Algorithm:

DESCRIPTION

Create a dynamic and responsive web application for booking movie tickets online for different genres and languages.

Background of the problem statement:
NMS Cinemas is a chain of single screen theatres that screen movie shows of different genres and languages at very genuine prices. It was established in 2004 in Pune, India. Recently, the business analysts noticed a decline in sales since 2010. They found out that the online booking of movie tickets from apps, such as BookMyShow and Paytm were gaining more profit by eliminating middlemen from the equation. As a result, the team decided to hire a Full Stack developer to develop an online movie ticket booking web application with a rich and user-friendly interface.
You are hired as the Full Stack Java developer and are asked to develop the web application. The management team has provided you with the requirements and their business model so that you can easily arrange different components of the application.

Features of the application:

1. Registration
2. Login
3. Payment gateway
4. Searching
5. Filtering
6. Sorting
7. Dynamic data
8. Responsive and compatible with different devices

Recommended technologies:

1. Database management: MySQL and Oracle
2. Backend logic: Java programming, NodeJS
3. Frontend development: JSP, Angular, Bootstrap, HTML/CSS, and Javascript
4. Automation and testing technologies: Selenium, Jasmine, and TestNG
5. DevOps and production technologies: Git, GitHub, Jenkins, Docker, Kubernetes, and AWS

Project development guidelines:

- The project will be delivered within four sprints with every sprint delivering a minimal viable product.
- It is mandatory to perform proper sprint planning with user stories to develop all the components of the project.
- The learner can use any technology from the above-mentioned technologies for different layers of the project.
- The web application should be responsive and should fetch or send data dynamically without hardcoded values.
- The learner must maintain the version of the application over GitHub and every new change should be sent to the repository.
- The learner must implement a CI/CD pipeline using Jenkins.

- The learner should also deploy and host the application on an AWS EC2 instance.
- The learner should also implement automation testing before the application enters the CI/CD pipeline.
- The learner should use Git branching to do basic automation testing of the application in it separately.
- The learner should make a rich frontend of the application, which is user- friendly and easy for the user to navigate through the application.
- There will be two portals in the application, namely admin and user portal.

Admin Portal:
It deals with all the backend data generation and product information. The admin user should be able to:

- Add or remove different genres to or from the application to build a rich product line
- Edit movie details like name, ticket price, language, description, and show timings to keep it aligned to the current prices
- Enable or disable the movie shows from the application

User Portal:
It deals with the user activities. The end-user should be able to:

- Sign-in to the application to maintain a record of activities
- Search for movie tickets based on the search keyword
- Apply filters and sort results based on different genres
- Add all the selected movie tickets to a cart and customize the purchase at the end
- Experience a seamless payment process
- Receive a booking summary page once the payment is complete

Source Code:

FRONT END:

About us:

```
<div class="about-us-container">
  <div class="about-us-wrapper postion-relative">
    <div class="about-us-banner">
      <div class="about-us-banner-text h-100">
        <h1 class="fs-1">
          {{ title }}
        </h1>
        <h2 class="fs-2 text-uppercase">WE provide entertainment</h2>
      </div>
    </div>
  </div>

  <div class="description mt-4">
    <div class="mx-4">
      <h1 class="fs-2">About Us</h1>
      <div class="underline text-warning"></div>
    </div>
    <p class="text-muted text-start px-3 p-lg-4 p-md-3">
      22 years ago in South Africa a seed of an idea was planted, a dream was
      shared. Inception happened. 22 years on, we look back at what we've
built.
      Leave it up to us, and we'd love to do it all over again. Here's our
story
    </p>
  </div>

  <div class="testimonials p-lg-5 bg-dark text-white p-md-3 mt-4">
    <h1 class="fs-2 pb-5 text-center">Our Happy Customer</h1>
    <div class="testimonials-list row g-0">
      <div class="testimonial-card col-4 px-4 my-4">
        <div class="testimonial text-center">
          <div class="image mb-2 col-3 mx-auto">
            <img
              mat-card-image=""
              src="https://material.angular.io/assets/img/examples/shiba2.jpg"
              alt="Photo of a Shiba Inu"
              class="mat-card-image rounded-circle"
              height="120"
              width="120"
            />
          </div>
          <div class="comments">
            <h3 class="fs-3 user">User 1</h3>
            <div class="text-muted text-start">
              22 years ago in South Africa a seed of an idea was planted, a
              dream was shared. Inception happened. 22 years on, we look back
at

              what we've built. Leave it up to us, and we'd love to do it all
              over again. Here's our story
            </div>
          </div>
        </div>
      </div>

      <div class="testimonial-card col-4 px-4 my-4">
        <div class="testimonial text-center">
          <div class="image mb-2 col-3 mx-auto">
            <img
              mat-card-image=""
              src="https://material.angular.io/assets/img/examples/shiba2.jpg"
              alt="Photo of a Shiba Inu"
              class="mat-card-image rounded-circle"
              height="120"
```

```html
              width="120"
            />
          </div>
          <div class="comments">
            <h3 class="fs-3 user">User 2</h3>
            <div class="text-muted text-start px-3">
              22 years ago in South Africa a seed of an idea was planted, a
              dream was shared. Inception happened. 22 years on, we look back
at

              what we've built. Leave it up to us, and we'd love to do it all
              over again. Here's our story
            </div>
          </div>
        </div>
      </div>
      <div class="testimonial-card col-4 px-4 my-4">
        <div class="testimonial text-center">
          <div class="image mb-2 col-3 mx-auto">
            <img
              mat-card-image=""
              src="https://material.angular.io/assets/img/examples/shiba2.jpg"
              alt="Photo of a Shiba Inu"
              class="mat-card-image rounded-circle border-1 border-info"
              height="120"
              width="120"
            />
          </div>
          <div class="comments">
            <h3 class="fs-3 user">User 3</h3>
            <div class="text-muted text-start px-3">
              22 years ago in South Africa a seed of an idea was planted, a
              dream was shared. Inception happened. 22 years on, we look back
at

              what we've built. Leave it up to us, and we'd love to do it all
              over again. Here's our story
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>

  <div class="branches mt-4 py-5">
    <div class="mx-4">
      <h1 class="fs-2">Our Branches</h1>
      <div class="underline"></div>
    </div>
    <div class="row g-0 justify-content-around align-content-center">
```

```
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Chennai</mat-card
      >
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Bangalore</mat-card
      >
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Coimbatore</mat-card
      >
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Hyderabad</mat-card
      >
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Goa</mat-card
      >
      <mat-card class="col-3 mx-3 my-3 bg-warning" color="accent"
        >Kochi</mat-card
      >
    </div>
  </div>
</div>
```

Auditorium Forms:

```
<div class="form-container">
  <div class="mx-auto text-center bg-color">
    <div class="container py-5">
      <h1 class="fs-2 text-dark py-5 text-uppercase">
        Register New Auditorium Here
      </h1>
    </div>
  </div>

  <div class="form-holder border border-4 border-info rounded mx-5">
    <form [formGroup]="auditoriumForm" (ngSubmit)="onSubmit()">
      <mat-vertical-stepper [linear]="true" #stepper>
        <mat-step [stepControl]="auditoriumForm">
          <ng-template matStepLabel>Auditorium Details</ng-template>
          <div class="row g-0 my-3 justify-content-around">
            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Name</mat-label>
                <input
```

```
                  matInput
                  placeholder="Auditorium Name"
                  formControlName="name"
                />
                <mat-error *ngIf="nameErrors">{{ nameErrors }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Image URL</mat-label>
                <input
                  matInput
                  placeholder="Image Link"
                  formControlName="image"
                />
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Seating Capacity</mat-label>
                <input matInput formControlName="seatCapacity" readonly />
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Email Id</mat-label>
                <input
                  matInput
                  placeholder="Email Id"
                  formControlName="email"
                />
                <mat-error *ngIf="emailErrors">{{ emailErrors }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
```

```
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Customer Care Number</mat-label>
                <input
                  matInput
                  placeholder="Cust Care No"
                  formControlName="customerCareNo"
                />
                <mat-error *ngIf="customerCareNoErrors">{{
                  customerCareNoErrors
                }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Address</mat-label>
                <input
                  matInput
                  placeholder="Address"
                  formControlName="address"
                />
                <mat-error *ngIf="addressErrors">{{ addressErrors }}</mat-
error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step [stepControl]="auditoriumForm.get('safeties')!" optional>
          <ng-template matStepLabel>Add Safeties</ng-template>
          <div class="row g-0 justify-content-around">
            <div
              class="col-sm-12 col-lg-5 form-field"
              formArrayName="safeties"
              *ngFor="let safety of safeties.controls; index as i"
            >
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Safety</mat-label>
                <input
                  matInput
                  placeholder="Hand Sanitizers Available"
                  [formControlName]="i"
                  [attr.list]="'safety-' + i"
                />
                <datalist [id]="'safety-' + i">
                  <option [value]="safety" *ngFor="let safety of allSafeties">
                    {{ safety }}
                  </option>
                </datalist>

                <mat-error *ngIf="safeties.touched">
                  <span *ngIf="safety.errors?.required"
                    >Safety cannot be empty</span
                  >
                  <span *ngIf="safety.errors?.uniqueName"
                    >Safety already added</span
                  >
                </mat-error>
              </mat-form-field>
              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeSafety(i)"
                aria-hidden="false"
                matTooltip="Remove the Safety"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more safety"
              aria-label="add_circle icon"
              (click)="addSafety()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>
```

```
<mat-step [stepControl]="auditoriumForm.get('facilities')!" optional>
  <ng-template matStepLabel>Add Facilities</ng-template>
  <div class="row g-0 justify-content-around">
    <div
      class="col-sm-12 col-lg-5 form-field"
      formArrayName="facilities"
      *ngFor="let facility of facilities.controls; index as i"
    >
      <mat-form-field
        class="d-block w-100 text-dark"
        appearance="outline"
      >
        <mat-label>Facility</mat-label>
        <input
          matInput
          placeholder="Parking Facility"
          [formControlName]="i"
          [attr.list]="'facility-' + i"
        />
        <datalist [id]="'facility-' + i">
          <option
            [value]="facility"
            *ngFor="let facility of allFacilities"
          >
            {{ facility }}
          </option>
        </datalist>

        <mat-error *ngIf="facilities.touched">
          <span *ngIf="facility.errors?.required"
            >Facility cannot be empty</span
          >
          <span *ngIf="facility.errors?.uniqueName"
            >Facility already added</span
          >
        </mat-error>
      </mat-form-field>
      <mat-icon
        class="fs-2 w-auto text-danger delete-icon"
        (click)="removeFacility(i)"
        aria-hidden="false"
        matTooltip="Remove the Facility"
        aria-label="delete icon"
        >delete</mat-icon
      >
    </div>
  </div>
```

```html
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more facility"
              aria-label="add_circle icon"
              (click)="addFacility()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step [stepControl]="auditoriumForm.get('shows')!" optional>
          <ng-template matStepLabel>Add Shows</ng-template>
          <div
            class="w-100"
            formArrayName="shows"
            *ngFor="let show of shows.controls; index as i"
          >
            <div
              class="
                row
                g-0
                justify-content-around
                align-content-center
                form-field
              "
              [formGroupName]="i"
            >
              <div class="col-lg-5 col-sm-12">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Show Name</mat-label>
                  <input
                    matInput
                    formControlName="name"
                    [attr.list]="'show-name-' + i"
                  />
                  <datalist [id]="'show-name-' + i">
                    <option *ngFor="let show of showNames" [value]="show">
                      {{ show }}
                    </option>
                  </datalist>
                  <mat-error *ngIf="shows.touched">
                    <span *ngIf="show.get('name')?.errors?.required"
                      >Show Name cannot be empty</span
                    >
                    <span *ngIf="show.get('name')?.errors?.uniqueName"
                      >Show Name already added</span
                    >
                  </mat-error>
                </mat-form-field>
              </div>
              <div class="col-lg-5 col-sm-12">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Show Timing</mat-label>
                  <input
                    type="time"
                    matInput
                    placeholder="Matnee"
                    formControlName="startTime"
                  />
                  <mat-error *ngIf="shows.touched">
                    <span *ngIf="show.get('startTime')?.errors?.required">
                      Show Start Time cannot be empty</span
                    >
                    <span *ngIf="show.get('startTime')?.errors?.uniqueTime">
                      Time gap between the shows must be at-least 3
hours</span
                    >
                  </mat-error>
                </mat-form-field>
              </div>
              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeShow(i)"
                aria-hidden="false"
                matTooltip="Remove the Show"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more facility"
              aria-label="add_circle icon"
              (click)="addShow()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step>
          <ng-template matStepLabel>Review</ng-template>
          <pre>
            {{ auditoriumForm.value | json }}
          </pre>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button
              mat-button
              color="primary"
              [disabled]="!auditoriumForm.valid"
            >
              Submit
            </button>
          </div>
        </mat-step>
      </mat-vertical-stepper>
    </form>
  </div>
</div>
```

Contact Us:

```
<div class="contact-us-container">
  <div class="contact-us-wrapper postion-relative">
    <div class="contact-us-banner">
      <div class="contact-us-banner-text h-100">
        <h1 class="fs-1">Contact Us</h1>
        <h2 class="fs-2 text-uppercase">Drop us a line, get in touch</h2>
      </div>
    </div>
  </div>

  <div class="get-in-touch my-5">
```

```html
    <div class="row g-0 justify-content-around">
      <div class="write-to-us col-sm-12 col-md-12 col-lg-7">
        <h1 class="fs-2">Write to us by filling in the form below</h1>
        <div class="underline"></div>
        <div class="form-container">
          <form class="example-form col-sm-12 col-lg-8 mx-auto">
            <mat-form-field
              class="example-full-width d-block text-dark"
              appearance="standard"
            >
              <mat-label class="text-dark">Your Name</mat-label>
              <input type="text" matInput class="fs-5 text-dark" />
            </mat-form-field>

            <mat-form-field
              class="example-full-width d-block w-100 text-dark"
              appearance="standard"
            >
              <mat-label class="text-dark">Email Id</mat-label>
              <input type="email" matInput class="fs-5 text-dark" />
            </mat-form-field>

            <mat-form-field
              class="example-full-width d-block w-100 text-dark"
              appearance="standard"
            >
              <mat-label class="text-dark">Message</mat-label>
              <textarea matInput placeholder="How can we help
you!"></textarea>
            </mat-form-field>

            <button mat-raised-button color="accent" type="submit">
              Contact Us
            </button>
          </form>
        </div>
      </div>
      <div class="address col-sm-12 col-md-12 col-lg-4">
        <h1 class="fs-2">Contact Us</h1>
        <div class="underline text-success"></div>

        <div class="contact-address">
          <mat-list>
            <mat-list-item role="listitem" class="py-2 h-auto">
              <mat-icon class="d-inline-block"> place </mat-icon>
              <div class="address ms-3 text-muted col-10">
                SPI Network Limited, 25, Mamatha Complex, 5th Floor, Whites
Rd, Royapettah, Chennai, Tamil Nadu 600014.
```

```
              </div>
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item role="listitem" class="py-2 h-auto">
              <mat-icon class="d-inline-block"> email </mat-icon>
              <div class="address ms-3 text-muted col-
10">ABC@cinemas.com</div>
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item role="listitem" class="py-2 h-auto">
              <mat-icon class="d-inline-block col-1"> support_agent </mat-
icon>
              <div class="address ms-3 text-muted col-10"> 1800-894-4059</div>

            </mat-list-item>
          </mat-list>
        </div>
      </div>
    </div>
  </div>
</div>
```

Footer:

```
<footer
  *ngIf="router.url !== '/select-seats'"
  class="row w-100 justify-content-around mx-0 footer"
  [ngClass]="fixed ? 'fixed-bottom' : ''"
>
  <div
    class="text-center text-white"
    style="height: 50px; line-height: 50px"
    [ngClass]="{
      'bg-dark': !bgColor,
      bgColor: bgColor,
      'text-white': !textColor,
      textColor: textColor
    }"
  >
    Designed & Developed by
    <a
      target="_blank"
      class="text-decoration-none mx-2"
      ><strong>Abishek S</strong></a
    >
  </div>
</footer>
```

Forgot Password:

```html
<main class="px-3 my-auto">
  <p
    class="lead mb-2 p-2 rounded"
    [ngClass]="(alertDanger$ | async) ? 'alert-danger' : 'alert-success'"
    *ngIf="showAlert$ | async"
  >
    {{ alertMessage$ | async }}
  </p>
  <h1 class="fs-2 py-4">Change Password Here</h1>

  <div class="example-container">
    <form
      #passwordForm
      class="row w-100 g-0"
      [formGroup]="forgotPasswordForm"
      (ngSubmit)="onSubmit()"
    >
      <div class="col-12 mb-3">
        <mat-form-field appearance="outline" class="text-white d-block">
          <mat-label>Enter your username</mat-label>
          <input
            matInput
            type="text"
            placeholder="ABCbranch@cinemas.com"
            formControlName="username"
            class="fs-5"
          />
          <mat-hint *ngIf="username.pending" class="text-warning"
            >Checking for username...</mat-hint
          >
          <mat-error *ngIf="usernameErrors">{{ usernameErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-12 mb-3">
        <mat-form-field appearance="outline" class="text-white d-block">
          <mat-label>Enter your password</mat-label>
          <input
            matInput
            [type]="hidePassword ? 'password' : 'text'"
            formControlName="password"
            class="fs-5"
          />
          <mat-error *ngIf="passwordErrors">{{ passwordErrors }}</mat-error>
          <button
            mat-icon-button
            matSuffix
            (click)="hidePassword = !hidePassword"
            [attr.aria-label]="'Hide password'"
            [attr.aria-pressed]="hidePassword"
          >
            <mat-icon>{{
              hidePassword ? "visibility_off" : "visibility"
            }}</mat-icon>
          </button>
        </mat-form-field>
      </div>

      <button
        mat-raised-button
        [color]="!forgotPasswordForm.valid ? 'accent' : 'primary'"
        class="col-12"
        type="submit"
      >
        Change Password
      </button>
    </form>
  </div>
</main>
```

