**MY MOVIE PLAN CAPSTONE PROJECT**

**FRONT END:**

**About us:**

<divclass="about-us-container">

  <divclass="about-us-wrapper postion-relative">

    <divclass="about-us-banner">

      <divclass="about-us-banner-text h-100">

        <h1class="fs-1">

          {{ title }}

        </h1>

        <h2class="fs-2 text-uppercase">WE provide entertainment</h2>

      </div>

    </div>

  </div>

  <divclass="description mt-4">

    <divclass="mx-4">

      <h1class="fs-2">About Us</h1>

      <divclass="underline text-warning"></div>

    </div>

    <pclass="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>

  <divclass="testimonials p-lg-5 bg-dark text-white p-md-3 mt-4">

    <h1class="fs-2 pb-5 text-center">Our Happy Customer</h1>

    <divclass="testimonials-list row g-0">

      <divclass="testimonial-card col-4 px-4 my-4">

        <divclass="testimonial text-center">

          <divclass="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>

          <divclass="comments">

            <h3class="fs-3 user">User</h3>

            <divclass="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>

      <divclass="testimonial-card col-4 px-4 my-4">

        <divclass="testimonial text-center">

          <divclass="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>

          <divclass="comments">

            <h3class="fs-3 user">User</h3>

            <divclass="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>

      <divclass="testimonial-card col-4 px-4 my-4">

        <divclass="testimonial text-center">

          <divclass="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>

          <divclass="comments">

            <h3class="fs-3 user">User</h3>

            <divclass="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>

  <divclass="branches mt-4 py-5">

    <divclass="mx-4">

      <h1class="fs-2">Our Branches</h1>

      <divclass="underline"></div>

    </div>

    <divclass="row g-0 justify-content-around align-content-center">

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 1</mat-card

      >

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 2</mat-card

      >

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 3</mat-card

      >

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 1</mat-card

      >

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 2</mat-card

      >

      <mat-cardclass="col-3 mx-3 my-3 bg-warning"color="accent"

        >Branch 3</mat-card

      >

    </div>

  </div>

</div>

**Auditorium Forms:**

<divclass="form-container">

  <divclass="mx-auto text-centerbg-color">

    <divclass="container py-5">

      <h1class="fs-2 text-dark py-5 text-uppercase">

        Register New Auditorium Here

      </h1>

    </div>

  </div>

  <divclass="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-templatematStepLabel>Auditorium Details</ng-template>

          <divclass="row g-0 my-3 justify-content-around">

            <divclass="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>

            <divclass="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>

            <divclass="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>

                <inputmatInputformControlName="seatCapacity"readonly/>

              </mat-form-field>

            </div>

            <divclass="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>

            <divclass="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>

            <divclass="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>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="auditoriumForm.get('safeties')!"optional>

          <ng-templatematStepLabel>Add Safeties</ng-template>

          <divclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="auditoriumForm.get('facilities')!"optional>

          <ng-templatematStepLabel>Add Facilities</ng-template>

          <divclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="auditoriumForm.get('shows')!"optional>

          <ng-templatematStepLabel>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"

            >

              <divclass="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>

              <divclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step>

          <ng-templatematStepLabel>Review</ng-template>

          <pre>

            {{ auditoriumForm.value | json }}

          </pre>

          <div>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <button

              mat-button

              color="primary"

              [disabled]="!auditoriumForm.valid"

            >

              Submit

            </button>

          </div>

        </mat-step>

      </mat-vertical-stepper>

    </form>

  </div>

</div>

**Contact Us:**

<divclass="contact-us-container">

  <divclass="contact-us-wrapper postion-relative">

    <divclass="contact-us-banner">

      <divclass="contact-us-banner-text h-100">

        <h1class="fs-1">Contact Us</h1>

        <h2class="fs-2 text-uppercase">Drop us a line, get in touch</h2>

      </div>

    </div>

  </div>

  <divclass="get-in-touch my-5">

    <divclass="row g-0 justify-content-around">

      <divclass="write-to-us col-sm-12 col-md-12 col-lg-7">

        <h1class="fs-2">Write to us by filling in the form below</h1>

        <divclass="underline"></div>

        <divclass="form-container">

          <formclass="example-form col-sm-12 col-lg-8 mx-auto">

            <mat-form-field

              class="example-full-width d-block text-dark"

              appearance="standard"

            >

              <mat-labelclass="text-dark">Your Name</mat-label>

              <inputtype="text"matInputclass="fs-5 text-dark"/>

            </mat-form-field>

            <mat-form-field

              class="example-full-width d-block w-100 text-dark"

              appearance="standard"

