

As an user, I need to be to able to checkout my cart and complete a purchase.

5.3.2.1 Task 8: Implement checkout component

5.3.2.2 Task 9: Implement order component

5.3.3 Story 5: User order history

As an user, I need to be able to see my order history.

5.3.3.1 Task 10: add user order history to user profile

5.3.4 Story 6: Dynamic Data

As an admin, I want the movie and genre information I add via the admin portal to be dynamic and stored in mysql so that it is persistent. Admin credentials should be stored in the DB along with user credentials for users that have signed up.

5.3.4.1 Task 11: Implement SpringBoot Rest API for Movie

5.3.4.2 Task 12: Implement SpringBoot Rest API for Genre

5.3.4.3 Task 13: Implement SpringBoot Rest API for Order

5.3.4.4 Task 14: Implement SpringBoot Rest API for Admin

5.3.4.5 Task 15: Implement SpringBoot Rest API for User

5.4 Sprint 4 (1 Week)

5.4.1 Story 7: CICD Pipeline

As a developer, I want to use a CICD pipeline built using Jenkins and Docker to automate the build and deployment process of the application.

5.4.1.1 Task 16: Setup Jenkins pipeline that hooks into github

5.4.1.2 Task 17: Setup Dockerfile for the Angular application

5.4.1.3 Task 18: Setup Dockerfile for the SpringBoot Rest APIs

5.4.1.4 Task 19: Setup docker compose to kickoff the web app, rest apis, and mysql

5.4.2 Story 7: Amazon AWS Hosting

As a developer, I want to make the application available to external users by hosting it on an amazon EC2 instance.

5.4.2.1 Task 20: Setup EC2 instance with docker and deploy the application stack