Header:

```html
<nav
  *ngIf="router.url !== '/select-seats'"
  id="nav-bar"
  class="navbar navbar-expand-lg navbar-dark bg-show text-white px-lg-3"
>
  <div class="container-fluid">
    <a mat-button class="navbar-brand ms-sm-3 fs-4" routerLink="/home">{{
      title
    }}</a>
    <button
      class="navbar-toggler"
      mat-icon-button
      aria-label="Example icon-button with a menu"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <mat-icon>more_vert</mat-icon>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0 text-sm-center">
        <li class="nav-item mx-lg-2 mx-md-auto">
          <a
            mat-button
            class="nav-link"
            aria-current="page"
            routerLinkActive="active"
            routerLink="/home"
            >Home</a
          >
        </li>
        <li class="nav-item mx-lg-2 mx-md-auto">
          <a
            mat-button
            class="nav-link"
            routerLinkActive="active"
            [routerLink]="['/movies']"
            >Movies</a
          >
        </li>
        <li class="nav-item mx-lg-2 mx-md-auto">
          <a
            mat-button
            class="nav-link"
            routerLinkActive="active"
            routerLink="/about"
            >About Us</a
          >
        </li>
        <li class="nav-item mx-lg-2 mx-md-auto">
          <a
            mat-button
            class="nav-link"
            routerLinkActive="active"
            routerLink="/contact"
            >Contact Us</a
          >
        </li>
        <li class="nav-item dropdown" *ngIf="service.isAdmin()">
          <a
            class="nav-link dropdown-toggle"
            href="#"
            id="admin-panel"
            role="button"
            data-bs-toggle="dropdown"
            aria-expanded="false"
          >
            Admin Panel
          </a>
          <ul class="dropdown-menu" aria-labelledby="admin-panel">
            <li class="nav-item mx-lg-2 mx-md-auto">
              <a
                class="dropdown-item"
                href="#"
                routerLinkActive="active"
                routerLink="/admin/add-auditorium"
                >Add Auditorium</a
              >
            </li>
            <li class="nav-item mx-lg-2 mx-md-auto">
              <a
                class="dropdown-item"
                href="#"
                routerLinkActive="active"
                routerLink="/admin/add-movie"
                >Add Movie</a
              >
            </li>
            <li class="nav-item mx-lg-2 mx-md-auto">
              <a
                class="dropdown-item"
                href="#"
                routerLinkActive="active"
                routerLink="/admin/manage"
                >Manage</a
              >
            </li>
          </ul>
        </li>
      </ul>
      <div class="profile text-sm-center">
        <!-- <button mat-button [matMenuTriggerFor]="menu">
          <img
            class="rounded-circle mx-2"
            src="https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSK9DmKOF9VCBqRzdHLzwyyOXjDm4kOYjD9IZQo2Wvi
iLzzxDmzwYQZaVK6_7o_Qc7famc&usqp=CAU"
            alt=""
            width="45"
            height="45"
          />
          My Account
        </button>
        <mat-menu #menu="matMenu">
          <a routerLink="/user/login" mat-menu-item>Login</a>
          <a routerLink="/user/register" mat-menu-item>Register</a>
          <a routerLink="/user/forgot-password" mat-menu-item>
            Forgot Password
          </a>
          <a routerLink="/user/profile" mat-menu-item>My Profile</a>
          <a routerLink="/user/bookings" mat-menu-item>My Bookings</a>
          <a routerLink="/user/logout" mat-menu-item>Logout</a>
        </mat-menu> -->
        <ul class="navbar-nav col-auto" *ngIf="service.isLoggedIn()">
          <li class="nav-item dropdown">
            <a
              class="nav-link dropdown-toggle fs-6 my-2"
              href="#"
              id="account-links"
              role="button"
              data-bs-toggle="dropdown"
              aria-expanded="false"
              >{{ service.getUser()?.email! }}
            </a>
            <ul
              class="dropdown-menu dropdown-menu-dark"
              aria-labelledby="account-links"
            >
              <li class="nav-item">
                <a class="dropdown-item py-1" routerLink="/my/profile"
                  >Profile</a
                >
              </li>
              <li class="nav-item">
                <a class="dropdown-item py-1" routerLink="/my/bookings"
                  >Bookings</a
                >
              </li>
              <li class="nav-item text-center">
                <a
                  class="btn btn-outline-warning btn-sm col-11 mt-2"
                  (click)="onLogout()"
                  >Logout</a
                >
              </li>
            </ul>
          </li>
        </ul>
```

```
        <div class="btn-group p-2" *ngIf="!service.isLoggedIn()">
          <a
            routerLink="/user/login"
            routerLinkActive="active"
            class="btn btn-primary btn-sm rounded mx-1"
            >Login</a
          >
          <a
            routerLink="/user/register"
            routerLinkActive="active"
            class="btn btn-success btn-sm rounded mx-1"
            >Register</a
          >
        </div>
      </div>
    </div>
  </div>
</nav>
```

Home Page:

```
<div class="container-fluid my-2 pb-5">
  <h1
    class="text-center text-warning h-100vh mx-auto"
    *ngIf="(carousel$ | async)?.length! < 1"
  >
    Movie List is empty
  </h1>
  <ng-container *ngIf="(carousel$ | async)?.length! > 0">
    <div id="movies-carousel" class="carousel slide" data-bs-ride="carousel">
      <div class="carousel-indicators">
        <button
          [ngClass]="i == 0 ? 'active' : ''"
          [attr.aria-current]="i == 0 ? true : false"
          *ngFor="let movie of carousel$ | async; index as i"
          type="button"
          data-bs-target="#movies-carousel"
          [attr.data-bs-slide-to]="i"
          [attr.aria-label]="'Slide ' + i"
        ></button>
      </div>

      <div class="carousel-inner">
        <div
          class="carousel-item"
          *ngFor="let movie of carousel$ | async; index as i"
          [ngClass]="i == 0 ? 'active' : ''"
        >
          <img
            [src]="movie.bgImage"
            class="d-block w-100"
            height="550"
            [alt]="movie.name"
          />
          <div class="carousel-caption d-none d-md-block">
            <h5 class="fs-1 mb-3">{{ movie.name | uppercase }}</h5>
            <p class="lead">
              {{
                (movie.language | titlecase) +
                  " | " +
                  (movie.genres?.join(" - ") | titlecase) +
                  " | " +
                  movie.year
              }}
            </p>
          </div>
        </div>
      </div>
      <button
        class="carousel-control-prev"
        type="button"
        data-bs-target="#movies-carousel"
        data-bs-slide="prev"
      >
        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Previous</span>
      </button>
      <button
        class="carousel-control-next"
        type="button"
        data-bs-target="#movies-carousel"
        data-bs-slide="next"
      >
        <span class="carousel-control-next-icon" aria-hidden="true"></span>
        <span class="visually-hidden">Next</span>
      </button>
    </div>

    <section
      *ngIf="(nowPlaying$ | async)?.length! > 0"
      class="now-showing my-5 bg-dark text-white p-lg-5"
    >
      <div class="row g-0 mb-5 justify-content-around">
        <h3 class="text-uppercase fs-2 col-11">Now Playing</h3>
        <a
          routerLink="../movies"
          [queryParams]="{ show: 'now-playing' }"
          class="d-block col-1 text-end"
          >view all</a
        >
      </div>
      <div class="now-showing-movies">
        <div class="card-group row g-0 justify-content-around">
          <div
            class="col-sm-11 col-md-5 col-lg-3 col-xxl-2 px-2 position-
relative"
            *ngFor="let movie of nowPlaying$ | async"
          >
            <div class="image-card postion-relative rounded-5 overflow-
hidden">
              <img
                [src]="movie.image"
                class="card-img-top"
                height="400"
                width="284"
                [alt]="movie.name"
              />
              <div
                class="
                  image-release
                  postion-absolute
                  bottom-0
                  start-0
                  bg-white
                  text-dark
                  w-100
                  text-center
                  py-2
                "
              >
                {{ formatRelease(movie.release) }}
              </div>
            </div>
            <div class="card-body text-center">
              <h2 class="card-title fs-4 mb-0">
                <a
                  class="stretched-link"
                  routerLink="../movie/{{ movie.id }}"
                  >{{ movie.name | uppercase }}</a
                >
              </h2>
              <h3 class="card-text text-muted">
```

```
                {{ movie.language | titlecase }}
              </h3>
            </div>
          </div>
        </div>
      </div>
    </section>

    <section
      *ngIf="(upComing$ | async)?.length! > 0"
      class="up-coming my-5 p-lg-5"
    >
      <div class="row g-0 mb-5 justify-content-around">
        <h3 class="text-uppercase fs-2 col-11">Up Coming</h3>
        <a
          routerLink="../movies"
          [queryParams]="{ show: 'up-coming' }"
          class="d-block col-1 text-end"
          >view all</a
        >
      </div>
      <div class="up-coming-movies">
        <div class="card-group row g-0 justify-content-around">
          <div
            class="col-sm-11 col-md-5 col-lg-3 col-xxl-2 px-2 position-
relative"
            *ngFor="let movie of upComing$ | async"
          >
            <div class="image-card postion-relative rounded-5 overflow-
hidden">
              <img
                [src]="movie.image"
                class="card-img-top"
                height="400"
                width="284"
                [alt]="movie.name"
              />
              <div
                class="
                  image-release
                  postion-absolute
                  bottom-0
                  start-0
                  bg-dark
                  text-white
                  w-100
                  text-center
                  py-2
                "
              >
                {{ formatRelease(movie.release) }}
              </div>
            </div>
            <div class="card-body text-center">
              <h2 class="card-title fs-4 mb-0">
                <a
                  class="stretched-link"
                  routerLink="../movie/{{ movie.id }}"
                  >{{ movie.name | uppercase }}</a
                >
              </h2>
              <h3 class="card-text text-muted">
                {{ movie.language | titlecase }}
              </h3>
            </div>
          </div>
        </div>
      </div>
    </section>
  </ng-container>
</div>
```

Layout:

```
<app-header></app-header>
<router-outlet></router-outlet>
<app-footer></app-footer>
```

Log In :

```
<main class="px-3 my-auto">
  <p
    class="lead mb-2 p-2 rounded"
    [ngClass]="(alertDanger$ | async) ? 'alert-danger' : 'alert-success'"
    *ngIf="showAlert$ | async"
  >
    {{ alertMessage$ | async }}
  </p>
  <h1 class="fs-2 py-4">Login Page</h1>

  <div class="example-container">
    <form
```

```
      action="#"
      class="row w-100 g-0"
      [formGroup]="loginForm"
      (ngSubmit)="onSubmit()"
    >
      <div class="col-12 mb-3">
        <mat-form-field appearance="outline" class="text-white d-block">
          <mat-label>Enter your username</mat-label>
          <input matInput type="text" formControlName="username" class="fs-5"
/>
          <mat-error *ngIf="usernameErrors">{{ usernameErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-12 mb-3">
        <mat-form-field appearance="outline" class="text-white d-block">
          <mat-label>Enter your password</mat-label>
          <input
            matInput
            [type]="hidePassword ? 'password' : 'text'"
            formControlName="password"
            class="fs-5"
          />
          <mat-error *ngIf="passwordErrors">{{ passwordErrors }}</mat-error>
          <button
            mat-icon-button
            matSuffix
            (click)="hidePassword = !hidePassword"
            [attr.aria-label]="'Hide password'"
            [attr.aria-pressed]="hidePassword"
          >
            <mat-icon>{{
              hidePassword ? "visibility_off" : "visibility"
            }}</mat-icon>
          </button>
        </mat-form-field>
      </div>

      <button
        mat-raised-button
        class="col-12"
        type="submit"
        [color]="!loginForm.valid ? 'accent' : 'primary'"
      >
        Login
      </button>
    </form>
  </div>
```

```
</main>
```

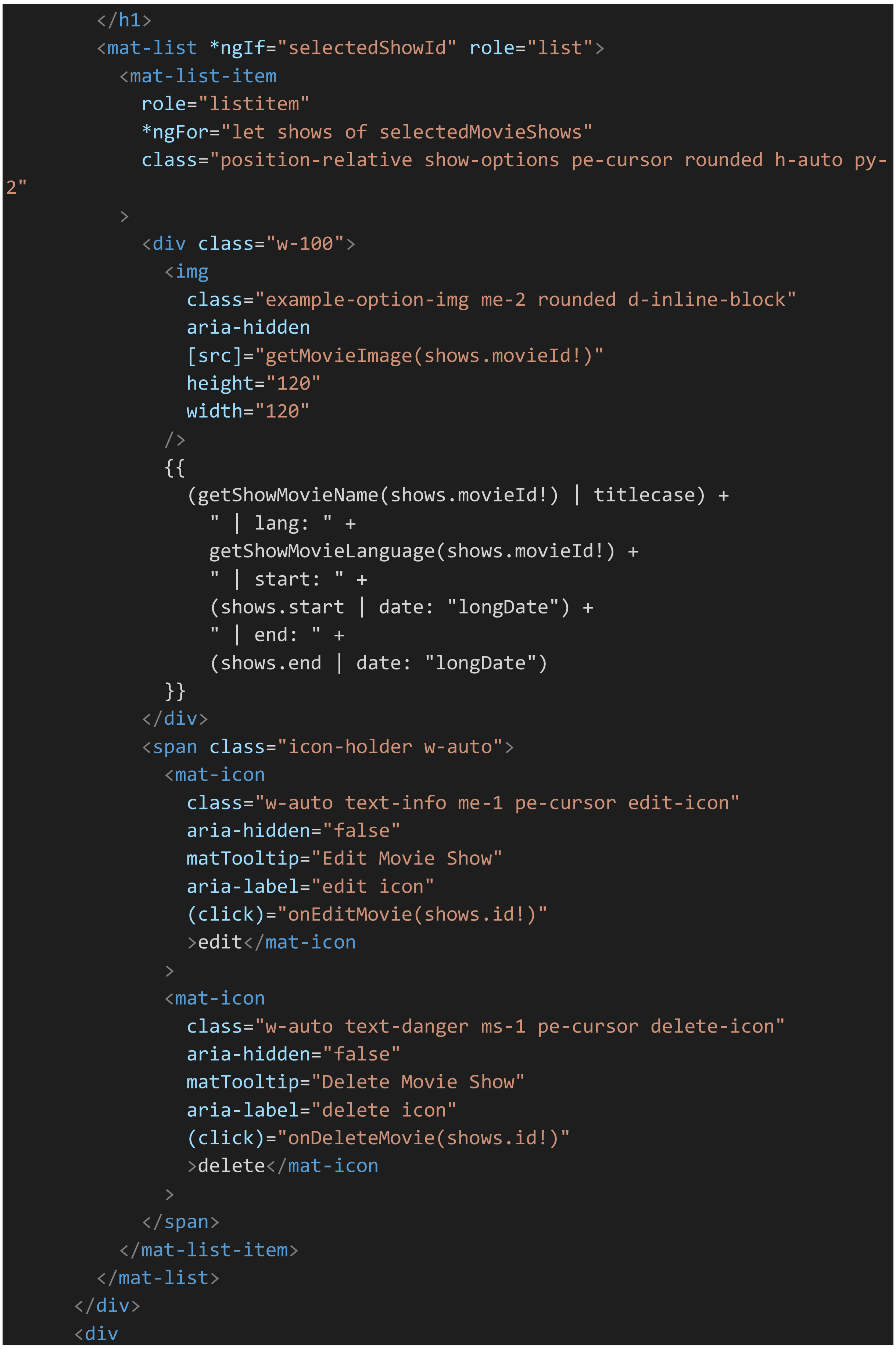
LogIn LayOut:

```
<div class="d-flex h-100 text-center text-white bg-dark cover-page mb-5">
  <div class="cover-container d-flex w-100 p-3 mx-auto flex-column">
    <header class="mb-5">
      <div class="clearfix">
        <h3 class="float-md-start mb-0 fs-4 text-md-center">{{ title }}</h3>
        <nav class="nav nav-masthead justify-content-center float-md-end">
          <a class="nav-link" aria-current="page" routerLink="/home">Home</a>
          <a class="nav-link" routerLinkActive="active"
routerLink="/user/login"
            >Login</a
          >
          <a
            class="nav-link"
            routerLinkActive="active"
            routerLink="/user/register"
            >Register</a
          >
          <a
            class="nav-link"
            routerLinkActive="active"
            routerLink="/user/forgot-password"
            >Forgot Password</a
          >
        </nav>
      </div>
    </header>

    <router-outlet></router-outlet>
  </div>
</div>

<app-footer [fixed]="true"></app-footer>

<!-- [fixed]="true"
  bgColor="bg-white"
  textColor="text-dark" -->
```

Manage:

```
<div class="wrapper w-100 my-2">
```

```
<div class="container-fluid row g-0 h-90vh">
  <div class="col-lg-3 px-2 border-3 border-end row g-0">
    <h2
      class="
        py-2
        rounded
        text-center text-dark
        shadow-sm
        border border-2 border-dark
        align-self-start
      "
    >
      Cinema Halls
    </h2>
    <div class="overflow-auto h-70vh">
      <h1 *ngIf="!allAuditoriums" class="text-danger text-center">
        Cinema Hall list is empty
      </h1>
      <mat-list role="list" *ngIf="allAuditoriums">
        <mat-list-item
          role="listitem"
          *ngFor="let hall of allAuditoriums"
          class="position-relative options pe-cursor rounded"
          [ngClass]="{ highlight: hall.id == selectedCinemaHallId }"
          (click)="onCinemaHallSelect(hall.id!)"
        >
          <div class="w-100">
            {{ hall.name | uppercase }}
          </div>
          <span class="icon-holder w-auto">
            <mat-icon
              class="w-auto text-info me-1 pe-cursor edit-icon"
              aria-hidden="false"
              matTooltip="Edit Cinema Hall"
              (click)="onEditCinemaHall(hall.id!)"
              aria-label="edit icon"
              >edit
            </mat-icon>
            <mat-icon
              class="w-auto text-danger ms-1 pe-cursor delete-icon"
              aria-hidden="false"
              matTooltip="Delete Cinema Hall"
              (click)="onDeleteCinemaHall(hall.id!)"
              aria-label="delete icon"
              >delete
            </mat-icon>
          </span>
        </mat-list-item>
      </mat-list>
    </div>
    <a
      mat-raised-button
      class="align-self-end mt-auto w-100 d-block"
      color="warn"
      routerLink="../add-auditorium"
    >
      Add Cinema Hall
    </a>
  </div>
  <div class="col-lg-3 px-2 border-3 border-end row g-0">
    <h2
      class="
        py-2
        rounded
        text-center text-dark
        shadow-sm
        align-self-start
        border border-2 border-dark
      "
    >
      Shows
    </h2>
    <div class="overflow-auto h-70vh">
      <h1 *ngIf="!selectedShows" class="text-danger text-center">
        Select a Cinema Hall
      </h1>
      <mat-list *ngIf="selectedShows" role="list">
        <mat-list-item
          role="listitem"
          *ngFor="let show of selectedShows"
          class="position-relative options pe-cursor rounded"
          [ngClass]="{ highlight: show.id == selectedShowId }"
          (click)="onShowSelect(show.id!)"
        >
          <div class="w-100 row g-0 justify-content-around">
            <div class="col-5">{{ show.name | titlecase }}</div>
            |
            <div class="col-5">
              {{ "Time: " + formatTime(show.startTime) }}
            </div>
          </div>
          <span class="icon-holder w-auto">
            <mat-icon
              class="w-auto text-info me-1 pe-cursor edit-icon"
              aria-hidden="false"
              matTooltip="Edit Show"
              aria-label="edit icon"
              (click)="onEditShow(show.id!)"
              >edit</mat-icon
            >
            <mat-icon
              class="w-auto text-danger ms-1 pe-cursor delete-icon"
              aria-hidden="false"
              matTooltip="Delete Show"
              aria-label="delete icon"
              (click)="onDeleteShow(show.id!)"
              >delete</mat-icon
            >
          </span>
        </mat-list-item>
      </mat-list>
    </div>
    <button
      *ngIf="selectedShows"
      mat-raised-button
      class="align-self-end mt-auto w-100 d-block"
      (click)="onAddShow()"
      color="accent"
    >
      Add Show
    </button>
  </div>
  <div class="col-lg-6 px-2 row g-0">
    <h2
      class="
        py-2
        rounded
        text-center text-dark
        shadow-sm
        align-self-start
        border border-2 border-dark
      "
    >
      Movies
    </h2>
    <div class="overflow-auto h-70vh">
      <h1
        *ngIf="selectedShowId && !selectedMovieShows"
        class="text-warning text-center"
      >
        Selected Show has no movies
      </h1>
      <h1 *ngIf="!selectedShowId" class="text-danger text-center">
        Select a Show
```



```
        </h1>
        <mat-list *ngIf="selectedShowId" role="list">
          <mat-list-item
            role="listitem"
            *ngFor="let shows of selectedMovieShows"
            class="position-relative show-options pe-cursor rounded h-auto py-
2"
          >
            <div class="w-100">
              <img
                class="example-option-img me-2 rounded d-inline-block"
                aria-hidden
                [src]="getMovieImage(shows.movieId!)"
                height="120"
                width="120"
              />
              {{
                (getShowMovieName(shows.movieId!) | titlecase) +
                  " | lang: " +
                  getShowMovieLanguage(shows.movieId!) +
                  " | start: " +
                  (shows.start | date: "longDate") +
                  " | end: " +
                  (shows.end | date: "longDate")
              }}
            </div>
            <span class="icon-holder w-auto">
              <mat-icon
                class="w-auto text-info me-1 pe-cursor edit-icon"
                aria-hidden="false"
                matTooltip="Edit Movie Show"
                aria-label="edit icon"
                (click)="onEditMovie(shows.id!)"
                >edit</mat-icon
              >
              <mat-icon
                class="w-auto text-danger ms-1 pe-cursor delete-icon"
                aria-hidden="false"
                matTooltip="Delete Movie Show"
                aria-label="delete icon"
                (click)="onDeleteMovie(shows.id!)"
                >delete</mat-icon
              >
            </span>
          </mat-list-item>
        </mat-list>
      </div>
      <div
```

```
        class="row g-0 align-self-end mt-auto col-12 justify-content-between"
      >
        <a
          mat-raised-button
          class="col-5 text-capitalize"
          routerLink="../add-movie"
          color="primary"
        >
          Add New Movie
        </a>
        <button
          *ngIf="selectedShowId"
          mat-raised-button
          class="col-5 text-capitalize"
          (click)="onAddMovieToTheShow()"
          color="warn"
        >
          Add movie to the show
        </button>
      </div>
    </div>
  </div>
</div>
```