            >

              <mat-labelclass="text-dark">Email Id</mat-label>

              <inputtype="email"matInputclass="fs-5 text-dark"/>

            </mat-form-field>

            <mat-form-field

              class="example-full-width d-block w-100 text-dark"

              appearance="standard"

            >

              <mat-labelclass="text-dark">Message</mat-label>

              <textareamatInputplaceholder="How can we help you!"></textarea>

            </mat-form-field>

            <buttonmat-raised-buttoncolor="accent"type="submit">

              Contact Us

            </button>

          </form>

        </div>

      </div>

      <divclass="address col-sm-12 col-md-12 col-lg-4">

        <h1class="fs-2">Contact Us</h1>

        <divclass="underline text-success"></div>

        <divclass="contact-address">

          <mat-list>

            <mat-list-itemrole="listitem"class="py-2 h-auto">

              <mat-iconclass="d-inline-block"> place </mat-icon>

              <divclass="address ms-3 text-muted col-10">

                Big Tree Entertainment Private Limited Ground Floor, Wajeda

                House, Gulmohar Cross Road No. 7, Juhu Scheme, Mumbai,

                Maharashtra - 400049

              </div>

            </mat-list-item>

            <mat-divider></mat-divider>

            <mat-list-itemrole="listitem"class="py-2 h-auto">

              <mat-iconclass="d-inline-block"> email </mat-icon>

              <divclass="address ms-3 text-muted col-10">nms@cinemas.com</div>

            </mat-list-item>

            <mat-divider></mat-divider>

            <mat-list-itemrole="listitem"class="py-2 h-auto">

              <mat-iconclass="d-inline-block col-1">support\_agent</mat-icon>

              <divclass="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

      href="https://jriyaz.github.io/Portfolio/"

      target="\_blank"

      class="text-decoration-none mx-2"

      ><strong>Hari G</strong></a

    >

  </div>

</footer>

**Forgot Password:**

<mainclass="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>

  <h1class="fs-2 py-4">Change Password Here</h1>

  <divclass="example-container">

    <form

      #passwordForm

      class="row w-100 g-0"

      [formGroup]="forgotPasswordForm"

      (ngSubmit)="onSubmit()"

    >

      <divclass="col-12 mb-3">

        <mat-form-fieldappearance="outline"class="text-white d-block">

          <mat-label>Enter your username</mat-label>

          <input

            matInput

            type="text"

            placeholder="nms@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>

      <divclass="col-12 mb-3">

        <mat-form-fieldappearance="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:**

<nav

  \*ngIf="router.url !== '/select-seats'"

  id="nav-bar"

  class="navbar navbar-expand-lg navbar-dark bg-show text-white px-lg-3"

>

  <divclass="container-fluid">

    <amat-buttonclass="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>

    <divclass="collapse navbar-collapse"id="navbarSupportedContent">

      <ulclass="navbar-nav me-auto mb-2 mb-lg-0 text-sm-center">

        <liclass="nav-item mx-lg-2 mx-md-auto">

          <a

            mat-button

            class="nav-link"

            aria-current="page"

            routerLinkActive="active"

            routerLink="/home"

            >Home</a

          >

        </li>

        <liclass="nav-item mx-lg-2 mx-md-auto">

          <a

            mat-button

            class="nav-link"

            routerLinkActive="active"

            [routerLink]="['/movies']"

            >Movies</a

          >

        </li>

        <liclass="nav-item mx-lg-2 mx-md-auto">

          <a

            mat-button

            class="nav-link"

            routerLinkActive="active"

            routerLink="/about"

            >About Us</a

          >

        </li>

        <liclass="nav-item mx-lg-2 mx-md-auto">

          <a

            mat-button

            class="nav-link"

            routerLinkActive="active"

            routerLink="/contact"

            >Contact Us</a

          >

        </li>

        <liclass="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>

          <ulclass="dropdown-menu"aria-labelledby="admin-panel">

            <liclass="nav-item mx-lg-2 mx-md-auto">

              <a

                class="dropdown-item"

                href="#"

                routerLinkActive="active"

                routerLink="/admin/add-auditorium"

                >Add Auditorium</a

              >

            </li>

            <liclass="nav-item mx-lg-2 mx-md-auto">

              <a

                class="dropdown-item"

                href="#"

                routerLinkActive="active"

                routerLink="/admin/add-movie"

                >Add Movie</a

              >

            </li>

            <liclass="nav-item mx-lg-2 mx-md-auto">

              <a

                class="dropdown-item"

                href="#"

                routerLinkActive="active"

                routerLink="/admin/manage"

