|  |  |
| --- | --- |
| Course: | ENSF607 |
| Term: | Fall 2022 |
|  |  |
| Final Project – API Reference | |
| Version: | 1.0 |
| Date: | Nov 30 2022 |
|  |  |
|  |  |

Contents

[UML 3](#_Toc120807228)

[Relationship Diagrams 4](#_Toc120807229)

[Movie 5](#_Toc120807230)

[Supported Use Cases 5](#_Toc120807231)

[HTTP Requests 5](#_Toc120807232)

[Getting all movies 5](#_Toc120807233)

[Adding a new movie 5](#_Toc120807234)

[Deleting a movie by id 5](#_Toc120807235)

[Modifying attributes (title, showTime) of a movie by id 5](#_Toc120807236)

[RegisteredUser 6](#_Toc120807237)

[Supported Use Cases 6](#_Toc120807238)

[HTTP Requests 6](#_Toc120807239)

[Seat 7](#_Toc120807240)

[Supported Use Cases 7](#_Toc120807241)

[HTTP Requests 7](#_Toc120807242)

[Ticket 8](#_Toc120807243)

[Supported Use Cases 8](#_Toc120807244)

[HTTP Requests 8](#_Toc120807245)

# UML

Diagram

Description automatically generated

# Relationship Diagrams

Diagram

Description automatically generated

# Movie

Located under /api/v1/movie

## Supported Use Cases

1. Get all movies in JSON format
2. Add a new movie by passing a JSON object
3. Delete movie using its ID
   1. TODO: deletes all seats with this movie ID
4. Update movie using ID and parameters (title, showTime)

## HTTP Requests

### Getting all movies

<http://localhost:8080/api/v1/movie/getAll>

### Adding a new movie

### Deleting a movie by id

### Modifying attributes (title, showTime) of a movie by id

Text

Description automatically generated

# RegisteredUser

Located under /api/v1/user

## Supported Use Cases

1. Get all users in JSON format
2. Get user by ID in JSON format
3. Get user by email in JSON format
4. Register a new user by passing a JSON object
5. Update user using ID and parameters (name, email)

## HTTP Requests

1. …/api/v1/user
2. …/api/v1/user/[user id]
3. …/api/v1/user/Email/[email]
4. POST …/api/v1/user
5. PUT …/api/v1/user/[user id]

# Seat

Located under /api/v1/seat

## Supported Use Cases

1. Get all seats in JSON format
2. Get all seats from the given movie ID in JSON format
3. Add new seat by passing JSON object
4. Delete seat by ID
   1. TODO: deletes associated ticket
5. Delete all seats by movie ID
   1. TODO: deletes all associated tickets
6. Update seat by ID (movieId, seatNum, price, available)

## HTTP Requests

1. …/api/v1/seat/getAll
2. …/api/v1/seat/movie/[movie id]
3. POST …/api/v1/seat/add
4. DELETE …/api/v1/seat/[seat id]
5. DELETE …/api/v1/seat/movie/[movie id]
6. PUT …/api/v1/seat/[seat id]?[param]=[value]

# Ticket

Located under /api/v1/ticket

## Supported Use Cases

1. Get all tickets in JSON format
2. Get ticket by ID in JSON format
3. Get ticket by seat ID in JSON format
4. Get ticket by buyer name in JSON format
5. Delete ticket by ID
6. Delete ticket by seat ID
7. Add a new ticket by passing JSON object

## HTTP Requests

1. …/api/v1/ticket/getAll
2. …/api/v1/ticket/[ticket id]
3. …/api/v1/ticket/seat/[seat id]
4. …/api/v1/ticket/buyer/[buyer name]
5. DELETE …/api/v1/ticket/[ticket id]
6. DELETE …/api/v1/ticket/seat/[seat id]
7. POST …/api/v1/ticket/add