Movies:

```
<div class="movie-section">
  <div class="position-relative poster-wrapper p-3">
    <div
      class="movie-banner"
      style="
        background: linear-gradient(
            90deg,
            rgb(34, 34, 34) 24.97%,
            rgb(34, 34, 34) 38.3%,
            rgba(34, 34, 34, 0.04) 97.47%,
            rgb(34, 34, 34) 100%
          )
          no-repeat center center;
        background-size: cover;
      "
      style.background-image="linear-gradient(
        90deg,
        rgb(34, 34, 34) 24.97%,
        rgb(34, 34, 34) 38.3%,
        rgba(34, 34, 34, 0.04) 97.47%,
        rgb(34, 34, 34) 100%
      ), url({{ (selectedMovie$ | async)?.bgImage }})"
    >
      <div class="movie-poster row g-0">
        <div class="movie-image col-lg-3 col-md-6 col-sm-10 rounded-5">
          <img
            [src]="(selectedMovie$ | async)?.image"
            [alt]="(selectedMovie$ | async)?.name"
            width="100%"
            height="100%"
          />
          <div
            class="
              movie-release
              position-absolute
              bottom-0
              start-0
              py-2
              w-100
              bg-white
              text-center
            "
          >
            {{ formatRelease((selectedMovie$ | async)?.release) }}
          </div>
        </div>
        <div class="movie-details text-white mx-3 px-3 col-lg-5 col-sm-12">
          <div class="movie-title mb-lg-5 fs-lg-1">
            <h1>
              {{
                ((selectedMovie$ | async)?.name | uppercase) +
                  " : " +
                  ((selectedMovie$ | async)?.caption | titlecase)
              }}
            </h1>
          </div>
          <div class="movie-format my-3">
            <button class="btn btn-light btn-sm diabled mx-1">4D</button>
            <button class="btn btn-light btn-sm diabled mx-1">
              {{ (selectedMovie$ | async)?.language }}
            </button>
          </div>
          <div class="movie-duration my-3">
            <p style="font: unset; font-size: 20px">
              {{
                ((selectedMovie$ | async)?.duration | titlecase) +
                  " . " +
                  (selectedMovie$ | async)?.genres?.join(", ")
              }}
            </p>
          </div>
          <div class="links mt-4 align-self-end">
            <a
              mat-raised-button
              color="accent"
              class="me-2"
              (click)="openBottomSheet()"
              >Book Tickets</a
            >
          </div>
        </div>
      </div>
    </div>
  </div>

  <div class="about-movie text-dark p-4">
    <h2>About the Movie</h2>
    <p class="lead" style="font-family: sans-serif">
      {{ (selectedMovie$ | async)?.story }}
    </p>
  </div>
  <!-- <nav class="px-4 py-3 w-100 shadow-sm bg-white text-dark">
    <h1>{{(selectedMovie$ | async)?.name + ' : '+ (selectedMovie$ |
async)?.caption}}</h1>
    <p class="lead">{{(selectedMovie$ | async)?.genres.join(', ')}}</p>
  </nav> -->

  <div class="movie-cast ms-5 my-4" *ngIf="(cast$ | async)?.length! > 0">
    <h1 class="fs-lg-2">Cast</h1>
    <div class="row g-0">
      <div
        class="cast text-center col-sm-12 col-md-6 col-lg-3"
        *ngFor="let cast of cast$ | async"
      >
        <div class="image rounded-circle mx-auto mb-3">
          <!-- class="style__Image-eykeme-1 dkwyqp" -->
          <!-- style="border-radius: 60px; opacity: 1" -->
          <img
            [alt]="cast.name"
            width="120px"
            height="120px"
            [src]="cast.image"
          />
        </div>
        <div class="cast-name">
          <h5 class="fs-5">{{ cast.name }}</h5>
        </div>
      </div>
    </div>
  </div>

  <div class="movie-crew ms-5 my-4 pb-5" *ngIf="(crew$ | async)?.length! > 0">
    <h1 class="fs-lg-2">Crew</h1>
    <div class="row g-0">
      <div
        class="crew text-center col-sm-12 col-md-6 col-lg-3"
        *ngFor="let crew of crew$ | async"
      >
        <div class="image rounded-circle mx-auto mb-3">
          <img
            [alt]="crew.name"
            width="120px"
            height="120px"
            [src]="crew.image"
          />
        </div>
        <div class="cast-name">
          <h5 class="fs-5">{{ crew.name }}</h5>
        </div>
      </div>
    </div>
  </div>
</div>
```

Movie Forms:

```
<div class="form-container">
  <div class="mx-auto text-center bg-color">
    <div class="container py-5">
      <h1 class="fs-2 text-dark py-5 text-uppercase">Add New Movie</h1>
    </div>
  </div>

  <div class="form-holder border border-4 border-warning rounded mx-5">
    <form [formGroup]="movieForm" (ngSubmit)="onSubmit()">
      <mat-horizontal-stepper [linear]="true" #stepper>
        <mat-step
          [stepControl]="validMovieDetails"
          errorMessage="Movie details required"
        >
          <ng-template matStepLabel>Movie Details</ng-template>
          <div class="row g-0 my-3 justify-content-around">
```

```
            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Name</mat-label>
                <input
                  matInput
                  placeholder="Movie Name"
                  formControlName="name"
                />
                <mat-error *ngIf="nameErrors">{{ nameErrors }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Release Date</mat-label>
                <input
                  matInput
                  [min]="todayDate"
                  [matDatepicker]="picker"
                  placeholder="Release Date"
                  formControlName="release"
                />
                <mat-datepicker #picker></mat-datepicker>
                <mat-datepicker-toggle
                  matSuffix
                  [for]="picker"
                ></mat-datepicker-toggle>
                <mat-error *ngIf="releaseErrors">{{ releaseErrors }}</mat-
error>

              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Image URL</mat-label>
                <input
                  matInput
                  placeholder="Image Link"
                  formControlName="image"
                />
                <mat-error *ngIf="imageErrors">{{ imageErrors }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Background Image URL</mat-label>
                <input
                  matInput
                  placeholder="Background Image Link"
                  formControlName="bgImage"
                />
                <mat-error *ngIf="bgImageErrors">{{ bgImageErrors }}</mat-
error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Duration</mat-label>
                <input
                  matInput
                  placeholder="2h 25m"
                  formControlName="duration"
                />
                <mat-error *ngIf="durationErrors">{{
                  durationErrors
                }}</mat-error>
              </mat-form-field>
            </div>

            <div class="col-sm-12 col-lg-5 mx-lg-2">
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Movie Caption</mat-label>
                <input matInput formControlName="caption" />
              </mat-form-field>
            </div>
```

```
        <div class="col-sm-12 col-lg-11">
          <mat-form-field
            class="d-block w-100 text-dark"
            appearance="outline"
          >
            <mat-label>Story</mat-label>
            <textarea
              matInput
              formControlName="story"
              placeholder="Narrate the story here"
            ></textarea>
            <mat-error *ngIf="storyErrors">{{ storyErrors }}</mat-error>
          </mat-form-field>
        </div>
      </div>
      <div>
        <button
          mat-button
          matStepperNext
          type="button"
          [disabled]="!validMovieDetails.valid"
        >
          Next
        </button>
      </div>
    </mat-step>

    <mat-step [stepControl]="movieForm.get('languages')!">
      <ng-template matStepLabel>Add Language</ng-template>

      <div class="col-12 my-4 px-sm-1 px xl-4">
        <mat-form-field
          class="example-chip-list"
          class="d-block w-100 text-dark"
        >
          <mat-label>Add Language</mat-label>
          <mat-chip-list #languageList aria-label="language selection">
            <mat-chip
              *ngFor="let language of languages"
              [selectable]="true"
              [removable]="true"
              (removed)="removeLanguage(language)"
            >
              {{ language }}
              <mat-icon matChipRemove *ngIf="true">cancel</mat-icon>
            </mat-chip>
            <input
              placeholder="New Language..."
```

```
                  #languageInput
                  formControlName="languages"
                  [matAutocomplete]="auto"
                  [matChipInputFor]="languageList"
                  [matChipInputSeparatorKeyCodes]="separatorKeysCodes"
                  (matChipInputTokenEnd)="addLanguage($event)"
                />
                <mat-error
                  *ngIf="movieForm.get('languages')?.hasError('required')"
                  >Language cannot be empty</mat-error
                >
              </mat-chip-list>
              <mat-autocomplete
                #auto="matAutocomplete"
                (optionSelected)="selectedLanguage($event)"
              >
                <mat-option
                  *ngFor="let language of filteredLanguages | async"
                  [value]="language"
                >
                  {{ language }}
                </mat-option>
              </mat-autocomplete>
            </mat-form-field>
          </div>

          <!-- <div class="row g-0 justify-content-around">
            <div
              class="col-sm-12 col-lg-5 form-field"
              formArrayName="genres"
              *ngFor="let genre of genres.controls; index as i"
            >
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Genre</mat-label>
                <input matInput placeholder="Romantic" [formControlName]="i"
/>
                <mat-error *ngIf="genre.errors?.required && genres.touched"
                  >Genre Cannot be empty</mat-error
                >
              </mat-form-field>
              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeGenre(i)"
                aria-hidden="false"
                matTooltip="Remove the Genre"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more Genre"
              aria-label="add_circle icon"
              (click)="addGenre()"
              >add_circle</mat-icon
            >
          </div> -->
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step [stepControl]="movieForm.get('genres')!">
          <ng-template matStepLabel>Add Language</ng-template>
          <div class="row g-0 justify-content-around">
            <div
              class="col-sm-12 col-lg-5 form-field"
              formArrayName="genres"
              *ngFor="let genres of genres.controls; index as i"
            >
              <mat-form-field
                class="d-block w-100 text-dark"
                appearance="outline"
              >
                <mat-label>Genre</mat-label>
                <input
                  matInput
                  placeholder="Comedy"
                  [formControlName]="i"
                  [attr.list]="'genres-' + i"
                />
                <datalist [id]="'genres-' + i">
                  <option [value]="genre" *ngFor="let genre of allGenres">
                    {{ genre }}
                  </option>
                </datalist>

                <mat-error *ngIf="genres.touched">
                  <span class="mx-1" *ngIf="genres.errors?.required"
                    >Genre cannot be empty</span
                  >
                  <span class="mx-1" *ngIf="genres.errors?.validGenre"
                    >Invalid movie genre</span
                  >
                  <span class="mx-1" *ngIf="genres.errors?.uniqueName"
                    >Genre already added</span
                  >
                </mat-error>
              </mat-form-field>

              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeGenre(i)"
                aria-hidden="false"
                matTooltip="Remove the Genre"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more Genre"
              aria-label="add_circle icon"
              (click)="addGenre()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step [stepControl]="movieForm.get('casts')!">
          <ng-template matStepLabel>Add Cast Details</ng-template>
          <div
            class="w-100"
            formArrayName="casts"
            *ngFor="let cast of casts.controls; index as i"
          >
            <div
              class="
                row
                g-0
                justify-content-around
                align-content-center
                form-field
              "
              [formGroupName]="i"
            >
              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Actor Name</mat-label>
                  <input
                    matInput
                    placeholder="Actor Name"
                    formControlName="name"
                  />
                  <mat-error
                    *ngIf="cast.get('name')?.errors?.required && cast.touched"
                  >
                    Name cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>

              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Actor Role</mat-label>
                  <input matInput placeholder="Hero" formControlName="role" />
                  <mat-error
                    *ngIf="cast.get('role')?.errors?.required && cast.touched"
                  >
                    Actor role cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>

              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Actor Image</mat-label>
                  <input
                    matInput
```

```
                    placeholder="Actor image"
                    formControlName="image"
                  />
                  <mat-error
                    *ngIf="cast.get('image')?.errors?.required &&
cast.touched"
                  >
                    Actor image cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>
              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeCast(i)"
                aria-hidden="false"
                matTooltip="Remove the Cast"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more Cast"
              aria-label="add_circle icon"
              (click)="addCast()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step [stepControl]="movieForm.get('crews')!">
          <ng-template matStepLabel>Add Crew Details</ng-template>
          <div
            class="w-100"
            formArrayName="crews"
            *ngFor="let crew of crews.controls; index as i"
          >
            <div
              class="
                row
                g-0
                justify-content-around
                align-content-center
                form-field
              "
              [formGroupName]="i"
            >
              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Crew Name</mat-label>
                  <input
                    matInput
                    placeholder="Crew Name"
                    formControlName="name"
                  />
                  <mat-error
                    *ngIf="crew.get('name')?.errors?.required && crew.touched"
                  >
                    Name cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>

              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Crew Role</mat-label>
                  <input
                    matInput
                    placeholder="Director"
                    formControlName="role"
                  />
                  <mat-error
                    *ngIf="crew.get('role')?.errors?.required && crew.touched"
                  >
                    Crew role cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>

              <div class="col-lg-3 col-sm-6">
                <mat-form-field
                  class="d-block w-100 text-dark"
                  appearance="outline"
                >
                  <mat-label>Crew Image</mat-label>
                  <input
                    matInput
                    placeholder="Crew image"
                    formControlName="image"
                  />
                  <mat-error
                    *ngIf="crew.get('image')?.errors?.required &&
crew.touched"
                  >
                    Crew image cannot be empty
                  </mat-error>
                </mat-form-field>
              </div>
              <mat-icon
                class="fs-2 w-auto text-danger delete-icon"
                (click)="removeCrew(i)"
                aria-hidden="false"
                matTooltip="Remove the Cast"
                aria-label="delete icon"
                >delete</mat-icon
              >
            </div>
          </div>
          <div class="text-center text-primary mt-4">
            <mat-icon
              class="pe-cursor fs-2"
              aria-hidden="false"
              matTooltip="Add one more Cast"
              aria-label="add_circle icon"
              (click)="addCrew()"
              >add_circle</mat-icon
            >
          </div>
          <div>
            <button mat-button matStepperPrevious type="button">Back</button>
            <button mat-button matStepperNext type="button">Next</button>
          </div>
        </mat-step>

        <mat-step>
          <ng-template matStepLabel>Review</ng-template>
          <pre>
              {{ movieForm.value | json }}
            </pre
          >
          <div>
```

```
            <button mat-button matStepperPrevious type="button">Back</button>
            <button
              mat-button
              class="text-success"
              [disabled]="!movieForm.valid"
            >
              Submit
            </button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </form>
  </div>
</div>
```

Movies:

```
<div class="px-lg-5 row g-0 my-5 justify-content-around position-relative">
  <div class="col-3">
    <div class="sorting text-center mb-3 align-self-center">
      <h2 class="fs-2 text-center text-dark text-uppercase">Show</h2>
      <br />
      <mat-form-field appearance="outline" class="d-block w-100">
        <mat-label>Show</mat-label>
        <mat-select name="showText" [(ngModel)]="showText">
          <mat-option
            *ngFor="let type of types"
            [value]="type"
            (onSelectionChange)="onTypeChange(type)"
            >{{ type }}</mat-option
          >
        </mat-select>
      </mat-form-field>
    </div>
    <br />
    <div class="filter">
      <h2 class="fs-2 text-center text-dark text-uppercase">Filters</h2>
      <br />
      <mat-accordion
        class="example-headers-align"
        multi
        togglePosition="before"
      >
        <mat-expansion-panel class="rounded my-1">
          <mat-expansion-panel-header>
            <mat-panel-title
```

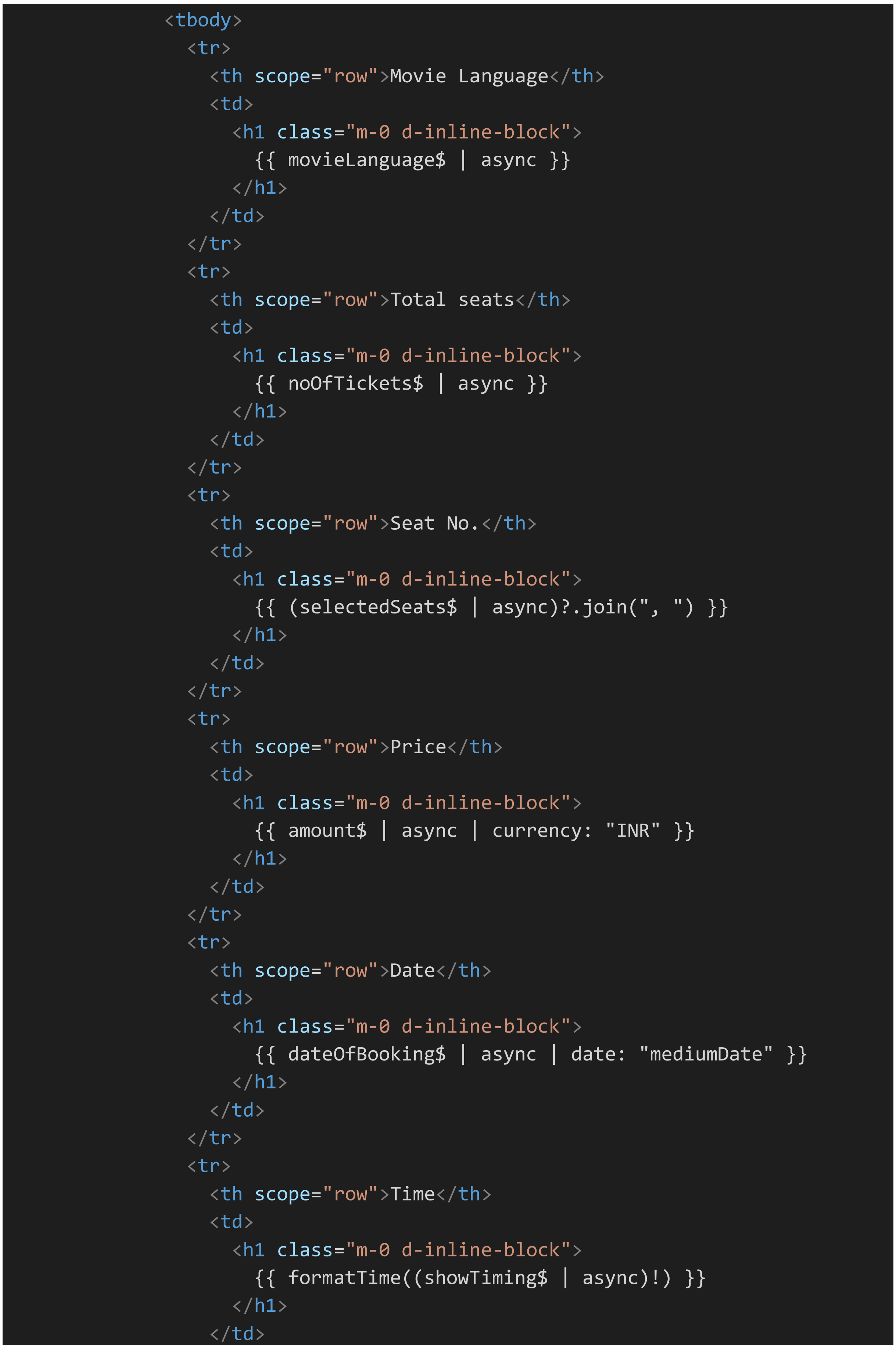
```
              [ngClass]="{
                'text-danger': languageFilter != null && languageFilter != ''
              }"
            >
              Language
            </mat-panel-title>
            <mat-panel-description
              (click)="clearLanguageFilter()"
              class="clear-filter"
              [ngClass]="{
                'text-danger': languageFilter != null && languageFilter != ''
              }"
            >
              Clear
            </mat-panel-description>
          </mat-expansion-panel-header>
          <div class="row g-0 justify-content-around">
            <div
              class="my-2 col-4 text-start"
              *ngFor="let language of allLanguages"
            >
              <input
                type="radio"
                autocomplete="off"
                [checked]="language == languageFilter"
                class="btn-check"
                [id]="language"
                name="selectedLanguage"
                (change)="handleLanguageChange(language)"
              />
              <label
                class="btn btn-sm"
                [for]="language"
                [ngClass]="{
                  'btn-warning': language == languageFilter,
                  'btn-info': language != languageFilter
                }"
                >{{ language }}</label
              >
            </div>
          </div>
        </mat-expansion-panel>

        <mat-expansion-panel class="rounded my-1">
          <mat-expansion-panel-header>
            <mat-panel-title
              [ngClass]="{
                'text-danger': genreFilter != null && genreFilter != ''
              }"
            >
              Genre
            </mat-panel-title>
            <mat-panel-description
              class="clear-filter"
              (click)="clearGenreFilter()"
              [ngClass]="{
                'text-danger': genreFilter != null && genreFilter != ''
              }"
              >Clear</mat-panel-description
            >
          </mat-expansion-panel-header>
          <div class="row g-0 justify-content-around">
            <div class="my-2 col-4 text-start" *ngFor="let genre of
allGenres">
              <input
                type="radio"
                autocomplete="off"
                [checked]="genre == genreFilter"
                class="btn-check"
                [id]="genre"
                name="selectedGenre"
                (change)="handleGenreChange(genre)"
              />
              <label
                class="btn btn-sm"
                [for]="genre"
                [ngClass]="{
                  'btn-warning': genre == genreFilter,
                  'btn-info': genre != genreFilter
                }"
                >{{ genre }}</label
              >
            </div>
          </div>
        </mat-expansion-panel>
      </mat-accordion>
    </div>
  </div>

  <div class="col-8 movies">
    <div class="now-showing">
      <mat-toolbar
        [color]="showText == 'Now Playing' ? 'primary' : 'accent'"
        class="rounded"
      >
        <div class="row g-0 justify-content-around w-100 h-auto py-3">
```

```
          <div class="col-sm-12 col-lg-3 text-center">{{ showText }}</div>
          <div class="col-sm-12 col-lg-9">
            <input
              class="form-control me-2 w-100"
              type="search"
              placeholder="Search for movie, genre or language"
              aria-label="Search"
              name="search"
              [(ngModel)]="search"
            />
          </div>
        </div>
      </mat-toolbar>
      <h1 class="text-center text-warning my-4" *ngIf="!(moviesList$ |
async)">
        No movies found
      </h1>
      <div class="now-showing-movies my-4" *ngIf="moviesList$ | async">
        <div class="card-group justify-content-around">
          <ng-container
            *ngIf="
              moviesList$
                | async
                | search: search:genreFilter:languageFilter as result
            "
          >
            <p
              class="lead my-2 p-2 rounded alert-danger"
              *ngIf="result.length < 1 || !result"
            >
              No result found with
              <span class="text-danger"
                >{{ search }}{{ " " + languageFilter
                }}{{ " " + genreFilter }}</span
              >
            </p>
            <div
              class="col-4 px-2 position-relative"
              *ngFor="let movie of result"
            >
              <div
                class="image-card position-relative rounded-5 overflow-hidden"
              >
                <img
                  [src]="movie.image"
                  class="card-img-top"
                  height="400"
                  width="280"
                  [alt]="movie.name"
                />
                <!-- <svg
                  class="bd-placeholder-img card-img-top"
                  width="100%"
                  height="350"
                  xmlns="http://www.w3.org/2000/svg"
                  role="img"
                  aria-label="Placeholder: Thumbnail"
                  preserveAspectRatio="xMidYMid slice"
                  focusable="false"
                >
                  <title>Placeholder</title>
                  <rect width="100%" height="100%" fill="#55595c"></rect>
                </svg> -->
                <div
                  class="
                    image-release
                    position-absolute
                    bottom-0
                    start-0
                    bg-dark
                    text-white
                    w-100
                    text-center
                    py-2
                  "
                >
                  {{ formatRelease(movie.release) }}
                </div>
              </div>
              <div class="card-body text-center">
                <h2 class="card-title fs-4 mb-0">
                  <a
                    class="stretched-link"
                    routerLink="../movie/{{ movie.id }}"
                    >{{ movie.name | uppercase }}</a
                  >
                </h2>
                <h3 class="card-text text-muted">
                  {{ movie.language | titlecase }}
                </h3>
              </div>
            </div>
          </ng-container>
        </div>
      </div>
    </div>
```

```
  </div>
</div>
```