                >Manage</a

              >

            </li>

          </ul>

        </li>

      </ul>

      <divclass="profile text-sm-center">

        <!--<button mat-button [matMenuTriggerFor]="menu">

          <img

            class="rounded-circle mx-2"

            src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSK9DmKOF9VCBqRzdHLzwyyOXjDm4kOYjD9IZQo2WviiLzzxDmzwYQZaVK6\_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> -->

        <ulclass="navbar-nav col-auto"\*ngIf="service.isLoggedIn()">

          <liclass="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"

            >

              <liclass="nav-item">

                <aclass="dropdown-item py-1"routerLink="/my/profile"

                  >Profile</a

                >

              </li>

              <liclass="nav-item">

                <aclass="dropdown-item py-1"routerLink="/my/bookings"

                  >Bookings</a

                >

              </li>

              <liclass="nav-item text-center">

                <a

                  class="btnbtn-outline-warning btn-sm col-11 mt-2"

                  (click)="onLogout()"

                  >Logout</a

                >

              </li>

            </ul>

          </li>

        </ul>

        <divclass="btn-group p-2"\*ngIf="!service.isLoggedIn()">

          <a

            routerLink="/user/login"

            routerLinkActive="active"

            class="btnbtn-primary btn-sm rounded mx-1"

            >Login</a

          >

          <a

            routerLink="/user/register"

            routerLinkActive="active"

            class="btnbtn-success btn-sm rounded mx-1"

            >Register</a

          >

        </div>

      </div>

    </div>

  </div>

</nav>

**Home Page:**

<divclass="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">

    <divid="movies-carousel"class="carousel slide"data-bs-ride="carousel">

      <divclass="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>

      <divclass="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"

          />

          <divclass="carousel-caption d-none d-md-block">

            <h5class="fs-1 mb-3">{{ movie.name | uppercase }}</h5>

            <pclass="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"

      >

        <spanclass="carousel-control-prev-icon"aria-hidden="true"></span>

        <spanclass="visually-hidden">Previous</span>

      </button>

      <button

        class="carousel-control-next"

        type="button"

        data-bs-target="#movies-carousel"

        data-bs-slide="next"

      >

        <spanclass="carousel-control-next-icon"aria-hidden="true"></span>

        <spanclass="visually-hidden">Next</span>

      </button>

    </div>

    <section

      \*ngIf="(nowPlaying$ | async)?.length! > 0"

      class="now-showing my-5 bg-dark text-white p-lg-5"

    >

      <divclass="row g-0 mb-5 justify-content-around">

        <h3class="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>

      <divclass="now-showing-movies">

        <divclass="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"

          >

            <divclass="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>

            <divclass="card-body text-center">

              <h2class="card-title fs-4 mb-0">

                <a

                  class="stretched-link"

                  routerLink="../movie/{{ movie.id }}"

                  >{{ movie.name | uppercase }}</a

                >

              </h2>

              <h3class="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"

    >

      <divclass="row g-0 mb-5 justify-content-around">

        <h3class="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>

      <divclass="up-coming-movies">

        <divclass="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"

          >

            <divclass="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>

            <divclass="card-body text-center">

              <h2class="card-title fs-4 mb-0">

                <a

                  class="stretched-link"

                  routerLink="../movie/{{ movie.id }}"

                  >{{ movie.name | uppercase }}</a

                >

              </h2>

              <h3class="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 :**

<mainclass="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>

  <h1class="fs-2 py-4">Login Page</h1>

  <divclass="example-container">

    <form

      action="#"

      class="row w-100 g-0"

      [formGroup]="loginForm"

      (ngSubmit)="onSubmit()"

    >

      <divclass="col-12 mb-3">

        <mat-form-fieldappearance="outline"class="text-white d-block">

          <mat-label>Enter your username</mat-label>

          <inputmatInputtype="text"formControlName="username"class="fs-5"/>

          <mat-error\*ngIf="usernameErrors">{{ usernameErrors }}</mat-error>

        </mat-form-field>

      </div>

      <divclass="col-12 mb-3">

        <mat-form-fieldappearance="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>

**LogInLayOut:**

<divclass="d-flex h-100 text-center text-white bg-dark cover-page mb-5">

  <divclass="cover-container d-flex w-100 p-3 mx-auto flex-column">

    <headerclass="mb-5">

      <divclass="clearfix">

        <h3class="float-md-start mb-0 fs-4 text-md-center">{{ title }}</h3>

        <navclass="nav nav-masthead justify-content-center float-md-end">

          <aclass="nav-link"aria-current="page"routerLink="/home">Home</a>

          <aclass="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:**

<divclass="wrapper w-100 my-2">

  <divclass="container-fluid row g-0 h-90vh">

    <divclass="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>

      <divclass="overflow-auto h-70vh">

        <h1\*ngIf="!allAuditoriums"class="text-danger text-center">

          Cinema Hall list is empty

        </h1>

        <mat-listrole="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!)"