My Bookings:

```
<div class="col-10 mx-auto my-4 h-80vh">
  <mat-toolbar color="primary" class="rounded text-center">
    <h1
      class="text-center mx-auto"
      [ngClass]="present ? '' : 'text-danger h-80vh'"
    >
      {{ heading }}
    </h1>
  </mat-toolbar>

  <mat-card *ngIf="present">
    <div class="row g-0 col-12">
      <div class="col-2 p-2">
        <div
          class="my-2 col-12 text-start"
          *ngFor="let booking of allBookings$ | async; index as i"
        >
          <input
            type="radio"
            autocomplete="off"
            [checked]="booking.id == selectedBookingId"
            class="btn-check"
            [id]="'booking-' + i"
            name="booking"
            (change)="onBookingChange(booking.id!)"
          />
          <label
            class="btn btn-sm w-100 clearfix"
            [for]="'booking-' + i"
            [ngClass]="{
              'btn-dark text-white': booking.id == selectedBookingId,
              'btn-info': booking.id != selectedBookingId
            }"
          >
            <span class="d-inline-block float-start">{{
              "Booking " + (i + 1)
            }}</span>
            <span class="d-inline-block float-end">{{ booking.bookedOn
}}</span>
          </label>
        </div>
      </div>
      <div class="col-10 px-2">
        <div class="row g-0 m-3 p-2 shadow-sm border">
          <div
            class="
              col-4
              mx-auto
              text-center
              px-1
              position-relative
              overflow-hidden
            "
          >
            <img
              [src]="movieImage$ | async"
              [alt]="movieName$ | async"
              height="400"
              width="300"
            />
            <div
              class="
                movie-name
                position-absolute
                w-100
                py-2
                fs-5
                bg-dark
                text-white
                start-0
                bottom-0
              "
            >
              {{ movieName$ | async | uppercase }}
            </div>
          </div>
          <div class="col-8 px-2 details h-100 my-auto">
            <table class="table ms-3">
              <thead>
                <tr>
                  <th
                    scope="col"
                    colspan="2"
                    class="text-center fs-4 text-capitalize"
                  >
                    {{ auditoriumName$ | async }}
                  </th>
                </tr>
              </thead>
```



```
<tbody>
  <tr>
    <th scope="row">Movie Language</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ movieLanguage$ | async }}
      </h1>
    </td>
  </tr>
  <tr>
    <th scope="row">Total seats</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ noOfTickets$ | async }}
      </h1>
    </td>
  </tr>
  <tr>
    <th scope="row">Seat No.</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ (selectedSeats$ | async)?.join(", ") }}
      </h1>
    </td>
  </tr>
  <tr>
    <th scope="row">Price</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ amount$ | async | currency: "INR" }}
      </h1>
    </td>
  </tr>
  <tr>
    <th scope="row">Date</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ dateOfBooking$ | async | date: "mediumDate" }}
      </h1>
    </td>
  </tr>
  <tr>
    <th scope="row">Time</th>
    <td>
      <h1 class="m-0 d-inline-block">
        {{ formatTime((showTiming$ | async)!) }}
      </h1>
    </td>
```

```
                </tr>
                <tr>
                  <th scope="row">Status</th>
                  <td>
                    <span class="text-muted text-success">Confirmed</span>
                  </td>
                </tr>
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
  </mat-card>
</div>
```

Profile:

```
<div class="h-80vh">
  <div class="col-10 mx-auto my-4">
    <mat-toolbar color="primary" class="rounded text-center"
      >Your profile</mat-toolbar
    >
  </div>
  <mat-card class="col-10 mx-auto">
    <table class="table">
      <tbody>
        <tr>
          <th scope="row">Name</th>
          <td>{{ (user | async)?.name }}</td>
        </tr>
        <tr>
          <th scope="row">Email</th>
          <td>{{ (user | async)?.email }}</td>
        </tr>
        <tr>
          <th scope="row">Mobile</th>
          <td>{{ (user | async)?.mobile }}</td>
        </tr>
        <tr>
          <th scope="row">Gender</th>
          <td>{{ (user | async)?.gender }}</td>
        </tr>
      </tbody>
    </table>
  </mat-card>
```

```
</div>
```

Register:

```
<main class="px-3 my-auto">
  <p
    class="lead mb-2 p-2 rounded"
    [ngClass]="(alertDanger$ | async) ? 'alert-danger' : 'alert-success'"
    *ngIf="showAlert$ | async"
  >
    {{ alertMessage$ | async }}
  </p>
  <h1 class="fs-2 py-4">Register Here</h1>

  <div class="example-container">
    <form
      class="row w-100 g-0"
      [formGroup]="registerForm"
      (ngSubmit)="onSubmit()"
    >
      <!-- Email Field -->
      <div class="col-12 mb-2 px-lg-1">
        <mat-form-field appearance="outline" class="d-block text-white">
          <mat-label>Enter your Email</mat-label>
          <input matInput type="email" formControlName="email" class="fs-5" />
          <mat-hint *ngIf="email.pending" class="text-warning"
            >Checking For uniqueness...</mat-hint
          >
          <mat-error *ngIf="emailErrors">{{ emailErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">
        <mat-form-field appearance="outline" class="d-block text-white">
          <mat-label>Enter your name</mat-label>
          <input matInput type="text" formControlName="name" class="fs-5" />
          <mat-error *ngIf="nameErrors">{{ nameErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">
        <mat-form-field appearance="outline" class="d-block text-white">
          <mat-label>Enter your Mobile</mat-label>
          <span matPrefix>+91 &nbsp;</span>
          <input
            matInput
            type="text"
            formControlName="mobile"
            class="fs-5"
          /><mat-hint *ngIf="mobile.pending" class="text-warning"
            >Checking For uniqueness...</mat-hint
          >
          <mat-error *ngIf="mobileErrors">{{ mobileErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">
        <mat-form-field appearance="outline" class="d-block text-white">
          <mat-label>Enter your password</mat-label>
          <input
            matInput
            [type]="hidePassword ? 'password' : 'text'"
            formControlName="password"
            class="fs-5"
          />
          <mat-error *ngIf="passwordErrors">{{ passwordErrors }}</mat-error>
          <button
            mat-icon-button
            matSuffix
            (click)="hidePassword = !hidePassword"
            [attr.aria-label]="'Hide password'"
            [attr.aria-pressed]="hidePassword"
          >
            <mat-icon>{{
              hidePassword ? "visibility_off" : "visibility"
            }}</mat-icon>
          </button>
        </mat-form-field>
      </div>

      <div class="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">
        <mat-form-field appearance="outline" class="d-block text-white">
          <mat-label>Select Gender</mat-label>
          <mat-select formControlName="gender" class="text-white fs-5">
            <mat-option *ngFor="let sex of genders" [value]="sex.name">
              {{ sex.name }}
            </mat-option>
          </mat-select>
          <mat-error *ngIf="passwordErrors">{{ passwordErrors }}</mat-error>
        </mat-form-field>
      </div>

      <div class="col-lg-12 col-md-12 mb-2">
        <mat-checkbox class="example-margin" formControlName="terms"
```

```
          >Agree to terms & conditions</mat-checkbox
        >
        <mat-error *ngIf="termsErrors">{{ termsErrors }}</mat-error>
      </div>

      <button
        mat-raised-button
        [color]="!registerForm.valid ? 'accent' : 'primary'"
        class="col-12"
        type="submit"
      >
        Register
      </button>
    </form>
  </div>
</main>
```

Add Movie To Show Form:

```
<div class="wrapper">
  <h1 class="text-center mb-3 p-2 rounded border border-2 shadow-sm">
    Add Movie To Show
  </h1>

  <form class="example-form p-4" [formGroup]="movieShowForm">
    <mat-form-field class="d-block w-100 text-dark" appearance="outline">
      <mat-label>Select Movie</mat-label>
      <mat-select
        aria-label="State"
        [formControl]="movieId"
        (click)="movieId.reset()"
        (selectionChange)="onMovieChange($event)"
      >
        <mat-option
          *ngFor="let movie of filteredMovies | async"
          [value]="movie.id"
          class="h-auto py-1"
        >
          <!-- (onSelectionChange)="onMovieChange($event, movie.id)" -->
          <img
            class="example-option-img me-2 rounded"
            aria-hidden
            [src]="movie.image"
            height="120"
            width="120"
          />
          <span class="me-1">{{ movie.name | uppercase }}</span> |
          <span class="me-1">{{ movie.language | titlecase }}</span> |
          <small class="me-1"
            >Release: {{ movie.release | date: "fullDate" }}</small
          >
          |
          <small class="me-1"
            >Added: {{ movie.addedOn | date: "fullDate" }}</small
          >
        </mat-option>
      </mat-select>
      <mat-error *ngIf="movieIdErrors">{{ movieIdErrors }}</mat-error>
    </mat-form-field>

    <mat-form-field appearance="outline" class="d-block w-100 text-dark">
      <mat-label>Enter a date range</mat-label>
      <mat-date-range-input
        [formGroup]="range"
        [rangePicker]="picker"
        [min]="startDate"
        [max]="endDate"
      >
        <input matStartDate formControlName="start" placeholder="Start date"
/>
        <input matEndDate formControlName="end" placeholder="End date" />
      </mat-date-range-input>
      <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
      <mat-date-range-picker #picker></mat-date-range-picker>
      <mat-error class="d-inline-block mx-1" *ngIf="startDateErrors">{{
        startDateErrors
      }}</mat-error>
      <mat-error class="d-inline-block mx-1" *ngIf="endDateErrors">{{
        endDateErrors
      }}</mat-error>
      <!-- <mat-error
*ngIf="range.controls.start.hasError('matStartDateInvalid')"
        >Invalid start date</mat-error
      >
      <mat-error *ngIf="range.controls.end.hasError('matEndDateInvalid')"
        >Invalid end date</mat-error
      > -->
    </mat-form-field>
    <fieldset>
      <legend class="lead">Price details</legend>
      <div class="row g-0 justify-content-around">
        <mat-form-field
          appearance="outline"
          class="d-block col-4 px-1 text-dark"
        >
          <mat-label>Gold Section</mat-label>
          <input type="number" matInput formControlName="gold" />
          <mat-error *ngIf="goldPriceErrors">{{ goldPriceErrors }}</mat-error>
        </mat-form-field>
        <mat-form-field
          appearance="outline"
          class="d-block col-4 px-1 text-dark"
        >
          <mat-label>Silver Section</mat-label>
          <input type="number" matInput formControlName="silver" />
          <mat-error *ngIf="silverPriceErrors">{{
            silverPriceErrors
          }}</mat-error>
        </mat-form-field>
        <mat-form-field
          appearance="outline"
          class="d-block col-4 px-1 text-dark"
        >
          <mat-label>General Section</mat-label>
          <input type="number" matInput formControlName="general" />
          <mat-error *ngIf="generalPriceErrors">{{
            generalPriceErrors
          }}</mat-error>
        </mat-form-field>
      </div>
    </fieldset>
  </form>
  <button mat-raised-button (click)="onCancel()" class="me-2">Cancel</button>
  <button
    mat-raised-button
    (click)="onSubmit()"
    [disabled]="!movieShowForm.valid"
    color="primary"
    class="ms-2"
  >
    Submit
  </button>
</div>
```

Payment Form:

```
<div class="wrapper p-5">
  <div class="text-center mb-3 p-2 rounded border border-2 shadow-sm">
    <h1 class="text-center d-inline-block my-2">Payment Gateway</h1>
```

```html
    (<span class="text-danger d-inline-block my-2">do not refresh the
page</span
    >)
  </div>
  <div class="my-auto h-70vh row g-0 justify-content-around">
    <div class="col-6 px-2 mx-auto order-2">
      <form [formGroup]="paymentForm" class="example-form p-4">
        <mat-form-field appearance="outline" class="d-block w-100 text-dark">
          <mat-label>Card Number</mat-label>
          <input
            matInput
            formControlName="cardNumber"
            type="tel"
            #ccNumber
            class="fs-3 bolder col-12"
            (keyup)="creditCardNumberSpacing()"
          />
          <mat-hint>16 digit card number</mat-hint>
          <mat-error *ngIf="cardNumberErrors">{{ cardNumberErrors }}</mat-
error>
        </mat-form-field>

        <div class="row g-0 justify-content-around">
          <mat-form-field
            appearance="outline"
            class="d-block text-dark col-sm-12 col-lg-4 px-1"
          >
            <mat-label>Month</mat-label>
            <mat-select formControlName="cardExpiryMonth">
              <mat-option *ngFor="let month of tempMonths" [value]="month">
                {{ month }}
              </mat-option>
            </mat-select>
            <mat-hint>Select card expiry month</mat-hint>
            <mat-error *ngIf="monthErrors">{{ monthErrors }}</mat-error>
          </mat-form-field>
          <mat-form-field
            appearance="outline"
            class="d-block text-dark col-sm-12 col-lg-4 px-1"
          >
            <mat-label>Year</mat-label>
            <mat-select formControlName="cardExpiryYear">
              <mat-option *ngFor="let year of tempYears" [value]="year">
                {{ year }}
              </mat-option>
            </mat-select>
            <mat-hint>select card expiry year</mat-hint>
            <mat-error *ngIf="yearErrors">{{ yearErrors }}</mat-error>
          </mat-form-field>
          <mat-form-field
            appearance="outline"
            class="d-block text-dark col-sm-12 col-lg-4 px-1"
          >
            <mat-label>CVV</mat-label>
            <input
              matInput
              [type]="cvvHide ? 'password' : 'text'"
              formControlName="cardCVV"
            />
            <button
              mat-icon-button
              matSuffix
              (click)="cvvHide = !cvvHide"
              [attr.aria-label]="'Hide password'"
              [attr.aria-pressed]="cvvHide"
            >
              <mat-icon>{{
                cvvHide ? "visibility_off" : "visibility"
              }}</mat-icon>
            </button>
            <mat-error *ngIf="cvvErrors">{{ cvvErrors }}</mat-error>
          </mat-form-field>
        </div>

        <button
          mat-raised-button
          (click)="onSubmit()"
          [disabled]="!paymentForm.valid"
          color="primary"
          class="col-12 my-3"
        >
          Pay {{ tempScreen.amount | currency: "INR" }}
        </button>
      </form>
    </div>
    <div class="col-6 px-2 mx-auto order-1">
      <table class="table">
        <thead>
          <tr>
            <th scope="col" colspan="2">Booking Details</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>Auditorium</td>
            <th scope="row">{{ selectedMembers.auditoriumName }}</th>
```

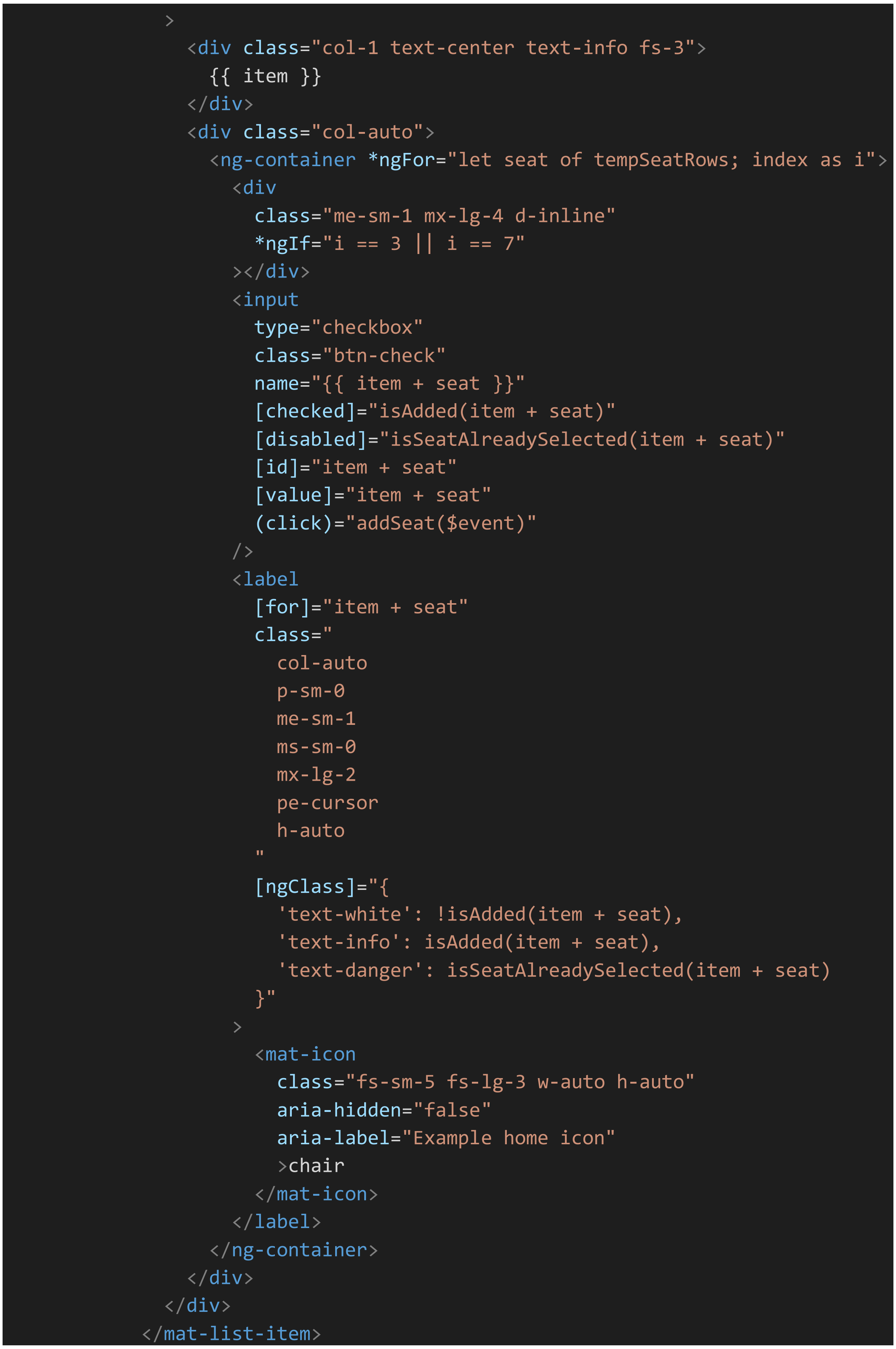
```
          </tr>
          <tr>
            <td>Show</td>
            <th scope="row">
              {{ selectedMembers.showName }}
            </th>
          </tr>
          <tr>
            <td>Show Timing</td>
            <th scope="row">
              {{ formatTime(selectedMembers.showTime) }}
            </th>
          </tr>
          <tr>
            <td>Movie Name</td>
            <th scope="row">{{ selectedMembers.movieName | uppercase }}</th>
          </tr>
          <tr>
            <td>Movie Lang</td>
            <th scope="row">{{ selectedMembers.movieLanguage }}</th>
          </tr>
          <tr>
            <td>Date</td>
            <th scope="row">{{ selectedMembers.date | date: "mediumDate"
}}</th>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

Screen:

```
<div class="screen-container w-100 py-3 bg-dark text-white">
  <span class="d-inline-block text-warning mx-auto text-center py-2 w-100"
    >NOTE: do not refresh the page</span
  >
  <div class="my-auto row g-0 align-content-center">
    <div class="ticket-details mb-3 row g-0 justify-content-between">
      <div class="col-4 text-center">
        Movie: &nbsp;
        <span class="lead text-white">{{
          selectMembers.movieName | uppercase
        }}</span>
      </div>
```

```html
    <div class="col-4 text-center">
      Time: &nbsp;
      <span class="lead">{{ formatTime(selectMembers.showTime) }}</span>
    </div>

    <div class="col-4 text-center">
      Date: &nbsp;
      <span class="lead">{{ selectMembers.date | date: "fullDate" }}</span>
    </div>
  </div>

  <div class="ticket-details mb-3 row g-0 justify-content-between">
    <div class="col-4 text-center">
      Seats Selected: &nbsp;
      <span class="lead text-danger">{{ getSelectedSeats() }}</span>
    </div>

    <div class="col-4 text-center">
      Amount: &nbsp;
      <span class="lead">{{ totalAmount }}</span>
    </div>

    <div class="col-4 text-center">
      <button
        mat-raised-button
        type="submit"
        [disabled]="seatsToBeSelected != 0"
        (click)="onProceed()"
      >
        Proceed
      </button>
    </div>
  </div>

  <!-- <div class="row g-0 text-center">
    <div class="col-4 text-center">
      <mat-icon
        class="fs-sm-5 fs-lg-3 w-auto h-auto text-danger"
        aria-hidden="false"
        aria-label="Example home icon"
        >chair
      </mat-icon>
      - <span class="text-white">Booked</span>
    </div>
    <div class="col-4 text-center">
      <mat-icon
        class="fs-sm-5 fs-lg-3 w-auto h-auto text-white"
```

```
          aria-hidden="false"
          aria-label="Example home icon"
          >chair
        </mat-icon>
        - <span class="text-white">Avaliable</span>
      </div>
      <div class="col-4 text-center">
        <mat-icon
          class="fs-sm-5 fs-lg-3 w-auto h-auto text-info"
          aria-hidden="false"
          aria-label="Example home icon"
          >chair
        </mat-icon>
        - <span class="text-white">Your Selection</span>
      </div>
    </div> -->

    <div
      class="screen-layout position-relative w-100 bg-dark text-white p-sm-0"
    >
      <mat-card class="bg-dark text-white text-center">
        <mat-list class="bg-dark text-white">
          <ng-container
            *ngFor="let item of tempSeatColumns.reverse(); index as i"
          >
            <h3 class="text-danger m-1" *ngIf="i == 7">
              <!-- EXECUTIVE &#8377;200.00 -->
              EXECUTIVE {{ generalPrice$ | async | currency: "INR" }}
            </h3>
            <h3 class="text-danger m-1" *ngIf="i == 3">
              SILVER {{ silverPrice$ | async | currency: "INR" }}
            </h3>
            <h3 class="text-danger m-1" *ngIf="i == 0">
              GOLD {{ goldPrice$ | async | currency: "INR" }}
            </h3>
            <mat-list-item class="justify-content-center py-1 h-auto px-sm-0">
              <!-- <div class="col-1 text-info fs-3 px-sm-0">{{ item }}</div>
-->

              <div
                class="
                  row
                  col-12
                  g-0
                  mx-auto
                  justify-content-center
                  px-sm-0
                  text-center
                "
```



```
  >
    <div class="col-1 text-center text-info fs-3">
      {{ item }}
    </div>
    <div class="col-auto">
      <ng-container *ngFor="let seat of tempSeatRows; index as i">
        <div
          class="me-sm-1 mx-lg-4 d-inline"
          *ngIf="i == 3 || i == 7"
        ></div>
        <input
          type="checkbox"
          class="btn-check"
          name="{{ item + seat }}"
          [checked]="isAdded(item + seat)"
          [disabled]="isSeatAlreadySelected(item + seat)"
          [id]="item + seat"
          [value]="item + seat"
          (click)="addSeat($event)"
        />
        <label
          [for]="item + seat"
          class="
            col-auto
            p-sm-0
            me-sm-1
            ms-sm-0
            mx-lg-2
            pe-cursor
            h-auto
          "
          [ngClass]="{
            'text-white': !isAdded(item + seat),
            'text-info': isAdded(item + seat),
            'text-danger': isSeatAlreadySelected(item + seat)
          }"
        >
          <mat-icon
            class="fs-sm-5 fs-lg-3 w-auto h-auto"
            aria-hidden="false"
            aria-label="Example home icon"
            >chair
          </mat-icon>
        </label>
      </ng-container>
    </div>
  </div>
</mat-list-item>
```

```
          </ng-container>
        </mat-list>
      </mat-card>
    </div>

    <div
      class="
        screen
        col-10
        mx-auto
        py-3
        text-white
        bg-danger
        text-center
        mt-sm-1 mt-lg-4
      "
    >
      Screen is this way
    </div>
  </div>
</div>
```

Select Members:

```
<mat-card>
  <form [formGroup]="ticketsForm" (ngSubmit)="proceed()">
    <mat-horizontal-stepper [linear]="true" labelPosition="bottom" #stepper>
      <mat-step [stepControl]="ticketsForm.get('auditoriumName')!">
        <ng-template matStepLabel>Select the Cinema Hall</ng-template>
        <mat-form-field class="d-block w-100 mt-2">
          <mat-label>Cinema Hall Name</mat-label>
          <mat-select class="text-dark" formControlName="auditoriumName">
            <mat-option
              *ngFor="let hall of allAuditoriums$ | async"
              [value]="hall.name"
              (onSelectionChange)="onAuditoriumSelect(hall.id!, hall.name!)"
            >
              {{ hall.name }}
            </mat-option>
          </mat-select>
          <mat-error *ngIf="auditoriumErrors">{{ auditoriumErrors }}</mat-
error>
        </mat-form-field>
        <div>
          <button mat-button matStepperNext type="button">Next</button>
        </div>
```

```html
    </mat-step>

    <mat-step
      [stepControl]="ticketsForm.get('showName')!"
      label="Select the Show Timing"
    >
      <mat-form-field class="d-block w-100 mt-2">
        <mat-label class="text-dark">Show Timing</mat-label>
        <mat-select class="text-dark" formControlName="showName">
          <mat-option
            *ngFor="let show of allShows$ | async"
            [value]="show.name"
            (onSelectionChange)="onShowSelect(show.id!, show.name!)"
          >
            {{ show.name + " | Time: " + formatTime(show.startTime) }}
          </mat-option>
        </mat-select>
        <mat-error *ngIf="showErrors">{{ showErrors }}</mat-error>
      </mat-form-field>
      <div>
        <button mat-button matStepperPrevious type="button">Back</button>
        <button mat-button matStepperNext type="button">Next</button>
      </div>
    </mat-step>

    <mat-step
      [stepControl]="ticketsForm.get('date')!"
      label="Select the Date"
    >
      <mat-form-field class="d-block w-100 mt-2">
        <mat-label class="text-dark">Choose a date</mat-label>
        <input
          class="text-dark"
          formControlName="date"
          (dateChange)="onDateSelect($event)"
          matInput
          [matDatepicker]="picker"
          [min]="startDate$ | async"
          [max]="endDate$ | async"
        />
        <mat-datepicker-toggle
          matSuffix
          [for]="picker"
        ></mat-datepicker-toggle>
        <mat-datepicker #picker></mat-datepicker>
        <mat-error *ngIf="dateErrors">{{ dateErrors }}</mat-error>
      </mat-form-field>
      <div>
```

```
        <button mat-button matStepperPrevious type="button">Back</button>
        <button mat-button matStepperNext type="button">Next</button>
      </div>
    </mat-step>

    <mat-step
      [stepControl]="ticketsForm.get('seats')!"
      label="Select No.of seats"
    >
      <div
        class="col-12 my-4 text-center"
        [ngClass]="selectedSeats! > 0 ? 'text-primary' : 'text-danger'"
      >
        <mat-icon
          class="fs-1"
          aria-hidden="false"
          aria-label="Example home icon"
        >
          {{ icon }}
        </mat-icon>
      </div>
      <div *ngIf="selectedSeats! < 1" class="text-center text-danger">
        No Seats avaliable, please select different show timing
      </div>
      <div
        *ngIf="selectedSeats! > 0"
        class="row g-0 justify-content-around mb-4"
      >
        <div
          class="col-auto mx-2"
          *ngFor="let seat of avaliableSeats$ | async"
        >
          <input
            type="radio"
            class="btn-check"
            formControlName="seats"
            [checked]="seat == selectedSeats"
            (change)="onSeatsChange(seat)"
            [id]="seat"
            [value]="seat"
          />
          <label
            [for]="seat"
            class="px-3 py-2 rounded-circle border border-4 pe-cursor fs-5"
            [ngClass]="
              seat == selectedSeats!
                ? 'text-success, border-success'
                : 'text-warning, border-secondary'
            "
          >
            {{ seat }}
          </label>
        </div>
      </div>
      <div>
        <button mat-button matStepperPrevious type="button">Back</button>
        <button
          mat-button
          matStepperNext
          [disabled]="selectedSeats! < 1"
          type="button"
        >
          next
        </button>
      </div>
    </mat-step>
    <mat-step>
      <ng-template matStepLabel>Review</ng-template>
      <table class="table table-striped">
        <tr>
          <th scope="col">Auditorium</th>
          <td>{{ ticketsForm.get("auditoriumName")?.value! }}</td>
        </tr>
        <tr>
          <th scope="col">Show</th>
          <td>
            {{
              ticketsForm.get("showName")?.value! +
                " | Time: " +
                formatTime(selectMembers?.showTime!)
            }}
          </td>
        </tr>
        <tr>
          <th scope="col">Date</th>
          <td>{{ ticketsForm.get("date")?.value! | date: "fullDate" }}</td>
        </tr>
        <tr>
          <th scope="col">Seats</th>
          <td>{{ ticketsForm.get("seats")?.value! }}</td>
        </tr>
      </table>
      <div>
        <button mat-button matStepperPrevious type="button">Back</button>
        <button mat-button color="primary" [disabled]="!ticketsForm.valid">
          Proceed
          </button>
        </div>
      </mat-step>
    </mat-horizontal-stepper>
  </form>
</mat-card>
```

Show Form:

```
<div class="wrapper">
  <h1 class="text-center mb-3 p-2 rounded border border-2 shadow-sm">
    Add New Show
  </h1>
  <form class="example-form p-4" [formGroup]="showForm">
    <mat-form-field class="d-block w-100 text-dark" appearance="outline">
      <mat-label>Show Name</mat-label>
      <input matInput formControlName="name" list="show-name" />
      <datalist id="show-name">
        <option *ngFor="let show of showNames" [value]="show">
          {{ show }}
        </option>
      </datalist>
      <mat-error *ngIf="nameErrors">
        {{ nameErrors }}
      </mat-error>
    </mat-form-field>

    <mat-form-field class="d-block w-100 text-dark" appearance="outline">
      <mat-label>Show Timing</mat-label>
      <input
        type="time"
        matInput
        placeholder="1H 20M"
        formControlName="startTime"
      />
      <mat-error *ngIf="startTimeErrors" color="accent"
        >{{ startTimeErrors }}
      </mat-error>
    </mat-form-field>
  </form>
  <button mat-raised-button (click)="onCancel()" class="me-2">Cancel</button>
  <button
    mat-raised-button
    (click)="onSubmit()"
    [disabled]="!showForm.valid"
    color="primary"
    class="ms-2"
  >
    Submit
  </button>
</div>
```

INDEX.HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>My Movie Plan</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />

    <!-- Bootstrap Css -->
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.mi
n.css"
      rel="stylesheet"
      integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x"
      crossorigin="anonymous"
    />

    <!-- Animated CSS -->
    <!-- <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.m
in.css"
      crossorigin="anonymous"
    /> -->
    <link rel="preconnect" href="https://fonts.gstatic.com" />
    <link
      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&d
isplay=swap"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet"
    />
    <link rel="preconnect" href="https://fonts.gstatic.com" />
    <link
```

```
      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&d
isplay=swap"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet"
    />
    <link rel="preconnect" href="https://fonts.gstatic.com" />
    <link
      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&d
isplay=swap"
      rel="stylesheet"
    />
    <link
      href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet"
    />
  </head>
  <body class="mat-typography">
    <app-root></app-root>

    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bund
le.min.js"
      integrity="sha384-
gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4"
      crossorigin="anonymous"
    ></script>

    <script type="text/javascript">
      const myCarousel = document.querySelector("#movies-carousel");
      if (myCarousel) {
        const carousel = new bootstrap.Carousel(myCarousel, {
          interval: 2000,
          wrap: false,
        });
      }
    </script>
  </body>
</html>
```

DOCKER FILE:

```
FROM node:current-alpine3.11 as build-stage
```

```
# RUN mkdir -p /app

WORKDIR /usr/src/app

COPY package.json package-lock.json ./

RUN npm install

COPY . .

RUN npm run build --prod

# CMD ["npm", "start"]

FROM nginx:latest-alpine as prod-stage

COPY --from=build-stage /usr/src/app/dist/my-movie-plan /user/share/nginx/html

COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 4040
```

JENKINS FILES:

```
pipeline {
    agent any
    stages {
        stage('git repo clone') {
            steps {
                git branch: 'main', url: 'https://github.com/JRiyaz/my-movie-
plan.git'
            }
        }
        // stage('clean') {
        //     steps {
        //         sh "mvn clean"
        //     }
        // }
        // stage('package') {
        //     steps {
        //         sh "mvn package"
        //     }
        // }
        // stage('docker build') {
        //     steps {
        //         sh "docker build -t employee-management ."
        //      }
        // }
        stage('docker compose build') {
             steps {
                 sh "docker-compose build"
             }
        }

        stage('docker compose start') {
             steps {
                 sh "docker-compose up"
             }
        }
    }
}
```

KARMA CONFIG:

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular-devkit/build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage'),
      require('@angular-devkit/build-angular/plugins/karma')
    ],
    client: {
      jasmine: {
        // you can add configuration options for Jasmine here
        // the possible options are listed at
https://jasmine.github.io/api/edge/Configuration.html
        // for example, you can disable the random execution with `random:
false`
        // or set a specific seed with `seed: 4321`
      },
      clearContext: false // leave Jasmine Spec Runner output visible in
browser
    },
    jasmineHtmlReporter: {
      suppressAll: true // removes the duplicated traces
    },
    coverageReporter: {
      dir: require('path').join(__dirname, './coverage/my-movie-plan'),
      subdir: '.',
      reporters: [
        { type: 'html' },
        { type: 'text-summary' }
      ]
    },
    reporters: ['progress', 'kjhtml'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false,
    restartOnFileChange: true
  });
};
```

BACK END

CONFIG:

Initial Data:

```
package com.MyMoviePlan.config;

import com.MyMoviePlan.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

@Component
public class InitialData implements CommandLineRunner {

    @Autowired
    private UserService service;
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Override
    public void run(String... args) throws Exception {

//        final UserEntity super_admin = new UserEntity("Riyaz J",
"j.riyazu@gmail.com",
//                "8099531318", "Male",
passwordEncoder.encode("super"), true,
//                true, true, true, true,
//                ROLE_SUPER_ADMIN);
//
//        final UserEntity admin = new UserEntity("Fayaz J",
"j.fayaz@gmail.com",
//                "9019168638", "Male",
passwordEncoder.encode("admin"), true,
//                true, true, true, true,
//                ROLE_ADMIN);
//
//        final UserEntity user = new UserEntity("Inthiyaz J",
"j.inthiyaz@gmail.com",
//                "8985462507", "Male",
passwordEncoder.encode("user"), true,
//                true, true, true, true,
//                ROLE_USER);
//
//        service.save(super_admin);
//        service.save(admin);
//        service.save(user);
    }
}
```

CONTROLLERS:

Auditorium Controller:

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.*;
import com.MyMoviePlan.exception.AuditoriumNotFoundException;
import com.MyMoviePlan.exception.BookingNotFoundException;
import com.MyMoviePlan.exception.MovieShowNotFoundException;
import com.MyMoviePlan.exception.ShowNotFoundException;
import com.MyMoviePlan.model.TicketDetails;
import com.MyMoviePlan.model.UserRole;
import com.MyMoviePlan.repository.*;
import com.MyMoviePlan.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.stream.Collectors;

@CrossOrigin
@RestController
```

```
@RequestMapping("/auditorium")
@AllArgsConstructor
public class AuditoriumController {

    private final ShowRepository show;
    private final UserService service;
    private final BookingRepository booking;
    private final MovieRepository movie;
    private final MovieShowsRepository movieShow;
    private final AuditoriumRepository auditorium;

    @GetMapping({"/", "all"})
    public List<AuditoriumEntity> findAllAuditoriums() {
        return this.auditorium.findAll();
    }

    @GetMapping("{auditorium_id}")
    @PreAuthorize("hasAuthority('WRITE')")
    public AuditoriumEntity findAuditoriumById(@PathVariable final
int auditorium_id) {
        return this.auditorium.findById(auditorium_id)
                .orElseThrow(() ->
                        new AuditoriumNotFoundException("Auditorium
with id: " + auditorium_id + " not found."));
    }

    @PostMapping("add")
    @PreAuthorize("hasAuthority('WRITE')")
    public AuditoriumEntity saveAuditorium(@RequestBody final
AuditoriumEntity auditorium) {
        return this.auditorium.save(auditorium);
    }

    @PutMapping("update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public AuditoriumEntity updateAuditorium(@RequestBody final
AuditoriumEntity auditorium) {
        return this.auditorium.save(auditorium);
    }

    @DeleteMapping("delete/{auditorium_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteAuditorium(@PathVariable final int
auditorium_id) {
        this.auditorium.deleteById(auditorium_id);
    }

    /*
     *   ============================= Show Controller
==========================
     */

    @GetMapping("{auditorium_id}/show/{show_id}")
    public ShowEntity findShowById(@PathVariable final int
auditorium_id,
                                   @PathVariable final int show_id)
{
        return this.findAuditoriumById(auditorium_id).getShows()
                .stream()
                .filter(show -> show.getId() == show_id)
                .findFirst()
                .orElseThrow(() ->
                        new ShowNotFoundException("Show with Id: " +
show_id + " not found"));
    }

    @GetMapping("{auditorium_id}/show/all")
    public List<ShowEntity> findAllShows(@PathVariable final int
auditorium_id) {
        return this.findAuditoriumById(auditorium_id).getShows();
    }

    @PostMapping("{auditorium_id}/show/add")
    @PreAuthorize("hasAuthority('WRITE')")
    public ShowEntity saveShow(@PathVariable final int
auditorium_id,
                               @RequestBody final ShowEntity show) {
        final AuditoriumEntity auditorium =
this.findAuditoriumById(auditorium_id);
        show.setAuditorium(auditorium);
        return this.show.save(show);
    }

    @PutMapping("{auditorium_id}/show/update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public ShowEntity updateShow(@PathVariable final int
auditorium_id,
                                 @RequestBody final ShowEntity show)
{
        final AuditoriumEntity auditorium =
this.findAuditoriumById(auditorium_id);
        show.setAuditorium(auditorium);
        return this.show.save(show);
    }

    @DeleteMapping("{auditorium_id}/show/delete/{show_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteShow(@PathVariable final int auditorium_id,
                           @PathVariable final int show_id) {
        final ShowEntity show = this.findShowById(auditorium_id,
show_id);
        this.show.deleteById(show.getId());
    }

    /*
     *   ============================= Movie Show Controller
==========================
     */

    @GetMapping("movie/{movieId}")
    public List<AuditoriumEntity>
findAuditoriumsByMovieId(@PathVariable final int movieId) {
        return this.findAllAuditoriums().stream()
                .filter(halls -> halls.getShows()
                        .stream()
                        .anyMatch(show -> show.getMovieShows()
                                .stream()
                                .anyMatch(m_show ->
m_show.getMovieId() == movieId)))
                .collect(Collectors.toList());
    }

    @GetMapping("{auditorium_id}/movie/{movieId}")
    public List<ShowEntity> findShowsByMovieId(@PathVariable final
int auditorium_id, @PathVariable final int movieId) {
        return this.findAllShows(auditorium_id).stream()
                .filter(show -> show.getMovieShows()
                        .stream()
                        .anyMatch(m_show -> m_show.getMovieId() ==
movieId))
                .collect(Collectors.toList());
    }

    @GetMapping("{auditorium_id}/show/{show_id}/movie-show/all")
    public List<MovieShowsEntity> findAllMovieShows(@PathVariable
final int auditorium_id,
                                                        @PathVariable
final int show_id) {
        return this.findShowById(auditorium_id, show_id)
                .getMovieShows();
    }

    @GetMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}")
    public MovieShowsEntity findMovieShowById(@PathVariable final
int auditorium_id,
                                              @PathVariable final
int show_id,
                                              @PathVariable final
int movie_show_id) {
        return this.findShowById(auditorium_id, show_id)
                .getMovieShows()
                .stream()
                .filter(movie_show -> movie_show.getId() ==
movie_show_id)
                .findFirst()
                .orElseThrow(
                        () -> new MovieShowNotFoundException("Movie
Show with id: "
                                + movie_show_id + " not found"));
    }

    @PostMapping("{auditorium_id}/show/{show_id}/movie-show/add")
    @PreAuthorize("hasAuthority('WRITE')")
    public MovieShowsEntity saveMovieShow(@PathVariable final int
auditorium_id,
                                          @PathVariable final int
show_id,
                                          @RequestBody final
MovieShowsEntity movieShow) {
        final ShowEntity show = this.findShowById(auditorium_id,
show_id);
        final int movieId = movieShow.getMovieId();
        movieShow.setShow(show);

movieShow.setMovieId(this.movie.findById(movieId).get().getId());
        return this.movieShow.save(movieShow);
    }

    @PutMapping("{auditorium_id}/show/{show_id}/movie-show/update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public MovieShowsEntity updateMovieShow(@PathVariable final int
auditorium_id,
                                            @PathVariable final int
show_id,
                                            @RequestBody final
MovieShowsEntity movieShow) {
        final ShowEntity show = this.findShowById(auditorium_id,
show_id);
        movieShow.setShow(show);
        return this.movieShow.save(movieShow);
    }

    @DeleteMapping("{auditorium_id}/show/{show_id}/movie-
show/delete/{movie_show_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteMovieShow(@PathVariable final int
auditorium_id,
                                @PathVariable final int show_id,
                                @PathVariable final int
movie_show_id) {
        final MovieShowsEntity movieShow =
this.findMovieShowById(auditorium_id, show_id, movie_show_id);
        this.movieShow.deleteById(movieShow.getMovieId());
    }

    /*
     *   ============================= Booking Controller
==========================
     */

    @GetMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}/booking/{booking_id}")
    @PreAuthorize("hasAuthority('READ')")
    public BookingEntity findBookingById(@PathVariable final int
auditorium_id,
                                         @PathVariable final int
show_id,
                                         @PathVariable final int
movie_show_id,
                                         @PathVariable final int
booking_id) {
        final MovieShowsEntity movieShow =
this.findMovieShowById(auditorium_id, show_id, movie_show_id);
        return movieShow.getBookings()
                .stream().filter(booking -> booking.getId() ==
booking_id)
                .findFirst()
                .orElseThrow(() -> new
BookingNotFoundException("Booking with id: "
                        + booking_id + " not found."));
    }

    @GetMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}/booking/all")
    @PreAuthorize("hasAuthority('WRITE')")
    public List<BookingEntity> allBookings(@PathVariable final int
auditorium_id,
                                           @PathVariable final int
show_id,
                                           @PathVariable final int
movie_show_id) {
        final UserEntity user = this.service.getLoggedInUser();
        if (user.getUserRole().equals(UserRole.ROLE_ADMIN) ||
user.getUserRole().equals(UserRole.ROLE_SUPER_ADMIN))
            return this.findMovieShowById(auditorium_id, show_id,
movie_show_id).getBookings();
        else
            return this.findMovieShowById(auditorium_id, show_id,
movie_show_id).getBookings()
                    .stream().filter(booking ->
booking.getUserId().equals(user.getId()))
                    .collect(Collectors.toList());
    }

    @PostMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}/booking/add")
//    @PreAuthorize("hasAuthority('WRITE')")
    public BookingEntity saveBooking(@PathVariable final int
auditorium_id,
                                     @PathVariable final int
show_id,
                                     @PathVariable final int
movie_show_id,
                                     @RequestBody final
BookingEntity booking) {
        final MovieShowsEntity moveShow =
this.findMovieShowById(auditorium_id, show_id, movie_show_id);
        booking.setUserId(this.service.getLoggedInUser().getId());
//
booking.setUserId(this.service.findByMobile("8099531318").get().getI
d());
        booking.setMovieShow(moveShow);
        booking.setBookingDetails(new
BookingDetailsEntity(auditorium_id, show_id, movie_show_id,
moveShow.getMovieId()));
        return this.booking.save(booking);
    }
    @PutMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}/booking/update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public BookingEntity updateBooking(@PathVariable final int
auditorium_id,
                                       @PathVariable final int
show_id,
                                       @PathVariable final int
movie_show_id,
                                       @RequestBody final
BookingEntity booking) {
        final MovieShowsEntity moveShow =
this.findMovieShowById(auditorium_id, show_id, movie_show_id);
        booking.setMovieShow(moveShow);
        return this.booking.save(booking);
    }

    @DeleteMapping("{auditorium_id}/show/{show_id}/movie-
show/{movie_show_id}/booking/delete/{booking_id}")
    @PreAuthorize("hasAuthority('READ')")
    public void deleteBookingById(@PathVariable final int
auditorium_id,
                                  @PathVariable final int show_id,
                                  @PathVariable final int
movie_show_id,
                                  @PathVariable final int
booking_id) {
        final BookingEntity booking =
this.findBookingById(auditorium_id, show_id, movie_show_id,
booking_id);
        this.booking.deleteById(booking.getId());
    }

    @GetMapping("ticket-details/{booking_id}")
    @PreAuthorize("hasAuthority('READ')")
    public TicketDetails getMovieDetails(@PathVariable final int
booking_id) {

        final PaymentEntity payment =
this.booking.findById(booking_id).get().getPayment();

        final MovieShowsEntity movieShow =
this.movieShow.findAll().stream().filter(m_show ->
m_show.getBookings()
                .stream().anyMatch(booking -> booking.getId() ==
booking_id)).findFirst().get();

        final MovieEntity movie =
this.movie.findById(movieShow.getMovieId()).get();

        final ShowEntity showEntity = show.findAll().stream()
                .filter(show -> show.getMovieShows()
                        .stream().anyMatch(m_show -> m_show.getId()
== movieShow.getId())).findFirst().get();
```

```
        final AuditoriumEntity auditorium =
this.auditorium.findAll().stream().filter(hall -> hall.getShows()
                .stream().anyMatch(show -> show.getId() ==
showEntity.getId())).findFirst().get();

        return new TicketDetails(auditorium.getName(),
showEntity.getName(), showEntity.getStartTime(),
payment.getAmount(), movie.getName(), movie.getImage(),
movie.getBgImage());
    }
}
```

Booking:

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.BookingDetailsEntity;
import com.MyMoviePlan.entity.BookingEntity;
import com.MyMoviePlan.entity.UserEntity;
import com.MyMoviePlan.exception.BookingNotFoundException;
import com.MyMoviePlan.model.UserRole;
import com.MyMoviePlan.repository.BookingRepository;
import com.MyMoviePlan.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin
@RestController
@RequestMapping("/booking")
@AllArgsConstructor
public class BookingController {

    private final BookingRepository repository;
    private final UserService service;

    @GetMapping("{id}")
    @PreAuthorize("hasAuthority('READ')")
    public BookingEntity findById(@PathVariable final int id) {
        return repository.findById(id)
                .orElseThrow(() -> new
BookingNotFoundException("Booking with id: " + id + " not found."));
    }

    @GetMapping("all")
    @PreAuthorize("hasAuthority('READ')")
    public List<BookingEntity> allBookings() {
        final UserEntity user = service.getLoggedInUser();
        if (user.getUserRole().equals(UserRole.ROLE_ADMIN) ||
user.getUserRole().equals(UserRole.ROLE_SUPER_ADMIN))
            return repository.findAll();
        else return
repository.findAllByUserIdOrderByBookedOnAsc(user.getId());
    }

    @GetMapping("{username}/all")
    @PreAuthorize("hasAuthority('READ')")
    public List<BookingEntity> findAllByUserId(@PathVariable String
username) {
        if (!(username.contains("-") && username.length() > 10))
            username = service.getUser(username).getId();
        return
repository.findAllByUserIdOrderByBookedOnAsc(username);
    }

    @DeleteMapping("delete/{id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteBooking(@PathVariable final int id) {
        repository.deleteById(id);
    }

    @GetMapping("{id}/details")
    @PreAuthorize("hasAuthority('READ')")
    public BookingDetailsEntity findByDetailsId(@PathVariable final
int id) {
        return this.findById(id).getBookingDetails();
    }
}
```

Movie:

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.MovieEntity;
import com.MyMoviePlan.exception.MovieNotFoundException;
import com.MyMoviePlan.repository.MovieRepository;
import com.MyMoviePlan.repository.MovieShowsRepository;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.*;

@CrossOrigin
@RestController
@RequestMapping("/movie")
@AllArgsConstructor
public class MovieController {

    private final MovieRepository movieRepository;
    private final MovieShowsRepository movieShowsRepository;

    @GetMapping({"/", "all"})
    public List<MovieEntity> findAll() {
        return movieRepository.findAll();
    }

    @GetMapping("{movie_id}")
    public MovieEntity findById(@PathVariable final int movie_id) {
        return movieRepository.findById(movie_id)
                .orElseThrow(() -> new MovieNotFoundException("Movie
with movie_id: " + movie_id + " not found."));
    }

    @GetMapping("up-coming")
    public List<MovieEntity> upComing(@RequestParam(value =
"records", required = false) Optional<String> records) {
        List<MovieEntity> movies;
        List<MovieEntity> allMovies;
        if (records.isPresent()) {
            movies = new ArrayList<>();
            allMovies = this.findAll();

movieShowsRepository.findFewUpComing(Integer.parseInt(records.get())
)
                    .forEach(m_show -> movies.add(allMovies.stream()
                            .filter(movie -> (movie.getId() ==
m_show.getMovieId() && movie.getRelease().getTime() > new
Date().getTime()))
                            .findFirst().orElse(null)));
        } else {
            movies = new ArrayList<>();
            allMovies = this.findAll();
            movieShowsRepository.findAllUpComing()
                    .forEach(m_show -> movies.add(allMovies.stream()
                            .filter(movie -> movie.getId() ==
m_show.getMovieId() && movie.getRelease().getTime() > new
Date().getTime())
                            .findFirst().orElse(null)));
        }
//        return (movies.size() > 0 && !movies.contains(null)) ?
movies : new ArrayList<>();
        movies.removeAll(Collections.singletonList(null));
        return movies;
    }

    @GetMapping("now-playing")
    public List<MovieEntity> nowPlaying(@RequestParam(value =
"records", required = false) Optional<String> records) {
        List<MovieEntity> movies;
        List<MovieEntity> allMovies;
        if (records.isPresent()) {
            movies = new ArrayList<>();
            allMovies = this.findAll();

movieShowsRepository.findFewNowPlaying(Integer.parseInt(records.get(
)))
                    .forEach(m_show -> movies.add(allMovies.stream()
                            .filter(movie -> movie.getId() ==
m_show.getMovieId())
```

```
                            .findFirst().orElse(null)));
        } else {
            movies = new ArrayList<>();
            allMovies = this.findAll();
            movieShowsRepository.findAllNowPlaying()
                    .forEach(m_show -> movies.add(allMovies.stream()
                            .filter(movie -> movie.getId() ==
m_show.getMovieId())
                            .findFirst().orElse(null)));
        }
        movies.removeAll(Collections.singletonList(null));
        return movies;
    }

    @GetMapping("now-playing-up-coming")
    public List<MovieEntity> nowPlayingAndUpComing() {
        final List<MovieEntity> movies = new ArrayList<>();
        final List<MovieEntity> allMovies = this.findAll();
        movieShowsRepository.findAllNowPlayingAndUpComing()
                .forEach(m_show -> movies.add(allMovies.stream()
                        .filter(movie -> movie.getId() ==
m_show.getMovieId())
                        .findFirst().orElse(null)));
        movies.removeAll(Collections.singletonList(null));
        return movies;
    }

    @GetMapping("not-playing")
    public List<MovieEntity> notPlaying() {
        final List<MovieEntity> movies = new ArrayList<>();
        final List<MovieEntity> allMovies = this.findAll();
        movieShowsRepository.findAllNotPlaying()
                .forEach(m_show -> movies.add(allMovies.stream()
                        .filter(movie -> movie.getId() ==
m_show.getMovieId())
                        .findFirst().orElse(null)));
        movies.removeAll(Collections.singletonList(null));
        return movies;
    }

    @PostMapping("add")
    @PreAuthorize("hasAuthority('WRITE')")
    public MovieEntity saveMovie(@RequestBody final MovieEntity
movie) {
        return movieRepository.save(movie);
    }

    @PutMapping("update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public MovieEntity updateMovie(@RequestBody final MovieEntity
movie) {
        return movieRepository.save(movie);
    }

    @DeleteMapping("delete/{movie_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteMovie(@PathVariable final int movie_id) {
        movieRepository.deleteById(movie_id);
    }
}
```

Movie Show :

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.BookingEntity;
import com.MyMoviePlan.entity.MovieShowsEntity;
import com.MyMoviePlan.exception.MovieShowNotFoundException;
import com.MyMoviePlan.model.BookedSeats;
import com.MyMoviePlan.repository.MovieShowsRepository;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

@CrossOrigin
@RestController
@RequestMapping("/movie-show")
@AllArgsConstructor
public class MovieShowController {

    private final MovieShowsRepository repository;

    @PostMapping("add")
    @PreAuthorize("hasAuthority('WRITE')")
    public MovieShowsEntity save(@RequestBody MovieShowsEntity
movieShow) {
        return repository.save(movieShow);
    }

    @GetMapping("up-coming")
    @PreAuthorize("hasAuthority('READ')")
    public List<MovieShowsEntity> upComing(@RequestParam(value =
"records", required = false) Optional<String> records) {
        if (records.isPresent())
            return
repository.findFewUpComing(Integer.parseInt(records.get()));
        return repository.findAllUpComing();
    }

    @GetMapping("now-playing")
    public List<MovieShowsEntity> nowPlaying(@RequestParam(value =
"records", required = false) Optional<String> records) {
        if (records.isPresent())
```

```
            return
repository.findFewNowPlaying(Integer.parseInt(records.get()));
        return repository.findAllNowPlaying();
    }

    @GetMapping("now-playing-up-coming")
    public List<MovieShowsEntity> nowPlayingAndUpComing() {
        return repository.findAllNowPlayingAndUpComing();
    }

    @GetMapping("not-playing")
    @PreAuthorize("hasAuthority('WRITE')")
    public List<MovieShowsEntity> notPlaying() {
        return repository.findAllNotPlaying();
    }

    @GetMapping("all")
    public List<MovieShowsEntity> findAllMovieShows() {
        return repository.findAll();
    }

    @GetMapping("{movie_show_id}")
    public MovieShowsEntity findMovieShowById(@PathVariable final
int movie_show_id) {
        return repository.findById(movie_show_id)
                .orElseThrow(
                        () -> new MovieShowNotFoundException("Movie
Show with id: " + movie_show_id + " not found")
                );
    }

    @DeleteMapping("delete/{movie_show_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteMovieShow(@PathVariable final int
movie_show_id) {
        repository.deleteById(movie_show_id);
    }


    /*
     *   ============================= Booking Controller
==========================
     */

    @GetMapping("{movie_show_id}/booked-seats/{on}")
    @PreAuthorize("hasAuthority('READ')")
    public BookedSeats bookedSeats(@PathVariable final int
movie_show_id, @PathVariable final String on) {
        final List<BookingEntity> bookings =
this.findMovieShowById(movie_show_id).getBookings()
                .stream().filter(m_show ->
m_show.getDateOfBooking().toString().equals(on))
                .collect(Collectors.toList());

        int count = 0;
        List<String> seats = new ArrayList<>();
        for (BookingEntity booking : bookings) {
            count += booking.getTotalSeats();
            seats.addAll(booking.getSeatNumbers());
        }
        return new BookedSeats(count, seats);
    }
}
```

Show:

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.BookingEntity;
import com.MyMoviePlan.entity.MovieShowsEntity;
import com.MyMoviePlan.entity.ShowEntity;
import com.MyMoviePlan.exception.BookingNotFoundException;
import com.MyMoviePlan.exception.MovieShowNotFoundException;
import com.MyMoviePlan.exception.ShowNotFoundException;
import com.MyMoviePlan.repository.BookingRepository;
import com.MyMoviePlan.repository.MovieRepository;
import com.MyMoviePlan.repository.MovieShowsRepository;
import com.MyMoviePlan.repository.ShowRepository;
import com.MyMoviePlan.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@CrossOrigin
@RestController
@RequestMapping("/show")
@AllArgsConstructor
public class ShowController {

    private final ShowRepository show;
    private final MovieShowsRepository movieShow;
    private final MovieRepository movie;
    private final UserService service;
    private final BookingRepository booking;

    @GetMapping("{show_id}")
    public ShowEntity findShowById(@PathVariable final int show_id)
{
        return this.show.findById(show_id)
                .orElseThrow(() -> new ShowNotFoundException("Show
with Id: " + show_id + " not found"));
    }

    @GetMapping({"/", "all"})
    public List<ShowEntity> findAllShows() {
        return this.show.findAll();
    }
    @DeleteMapping("delete/{show_id}")
    @PreAuthorize("hasAuthority('DELETE')")
    public void deleteShow(@PathVariable final int show_id) {
        this.show.deleteById(show_id);
    }


    /*
     *   ============================= Movie Show Controller
==========================
     */

    @GetMapping("{show_id}/movie-show/all")
    public List<MovieShowsEntity> findAllMovieShows(@PathVariable
final int show_id) {
        return this.findShowById(show_id)
                .getMovieShows();
    }

    @GetMapping("{show_id}/movie-show/{movie_show_id}")
    public MovieShowsEntity findMovieShowById(@PathVariable final
int show_id,
                                              @PathVariable final
int movie_show_id) {
        return this.findShowById(show_id)
                .getMovieShows()
                .stream()
                .filter(movie_show -> movie_show.getId() ==
movie_show_id)
                .findFirst()
                .orElseThrow(
                        () -> new MovieShowNotFoundException("Movie
Show with id: "
                                + movie_show_id + " not found"));
    }

    @PostMapping("{show_id}/movie-show/add")
    @PreAuthorize("hasAuthority('WRITE')")
    public MovieShowsEntity saveMovieShow(@PathVariable final int
show_id,
                                          @RequestBody final
MovieShowsEntity movieShow) {
        final ShowEntity show = this.findShowById(show_id);
        final int movieId = movieShow.getMovieId();
        movieShow.setShow(show);

movieShow.setMovieId(this.movie.findById(movieId).get().getId());
        return this.movieShow.save(movieShow);
    }

    @PutMapping("{show_id}/movie-show/update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public MovieShowsEntity updateMovieShow(@PathVariable final int
show_id,
                                            @RequestBody final
MovieShowsEntity movieShow) {
        final ShowEntity show = this.findShowById(show_id);
        movieShow.setShow(show);
        return this.movieShow.save(movieShow);
    }

    @DeleteMapping("{show_id}/movie-show/delete/{movie_show_id}")
    @PreAuthorize("hasAuthority('UPDATE')")
    public void deleteMovieShow(@PathVariable final int show_id,
                                @PathVariable final int
movie_show_id) {
        final MovieShowsEntity movieShow =
this.findMovieShowById(show_id, movie_show_id);
        this.movieShow.deleteById(movieShow.getMovieId());
    }

    /*
     *   ============================= Booking Controller
==========================
     */

    @GetMapping("{show_id}/movie-
show/{movie_show_id}/booking/{booking_id}")
    @PreAuthorize("hasAuthority('READ')")
    public BookingEntity findBookingById(@PathVariable final int
show_id,
                                         @PathVariable final int
movie_show_id,
                                         @PathVariable final int
booking_id) {
        final MovieShowsEntity movieShow =
this.findMovieShowById(show_id, movie_show_id);
        return movieShow.getBookings()
                .stream().filter(booking -> booking.getId() ==
booking_id)
                .findFirst()
                .orElseThrow(() -> new
BookingNotFoundException("Booking with id: "
                        + booking_id + " not found."));
    }

    @GetMapping("{show_id}/movie-show/{movie_show_id}/booking/all")
    @PreAuthorize("hasAuthority('READ')")
    public List<BookingEntity> allBookings(@PathVariable final int
show_id,
                                           @PathVariable final int
movie_show_id) {
        return this.findMovieShowById(show_id,
movie_show_id).getBookings();
    }

    @PostMapping("{show_id}/movie-show/{movie_show_id}/booking/add")
    @PreAuthorize("hasAuthority('WRITE')")
    public BookingEntity saveBooking(@PathVariable final int
show_id,
                                     @PathVariable final int
movie_show_id,
                                     @RequestBody final
BookingEntity booking) {
        final MovieShowsEntity moveShow =
this.findMovieShowById(show_id, movie_show_id);
//        booking.setUserId(this.service.getLoggedInUser().getId());

booking.setUserId(this.service.findByMobile("8099531318").get().getI
d());
        booking.setMovieShow(moveShow);
        return this.booking.save(booking);
    }

    @PutMapping("{show_id}/movie-
show/{movie_show_id}/booking/update")
    @PreAuthorize("hasAuthority('UPDATE')")
    public BookingEntity updateBooking(@PathVariable final int
show_id,
                                       @PathVariable final int
movie_show_id,
                                       @RequestBody final
BookingEntity booking) {
        final MovieShowsEntity moveShow =
this.findMovieShowById(show_id, movie_show_id);
        booking.setMovieShow(moveShow);
        return this.booking.save(booking);
    }

    @DeleteMapping("{show_id}/movie-
show/{movie_show_id}/booking/delete/{booking_id}")
    @PreAuthorize("hasAuthority('READ')")
    public void deleteBookingById(@PathVariable final int show_id,
                                  @PathVariable final int
movie_show_id,
                                  @PathVariable final int
booking_id) {
        final BookingEntity booking = this.findBookingById(show_id,
movie_show_id, booking_id);
        this.booking.deleteById(booking.getId());
    }
}
```

User:

```
package com.MyMoviePlan.controller;

import com.MyMoviePlan.entity.UserEntity;
import com.MyMoviePlan.model.Credentials;
import com.MyMoviePlan.model.HttpResponse;
import com.MyMoviePlan.model.Token;
import com.MyMoviePlan.service.UserService;
import lombok.AllArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpServletRequest;
import java.util.List;

@CrossOrigin
@RestController
@RequestMapping("/user")
@AllArgsConstructor
public class UserController {

    private final UserService service;
    private final HttpServletRequest request;

    @GetMapping("/")
    public String index() {
        return "Welcome " + service.getUserName();
    }

    @PostMapping("authenticate")
    public Token authenticate(@RequestBody final Credentials
credentials) {

        return service.authenticate(credentials);
    }

    @GetMapping("check/{username}")
    public Token checkUniqueness(@PathVariable final String
username) {
        return service.checkUniqueness(username);
    }

    @GetMapping("get-user")
    @PreAuthorize("hasAuthority('READ')")
    public UserEntity user() {
        return service.getLoggedInUser()
                .setPassword(null);
    }

    @GetMapping("all")
    @PreAuthorize("hasAuthority('WRITE')")
    public List<UserEntity> allUsers() {
        return service.findAll();
    }

    @PutMapping("update/{username}")
    @PreAuthorize("hasAuthority('READ')")
    public UserEntity updateUser(@RequestBody final UserEntity
userEntity,
                                 @PathVariable final String
username) {

        return service.update(userEntity, username);
    }
    @PostMapping("sign-up")
    public HttpResponse signUp(@RequestBody final UserEntity
userEntity) {

        return service.register(userEntity);
    }

    @PutMapping("change-password")
    @PreAuthorize("hasAuthority('READ')")
    public HttpResponse changePassword(@RequestBody final
Credentials credentials) {

        return service.changePassword(credentials);
    }

    @PutMapping("forgot-password")
    public HttpResponse forgotPassword(@RequestBody final
Credentials credentials) {
        return service.forgotPassword(credentials);
    }

    @DeleteMapping("delete/{username}")
    @PreAuthorize("hasAuthority('DELETE')")
    public HttpResponse delete(@PathVariable final String username)
{
        return service.deleteById(username);
    }
}
```

ENTITY:

Actor:

```
package com.MyMoviePlan.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.*;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "actors")
public class ActorEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column(name = "is_cast")
    private String isCast;

    private String name;

    private String role;

    @Column(length = Integer.MAX_VALUE, columnDefinition="TEXT")
    private String image;

      @JsonIgnore
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ManyToOne(targetEntity = MovieEntity.class)
    private MovieEntity movie;

    public ActorEntity(String name, String role, String image) {
        this.name = name;
        this.role = role;
        this.image = image;
    }
}
```

Auditorium:

```
package com.MyMoviePlan.entity;

import lombok.*;

import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

//@JsonIdentityInfo(generator =
ObjectIdGenerators.PropertyGenerator.class,
//        property = "id", scope = ShowEntity.class)
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@EqualsAndHashCode
@Table(name = "auditoriums")
public class AuditoriumEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    @Column(length = Integer.MAX_VALUE, columnDefinition="TEXT")
    private String image;
    private String email;

    @Column(name = "customer_care_no")
    private String customerCareNo;

    private String address;

    @Column(name = "seat_capacity")
    private int seatCapacity;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ElementCollection
    @CollectionTable(name = "auditorium_facilities", joinColumns =
@JoinColumn(name = "auditorium_id"))
    @Column(name = "facility")
    private List<String> facilities;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ElementCollection
    @CollectionTable(name = "auditorium_safeties", joinColumns =
@JoinColumn(name = "auditorium_id"))
    @Column(name = "safety")
    private List<String> safeties;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @JoinColumn(name = "auditorium_id", referencedColumnName = "id")
    @OneToMany(targetEntity = ShowEntity.class, cascade =
CascadeType.ALL)
//    @JoinTable(name = "auditorium_shows",
//            joinColumns = @JoinColumn(name = "auditorium_id",
unique = false),
//            inverseJoinColumns = @JoinColumn(name = "show_id",
unique = false))
    private List<ShowEntity> shows;

    public AuditoriumEntity(String name, String image, String email,
String customerCareNo, String address,
                            int seatCapacity, List<String>
facilities, List<String> safeties, List<ShowEntity> shows) {
        this.name = name;
        this.image = image;
        this.email = email;
        this.customerCareNo = customerCareNo;
        this.address = address;
        this.seatCapacity = seatCapacity;
        this.facilities = facilities;
        this.safeties = safeties;
        this.shows = shows;
    }

    public AuditoriumEntity setId(int id) {
        this.id = id;
        return this;
    }

    public AuditoriumEntity setName(String name) {
        this.name = name;
        return this;
    }

    public AuditoriumEntity setImage(String image) {
        this.image = image;
        return this;
    }

    public AuditoriumEntity setEmail(String email) {
        this.email = email;
        return this;
    }

    public AuditoriumEntity setCustomerCare(String customerCareNo) {
        this.customerCareNo = customerCareNo;
        return this;
    }

    public AuditoriumEntity setAddress(String address) {
        this.address = address;
        return this;
    }

    public AuditoriumEntity setSeatCapacity(int seatCapacity) {
        this.seatCapacity = seatCapacity;
        return this;
    }

    public AuditoriumEntity setFacilities(List<String> facilities) {
        this.facilities = facilities;
        return this;
    }

    public AuditoriumEntity setSafeties(List<String> safeties) {
        this.safeties = safeties;
        return this;
    }

    public AuditoriumEntity setShows(List<ShowEntity> shows) {
        this.shows = shows;
        return this;
    }
}
```

Booking Digitals:

```
package com.MyMoviePlan.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "booking_details")
public class BookingDetailsEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "auditorium_id")
    private int auditoriumId;

    @Column(name = "show_id")
    private int showId;

    @Column(name = "movie_show_id")
    private int movieShowId;

    @Column(name = "movie_id")
    private int movieId;

    public BookingDetailsEntity(int auditoriumId, int showId, int
movieShowId, int movieId) {
        this.auditoriumId = auditoriumId;
        this.showId = showId;
        this.movieShowId = movieShowId;
        this.movieId = movieId;
    }
}
```

Booking:

```
package com.MyMoviePlan.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.*;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
```

```
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "bookings")
public class BookingEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private double amount;

    @Column(name = "total_seats")
    private int totalSeats;

    @Column(name = "booked_on")
    @Temporal(TemporalType.DATE)
    private Date bookedOn;

    @Column(name = "date_of_booking")
    @Temporal(TemporalType.DATE)
    private Date dateOfBooking;

    @Column(name = "user_id")
    private String userId;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ElementCollection
    @CollectionTable(name = "booked_seats", joinColumns =
@JoinColumn(name = "booking_id"))
    @Column(name = "seat_numbers")
    private List<String> seatNumbers;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToOne(targetEntity = PaymentEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "payment_id")
    private PaymentEntity payment;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToOne(targetEntity = BookingDetailsEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "booking_details_id")
    private BookingDetailsEntity bookingDetails;

    @JsonIgnore
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ManyToOne(targetEntity = MovieShowsEntity.class)
    private MovieShowsEntity movieShow;

    public BookingEntity(double amount, int totalSeats, Date
bookedOn, Date dateOfBooking, List<String> seatNumbers,
```

```java
                         PaymentEntity payment, String userId,
MovieShowsEntity movieShow) {
        this.amount = amount;
        this.totalSeats = totalSeats;
        this.bookedOn = bookedOn;
        this.dateOfBooking = dateOfBooking;
        this.seatNumbers = seatNumbers;
        this.payment = payment;
        this.userId = userId;
        this.movieShow = movieShow;
    }

    public BookingEntity setMovieShow(MovieShowsEntity movieShow) {
        this.movieShow = movieShow;
        return this;
    }

    public BookingEntity setId(int id) {
        this.id = id;
        return this;
    }

    public BookingEntity setAmount(double amount) {
        this.amount = amount;
        return this;
    }

    public BookingEntity setTotalSeats(int totalSeats) {
        this.totalSeats = totalSeats;
        return this;
    }

    public BookingEntity setStatus(Date bookedOn) {
        this.bookedOn = bookedOn;
        return this;
    }

    public BookingEntity setDateOfBooking(Date dateOfBooking) {
        this.dateOfBooking = dateOfBooking;
        return this;
    }

    public BookingEntity setSeatNumbers(List<String> seatNumbers) {
        this.seatNumbers = seatNumbers;
        return this;
    }

    public BookingEntity setPayment(PaymentEntity payment) {
        this.payment = payment;
        return this;
    }

    public BookingEntity setUserId(String userId) {
        this.userId = userId;
        return this;
    }
```

```
}
```

Movie:

```
package com.MyMoviePlan.entity;

import lombok.*;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "movies")
public class MovieEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    @Column(length = Integer.MAX_VALUE, columnDefinition = "TEXT")
    private String image;

    @Column(name = "bg_image", length = Integer.MAX_VALUE,
columnDefinition="TEXT")
    private String bgImage;

    @Column(length = 9000)
    private String story;

    private String year;

    private String duration;

    private String caption;

    @Column(name = "added_on")
    @Temporal(TemporalType.DATE)
    private Date addedOn;

    @Temporal(TemporalType.DATE)
    private Date release;

    private String language;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ElementCollection
    @CollectionTable(name = "movie_genres", joinColumns =
@JoinColumn(name = "movie_id"))
    @Column(name = "genre")
    private List<String> genres;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToMany(targetEntity = ActorEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "movie_id", referencedColumnName = "id")
    private List<ActorEntity> casts;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToMany(targetEntity = ActorEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "movie_id", referencedColumnName = "id")
    private List<ActorEntity> crews;

    public MovieEntity(String name, String image, String bgImage,
String story, String year,
                       String duration, String caption, Date
addedOn, Date release, String language,
                       List<String> genres, List<ActorEntity> casts,
List<ActorEntity> crews) {
        this.name = name;
        this.image = image;
        this.bgImage = bgImage;
        this.story = story;
        this.year = year;
        this.duration = duration;
        this.caption = caption;
        this.addedOn = addedOn;
        this.release = release;
        this.language = language;
        this.genres = genres;
        this.casts = casts;
        this.crews = crews;
    }

    public MovieEntity setId(int id) {
        this.id = id;
        return this;
    }

    public MovieEntity setName(String name) {
        this.name = name;
        return this;
    }

    public MovieEntity setImage(String image) {
        this.image = image;
        return this;
    }
    public MovieEntity setBgImage(String bgImage) {
        this.bgImage = bgImage;
        return this;
    }

    public MovieEntity setStory(String story) {
        this.story = story;
        return this;
    }

    public MovieEntity setYear(String year) {
        this.year = year;
        return this;
    }

    public MovieEntity setDuration(String duration) {
        this.duration = duration;
        return this;
    }

    public MovieEntity setCaption(String caption) {
        this.caption = caption;
        return this;
    }

    public MovieEntity setAddedOn(Date addedOn) {
        this.addedOn = addedOn;
        return this;
    }

    public MovieEntity setRelease(Date release) {
        this.release = release;
        return this;
    }

    public MovieEntity setLanguages(String language) {
        this.language = language;
        return this;
    }

    public MovieEntity setGenres(List<String> genres) {
        this.genres = genres;
        return this;
    }

    public MovieEntity setCasts(List<ActorEntity> casts) {
        this.casts = casts;
        return this;
    }

    public MovieEntity setCrews(List<ActorEntity> crews) {
        this.crews = crews;
        return this;
    }
}
```

Movie Show:

```
package com.MyMoviePlan.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.*;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "movie_shows")
public class MovieShowsEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Temporal(TemporalType.DATE)
    @Column(name = "show_start")
    private Date start;

    @Temporal(TemporalType.DATE)
    @Column(name = "show_end")
    private Date end;

    @Column(name = "movie_id")
    private int movieId;

    @JsonIgnore
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ManyToOne(targetEntity = ShowEntity.class)
    private ShowEntity show;

    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @JoinColumn(name = "movie_show_id", referencedColumnName = "id")
    @OneToMany(targetEntity = BookingEntity.class, cascade =
CascadeType.ALL)
//    @JoinTable(name = "movie_show_bookings",
//            joinColumns = @JoinColumn(name = "movie_show_id",
unique = false),
//            inverseJoinColumns = @JoinColumn(name = "booking_id",
unique = false))
    private List<BookingEntity> bookings;
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToOne(targetEntity = PriceEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "price_id")
    private PriceEntity price;

    public MovieShowsEntity(int id, Date start, Date end,
List<BookingEntity> bookings, int movieId) {
        this.id  = id;
        this.start = start;
        this.end = end;
        this.bookings = bookings;
        this.movieId = movieId;
    }

    public MovieShowsEntity setId(int id) {
        this.id = id;
        return this;
    }

    public MovieShowsEntity setStart(Date start) {
        this.start = start;
        return this;
    }

    public MovieShowsEntity setEnd(Date end) {
        this.end = end;
        return this;
    }

    public MovieShowsEntity setShow(ShowEntity show) {
        this.show = show;
        return this;
    }

    public MovieShowsEntity setMovieId(int movieId) {
        this.movieId = movieId;
        return this;
    }
}
```

Payment:

```
package com.MyMoviePlan.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "payments")
public class PaymentEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private double amount;

    @Column(name = "payment_date")
    @Temporal(TemporalType.DATE)
    private Date paymentDate;

    @Column(name = "card_number", length = 20)
    private String cardNumber;

    @Column(name = "card_expiry_month", length = 5)
    private String cardExpiryMonth;

    @Column(name = "card_expiry_year", length = 5)
    private String cardExpiryYear;

    @Column(name = "card_cvv", length = 5)
    private String cardCVV;

    public PaymentEntity(double amount, Date paymentDate, String
cardNumber, String cardExpiryMonth,
                         String cardExpiryYear, String cardCVV) {
        this.amount = amount;
        this.paymentDate = paymentDate;
        this.cardNumber = cardNumber;
        this.cardExpiryMonth = cardExpiryMonth;
        this.cardExpiryYear = cardExpiryYear;
        this.cardCVV = cardCVV;
    }

    public PaymentEntity setId(int id) {
        this.id = id;
        return this;
    }

    public PaymentEntity setAmount(double amount) {
        this.amount = amount;
        return this;
    }

    public PaymentEntity setPaymentDate(Date paymentDate) {
        this.paymentDate = paymentDate;
        return this;
    }

    public PaymentEntity setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
        return this;
    }

    public PaymentEntity setCardExpiryMonth(String cardExpiryMonth)
{
        this.cardExpiryMonth = cardExpiryMonth;
        return this;
    }

    public PaymentEntity setCardExpiryYear(String cardExpiryYear) {
        this.cardExpiryYear = cardExpiryYear;
        return this;
    }

    public PaymentEntity setCardCVV(String cardCVV) {
        this.cardCVV = cardCVV;
        return this;
    }
}
```

Price:

```
package com.MyMoviePlan.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "prices")
public class PriceEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private double general;

    private double silver;
    private double gold;

    public PriceEntity(double general, double silver, double gold) {
        this.general = general;
        this.silver = silver;
        this.gold = gold;
    }
}
```

Show:

```
package com.MyMoviePlan.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.*;

import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "shows")
public class ShowEntity implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;

    @Column(name = "start_time")
    private String startTime;

    @JsonIgnore
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @ManyToOne(targetEntity = AuditoriumEntity.class)
    private AuditoriumEntity auditorium;

    //    @JsonManagedReference
    @ToString.Exclude
    @EqualsAndHashCode.Exclude
    @OneToMany(targetEntity = MovieShowsEntity.class, cascade =
CascadeType.ALL)
    @JoinColumn(name = "show_id", referencedColumnName = "id")
    private List<MovieShowsEntity> movieShows;

    public ShowEntity(String name, String startTime,
List<MovieShowsEntity> movieShows) {
        this.name = name;
        this.startTime = startTime;
        this.movieShows = movieShows;
    }

    public ShowEntity setId(int id) {
        this.id = id;
        return this;
    }

    public ShowEntity setName(String name) {
        this.name = name;
        return this;
    }

    public ShowEntity setStartTime(String startTime) {
        this.startTime = startTime;
        return this;
    }

    public ShowEntity setAuditorium(AuditoriumEntity auditorium) {
        this.auditorium = auditorium;
        return this;
    }

    public ShowEntity setMovieShows(List<MovieShowsEntity>
movieShows) {
        this.movieShows = movieShows;
        return this;
    }
}
```

User:

```
package com.MyMoviePlan.entity;

import com.MyMoviePlan.model.UserRole;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.GenericGenerator;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
@Table(name = "users")
public class UserEntity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY, generator =
"uuid2")
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    private String id;

    @Column(length = 50)
    private String name;

    @Column(nullable = false, length = 50, unique = true)
    private String email;

    @Column(nullable = false, length = 10, unique = true)
    private String mobile;

    @Column(length = 60)
    private String gender;

    private String password;

    private Boolean terms;

    @Column(name = "is_account_non_expired")
    private Boolean isAccountNonExpired;

    @Column(name = "is_account_non_locked")
    private Boolean isAccountNonLocked;

    @Column(name = "is_credentials_non_expired")
    private Boolean isCredentialsNonExpired;

    @Column(name = "is_enabled")
    private Boolean isEnabled;

    @Column(name = "user_role", length = 20)
    @Enumerated(EnumType.STRING)
    private UserRole userRole;

    public UserEntity(String name, String email, String mobile,
String gender, String password, Boolean terms,
                      Boolean isAccountNonExpired, Boolean
isAccountNonLocked,
                      Boolean isCredentialsNonExpired, Boolean
isEnabled, UserRole userRole) {
        this.name = name;
        this.email = email;
        this.mobile = mobile;
        this.gender = gender;
        this.password = password;
        this.terms = terms;
        this.isAccountNonExpired = isAccountNonExpired;
        this.isAccountNonLocked = isAccountNonLocked;
        this.isCredentialsNonExpired = isCredentialsNonExpired;
        this.isEnabled = isEnabled;
        this.userRole = userRole;
    }

    public UserEntity setId(String id) {
        this.id = id;
        return this;
    }

    public UserEntity setName(String name) {
        this.name = name;
        return this;
    }

    public UserEntity setEmail(String email) {
        this.email = email;
        return this;
    }

    public UserEntity setMobile(String mobile) {
        this.mobile = mobile;
        return this;
    }

    public UserEntity setGender(String gender) {
        this.gender = gender;
        return this;
    }

    public UserEntity setPassword(String password) {
        this.password = password;
        return this;
    }

    public UserEntity setActive(Boolean active) {
        terms = active;
        return this;
    }

    public UserEntity setAccountNonExpired(Boolean
accountNonExpired) {
        isAccountNonExpired = accountNonExpired;
        return this;
    }

    public UserEntity setAccountNonLocked(Boolean accountNonLocked)
{
        isAccountNonLocked = accountNonLocked;
        return this;
    }

    public UserEntity setCredentialsNonExpired(Boolean
credentialsNonExpired) {
        isCredentialsNonExpired = credentialsNonExpired;
        return this;
    }

    public UserEntity setEnabled(Boolean enabled) {
        isEnabled = enabled;
        return this;
    }

    public UserEntity setUserRole(UserRole userRole) {
        this.userRole = userRole;
        return this;
    }

    public UserEntity setTerms(Boolean terms) {
        this.terms = terms;
        return this;
    }


}
```

EXCEPTION:

Auditorium NotFound:

```
package com.MyMoviePlan.exception;

public class AuditoriumNotFoundException extends RuntimeException {

    public AuditoriumNotFoundException(String message) {
        super(message);
    }
}
```

Booking Notfound:

```
package com.MyMoviePlan.exception;

public class BookingNotFoundException extends RuntimeException{

    public BookingNotFoundException(String message) {
        super(message);
    }
}
```

Global:

```
package com.MyMoviePlan.exception;

import com.MyMoviePlan.model.HttpResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
```

```
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntity
ExceptionHandler;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@ControllerAdvice
public class GlobalExceptionHandler extends
ResponseEntityExceptionHandler {

    @ExceptionHandler(Exception.class)
    public ResponseEntity<HttpResponse> handleException(final
Exception exception,
                                                        final
HttpServletRequest request,
                                                        final
HttpServletResponse response) {
        Integer statusCode = (Integer) request
                .getAttribute("javax.servlet.error.status_code");

        final int status = response.getStatus();

        final String exceptionMessage = exception.getMessage();
        if (statusCode == null || statusCode == 0) {
            statusCode = status;
            if
(HttpStatus.valueOf(status).getReasonPhrase().equals("OK"))
                statusCode = 403;
        }

        final HttpStatus httpStatus =
HttpStatus.valueOf(statusCode);

        final HttpResponse httpResponse =
                new HttpResponse(statusCode,
httpStatus.getReasonPhrase(), exceptionMessage);
        return new ResponseEntity<HttpResponse>(httpResponse,
httpStatus);
    }
}
```

Movie Notfound:

```
package com.MyMoviePlan.exception;

public class MovieNotFoundException extends RuntimeException {

    public MovieNotFoundException(String message) {
        super(message);
    }
}
```

Movie Show Notfound:

```
package com.MyMoviePlan.exception;

public class MovieShowNotFoundException extends RuntimeException {
    public MovieShowNotFoundException(String message) {
        super(message);
    }
}
```

Show Notfound:

```
package com.MyMoviePlan.exception;

public class ShowNotFoundException extends RuntimeException{

    public ShowNotFoundException(String message) {
        super(message);
    }
}
```

Un Authorized:

```
package com.MyMoviePlan.exception;

public class UnAuthorizedException extends RuntimeException{

    public UnAuthorizedException(String message) {
        super(message);
    }


}
```

User Notfound:

```
package com.MyMoviePlan.exception;

public class UserNotFoundException extends RuntimeException {

    public UserNotFoundException(String message) {
        super(message);
    }
}
```

FILTER:

JWT Filter:

```
package com.MyMoviePlan.filter;

import com.MyMoviePlan.model.HttpResponse;
import com.MyMoviePlan.security.ApplicationUserDetailsService;
import com.MyMoviePlan.util.JWTUtil;
import io.jsonwebtoken.JwtException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import
org.springframework.security.authentication.UsernamePasswordAuthenti
cationToken;
import
org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.WebAuthenticationDet
ailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component()
public class JWTFilter extends OncePerRequestFilter {

    @Autowired
    private JWTUtil jwtUtil;

    @Autowired
    private ApplicationUserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws
ServletException, IOException {

        try {
            String authorization =
request.getHeader("Authorization");
            String token = null;
            String userName = null;

            if (authorization != null &&
authorization.startsWith("Bearer ")) {
                token = authorization.substring(7);
                userName = jwtUtil.getUsernameFromToken(token);
            }

            if (userName != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
                UserDetails userDetails
                        =
userDetailsService.loadUserByUsername(userName);

                if (jwtUtil.validateToken(token, userDetails)) {
                    UsernamePasswordAuthenticationToken
authenticationToken
                            = new
UsernamePasswordAuthenticationToken(userDetails,
                            null, userDetails.getAuthorities());

                    authenticationToken.setDetails(
                            new
WebAuthenticationDetailsSource().buildDetails(request)
                    );

SecurityContextHolder.getContext().setAuthentication(authenticationT
oken);
                }
            }
            filterChain.doFilter(request, response);
        } catch (JwtException exception) {
            setErrorResponse(HttpStatus.NOT_ACCEPTABLE, response,
exception);
        } catch (Exception exception) {
            setErrorResponse(HttpStatus.INTERNAL_SERVER_ERROR,
response, exception);
        }
    }

    private void setErrorResponse(HttpStatus status,
HttpServletResponse response, Exception exception) {
        response.setStatus(status.value());
        response.setContentType("application/json");
        final HttpResponse httpResponse =
                new HttpResponse(status.value(),

HttpStatus.valueOf(status.value()).getReasonPhrase(),
                        exception.getMessage());
        try {
            final String json = httpResponse.covertToJson();
            response.getWriter().write(json);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

MODEL:

Booked Seats:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class BookedSeats {

    private int count;
    private List<String> seats;
}
```

Credentials:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Credentials {

    private String username;
    private String password;
}
```

HTTP Response:

```
package com.MyMoviePlan.model;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class HttpResponse {

    private int statusCode; // 200, 201, 404

    private String error;  // OK, CREATED

    private String message;

    public String covertToJson() throws JsonProcessingException {
        if (this == null)
            return null;

        ObjectMapper mapper = new ObjectMapper();
        mapper.registerModule(new JavaTimeModule());

mapper.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS);

        return mapper.writeValueAsString(this);
    }
}
```

Ticket Details:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@AllArgsConstructor
@NoArgsConstructor
public class TicketDetails {

    private String auditoriumName;

    private String showName;

    private String showTiming;

    private double amount;

    private String movieName;

    private String movieImage;

    private String movieBgImage;
}
```

Token:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Token {

    private String token;
}
```

UserPermisions:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public enum UserPermission {
    READ,
    WRITE,
    UPDATE,
    DELETE
}
```

User Role:

```
package com.MyMoviePlan.model;

import lombok.AllArgsConstructor;
import lombok.Getter;

import java.util.Arrays;
import java.util.List;

import static com.MyMoviePlan.model.UserPermission.*;

@Getter
@AllArgsConstructor
public enum UserRole {

    ROLE_USER(Arrays.asList(READ)),
    ROLE_MANAGER(Arrays.asList(READ, WRITE)),
    ROLE_ADMIN(Arrays.asList(READ, WRITE, UPDATE)),
    ROLE_SUPER_ADMIN(Arrays.asList(READ, WRITE, UPDATE, DELETE));

    private final List<UserPermission> permissions;
}
```

REPOSITORY:

Actor:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.ActorEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ActorRepository extends JpaRepository<ActorEntity,
Integer> {
}
```

Auditorium:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.AuditoriumEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AuditoriumRepository extends
JpaRepository<AuditoriumEntity, Integer> {
}
```

Booking Repository:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.BookingEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface BookingRepository extends
JpaRepository<BookingEntity, Integer> {
    List<BookingEntity> findAllByUserIdOrderByBookedOnAsc(final
String userId);
}
```

Movie:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.MovieEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface MovieRepository extends JpaRepository<MovieEntity,
Integer> {
}
```

MovieShow:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.MovieShowsEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface MovieShowsRepository extends
JpaRepository<MovieShowsEntity, Integer> {

    //    https://docs.spring.io/spring-
data/commons/docs/current/reference/html/#repositories.limit-query-
result
    //    https://stackoverflow.com/questions/11401229/how-to-use-
select-distinct-with-random-function-in-postgresql
    //    https://stackoverflow.com/questions/32079084/how-to-find-
distinct-rows-with-field-in-list-using-jpa-and-spring
    //    https://dev.to/golovpavel/make-a-request-with-sub-
condition-for-child-list-via-spring-data-jpa-4inn
    //    https://docs.spring.io/spring-
data/jpa/docs/current/reference/html/#jpa.query-methods.query-
creation

    //    @Query(value = "SELECT DISTINCT ON(movie_id) * FROM
movie_shows WHERE start >= CURRENT_DATE", nativeQuery = true)
```

```
    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_start > CURRENT_DATE", nativeQuery =
true)
    List<MovieShowsEntity> findAllUpComing();

    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_start <= CURRENT_DATE AND ms.show_end
>= CURRENT_DATE", nativeQuery = true)
    List<MovieShowsEntity> findAllNowPlaying();

    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_end >= CURRENT_DATE", nativeQuery =
true)
    List<MovieShowsEntity> findAllNowPlayingAndUpComing();

    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_end < CURRENT_DATE", nativeQuery =
true)
    List<MovieShowsEntity> findAllNotPlaying();

    //    @Query("FROM MovieShowsEntity ms LEFT JOIN ms.bookings b
WHERE ms.id = ?1 AND b.dateOfBooking = ?2")
    //    @Query(value = "SELECT * FROM movie_shows ms INNER JOIN
bookings b ON ms.id = :id and b.date_of_booking = ':dateOfBooking'",
nativeQuery = true)
    //    Optional<MovieShowsEntity>
findByIdAndDateOfBooking(@Param("id") final int id,
@Param("dateOfBooking") final String dateOfBooking);

    //      SELECT * FROM (SELECT DISTINCT movie_id FROM movie_shows
WHERE start > CURRENT_DATE) ms ORDER BY random() LIMIT :records
    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_start > CURRENT_DATE LIMIT :records",
nativeQuery = true)
    List<MovieShowsEntity> findFewUpComing(@Param("records") final
int records);

    @Query(value = "SELECT DISTINCT ON(ms.movie_id) * FROM
movie_shows ms WHERE ms.show_start <= CURRENT_DATE AND ms.show_end
>= CURRENT_DATE LIMIT :records", nativeQuery = true)
    List<MovieShowsEntity> findFewNowPlaying(@Param("records") final
int records);

    //    @Query(value = "SELECT ms.id, ms.show_end, ms.movie_id,
ms.show_start, ms.show_id, ms.price_id FROM movie_shows ms INNER
JOIN bookings b ON b.id = :bookingId", nativeQuery = true)
    //    MovieShowsEntity findByBookingId(@Param("bookingId") final
int bookingId);
}
```

Payment:

```
package com.MyMoviePlan.repository;
```

```
import com.MyMoviePlan.entity.PaymentEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PaymentRepository extends
JpaRepository<PaymentEntity, Integer> {
}
```

Price:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.PriceEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PriceRepository extends JpaRepository<PriceEntity,
Integer> {
}
```

Show:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.ShowEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface ShowRepository extends JpaRepository<ShowEntity,
Integer> {
}
```

User:

```
package com.MyMoviePlan.repository;

import com.MyMoviePlan.entity.UserEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
```

```
public interface UserRepository extends JpaRepository<UserEntity,
String> {

    Optional<UserEntity> findByEmail(final String email);

    Optional<UserEntity> findByMobile(final String mobile);
}
```

SECURITY:

Application Security:

```
package com.MyMoviePlan.security;

import com.MyMoviePlan.filter.JWTFilter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.dao.DaoAuthenticationPro
vider;
import
org.springframework.security.config.annotation.authentication.builde
rs.AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.
EnableGlobalMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecu
rity;
import
org.springframework.security.config.annotation.web.builders.WebSecur
ity;
import
org.springframework.security.config.annotation.web.configuration.Ena
bleWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.Web
SecurityConfigurerAdapter;
import
org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.password.PasswordEncoder;
import
org.springframework.security.web.authentication.UsernamePasswordAuth
enticationFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import java.util.Arrays;
```

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class ApplicationSecurity extends
WebSecurityConfigurerAdapter {

    @Autowired
    private ApplicationUserDetailsService userDetailsService;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private JWTFilter jwtFilter;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.headers().frameOptions().sameOrigin();
        http.csrf().disable().cors().disable()

.cors().configurationSource(corsConfigurationSource())
                .and()
        .authorizeRequests()
                .antMatchers(HttpMethod.OPTIONS, "/**")
                .permitAll()
                .anyRequest()
                .fullyAuthenticated()
                .and()
                .httpBasic()
                .and()
                .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.addFilterBefore(jwtFilter,
UsernamePasswordAuthenticationFilter.class);
    }

//                .antMatchers(HttpMethod.POST,
"/user/authenticate", "/user/sign-up")
//                .permitAll()
//                .antMatchers(HttpMethod.PUT, "/user/forgot-
password")
//                .permitAll()
//                .antMatchers(HttpMethod.GET, "/auditorium/**",
"/movie/**", "/show/**", "/user/check/**")
//                .permitAll()

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring()
                .antMatchers("/h2-console/**", "/auditorium/**",
"/movie/**", "/show/**", "/user/**",
                        "/user/forgot-password",
"/user/authenticate", "/movie-show/**",
                        "/booking/**", "/logout");
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth)
throws Exception {
        auth.authenticationProvider(authenticationProvider());
    }

    @Bean
    public DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authenticationProvider = new
DaoAuthenticationProvider();
        authenticationProvider.setPasswordEncoder(passwordEncoder);

authenticationProvider.setUserDetailsService(userDetailsService);
        return authenticationProvider;
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(Arrays.asList("*"));
        configuration.setAllowedMethods(Arrays.asList("GET", "POST",
"PUT", "PATCH", "DELETE", "OPTIONS"));
        configuration.setAllowCredentials(true);
        //the below three lines will add the relevant CORS response
headers
        configuration.addAllowedOrigin("*");
        configuration.addAllowedHeader("*");
        configuration.addAllowedMethod("*");
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }


    @Override
    @Bean
    protected AuthenticationManager authenticationManager() throws
Exception {
        return super.authenticationManager();
    }
}
```

Application User Details:

```
package com.MyMoviePlan.security;

import com.MyMoviePlan.entity.UserEntity;
import lombok.AllArgsConstructor;
import org.springframework.security.core.GrantedAuthority;
```

```
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;

@AllArgsConstructor
public class ApplicationUserDetails implements UserDetails {

    private final UserEntity user;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        final List<SimpleGrantedAuthority> authorities =
user.getUserRole()
                .getPermissions()
                .stream()
                .map(permission -> new
SimpleGrantedAuthority(permission.name()))
                .collect(Collectors.toList());
        authorities.add(new
SimpleGrantedAuthority(user.getUserRole().name()));
        return authorities;
    }

    @Override
    public String getPassword() {
        return user.getPassword();
    }

    @Override
    public String getUsername() {
        return user.getEmail();
    }

    @Override
    public boolean isAccountNonExpired() {
        return user.getIsAccountNonExpired();
    }

    @Override
    public boolean isAccountNonLocked() {
        return user.getIsAccountNonLocked();
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return user.getIsCredentialsNonExpired();
    }

    @Override
    public boolean isEnabled() {
        return user.getIsEnabled();
    }
}
```

Application User Srvise:

```
package com.MyMoviePlan.security;

import com.MyMoviePlan.entity.UserEntity;
import com.MyMoviePlan.exception.UserNotFoundException;
import com.MyMoviePlan.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundExcept
ion;
import org.springframework.stereotype.Service;

@Service
public class ApplicationUserDetailsService implements
UserDetailsService {

    @Autowired
    private UserService service;

    @Override
    public UserDetails loadUserByUsername(final String username)
throws UsernameNotFoundException {
        final UserEntity userEntity = service.getUser(username);
        return new ApplicationUserDetails(userEntity);
    }
}
```

SERVLETS UTILIZER:

```
package com.MyMoviePlan;

import org.springframework.boot.builder.SpringApplicationBuilder;
import
org.springframework.boot.web.servlet.support.SpringBootServletInitia
lizer;

public class ServletInitializer extends SpringBootServletInitializer
{

     @Override
     protected SpringApplicationBuilder
configure(SpringApplicationBuilder application) {
          return application.sources(MyMoviePlanApplication.class);
     }
}
```

TEST:

My Movie Application Test:

```
package com.MyMoviePlan;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class MyMoviePlanApplicationTests {

	@Test
	void contextLoads() {
	}

}
```