          >

            <divclass="w-100">

              {{ hall.name | uppercase }}

            </div>

            <spanclass="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>

    <divclass="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>

      <divclass="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!)"

          >

            <divclass="w-100 row g-0 justify-content-around">

              <divclass="col-5">{{ show.name | titlecase }}</div>

              |

              <divclass="col-5">

                {{ "Time: " + formatTime(show.startTime) }}

              </div>

            </div>

            <spanclass="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>

    <divclass="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>

      <divclass="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"

          >

            <divclass="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>

            <spanclass="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:**

<divclass="movie-section">

  <divclass="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 centercenter;

        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 }})"

    >

      <divclass="movie-poster row g-0">

        <divclass="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>

        <divclass="movie-details text-white mx-3 px-3 col-lg-5 col-sm-12">

          <divclass="movie-title mb-lg-5 fs-lg-1">

            <h1>

              {{

                ((selectedMovie$ | async)?.name | uppercase) +

                  " : " +

                  ((selectedMovie$ | async)?.caption | titlecase)

              }}

            </h1>

          </div>

          <divclass="movie-format my-3">

            <buttonclass="btnbtn-light btn-smdiabled mx-1">4D</button>

            <buttonclass="btnbtn-light btn-smdiabled mx-1">

              {{ (selectedMovie$ | async)?.language }}

            </button>

          </div>

          <divclass="movie-duration my-3">

            <pstyle="font: unset; font-size: 20px">

              {{

                ((selectedMovie$ | async)?.duration | titlecase) +

                  " . " +

                  (selectedMovie$ | async)?.genres?.join(", ")

              }}

            </p>

          </div>

          <divclass="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>

  <divclass="about-movie text-dark p-4">

    <h2>About the Movie</h2>

    <pclass="lead"style="font-family: sans-serif">

      {{ (selectedMovie$ | async)?.story }}

    </p>

  </div>

  <!--<nav class="px-4 py-3 w-100 shadow-smbg-white text-dark">

    <h1>{{(selectedMovie$ | async)?.name + ' : '+ (selectedMovie$ | async)?.caption}}</h1>

    <p class="lead">{{(selectedMovie$ | async)?.genres.join(', ')}}</p>

  </nav> -->

  <divclass="movie-cast ms-5 my-4"\*ngIf="(cast$ | async)?.length! > 0">

    <h1class="fs-lg-2">Cast</h1>

    <divclass="row g-0">

      <div

        class="cast text-center col-sm-12 col-md-6 col-lg-3"

        \*ngFor="let cast of cast$ | async"

      >

        <divclass="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>

        <divclass="cast-name">

          <h5class="fs-5">{{ cast.name }}</h5>

        </div>

      </div>

    </div>

  </div>

  <divclass="movie-crew ms-5 my-4 pb-5"\*ngIf="(crew$ | async)?.length! > 0">

    <h1class="fs-lg-2">Crew</h1>

    <divclass="row g-0">

      <div

        class="crew text-center col-sm-12 col-md-6 col-lg-3"

        \*ngFor="let crew of crew$ | async"

      >

        <divclass="image rounded-circle mx-auto mb-3">

          <img

            [alt]="crew.name"

            width="120px"

            height="120px"

            [src]="crew.image"

          />

        </div>

        <divclass="cast-name">

          <h5class="fs-5">{{ crew.name }}</h5>

        </div>

      </div>

    </div>

  </div>

</div>

**Movie Forms:**

<divclass="form-container">

  <divclass="mx-auto text-centerbg-color">

    <divclass="container py-5">

      <h1class="fs-2 text-dark py-5 text-uppercase">Add New Movie</h1>

    </div>

  </div>

  <divclass="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-templatematStepLabel>Movie Details</ng-template>

          <divclass="row g-0 my-3 justify-content-around">

            <divclass="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>

            <divclass="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>

            <divclass="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>

            <divclass="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>

            <divclass="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>

            <divclass="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>

                <inputmatInputformControlName="caption"/>

              </mat-form-field>

            </div>

            <divclass="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-templatematStepLabel>Add Language</ng-template>

          <divclass="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#languageListaria-label="language selection">

                <mat-chip

                  \*ngFor="let language of languages"

                  [selectable]="true"

                  [removable]="true"

                  (removed)="removeLanguage(language)"

                >

                  {{ language }}

                  <mat-iconmatChipRemove\*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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="movieForm.get('genres')!">

          <ng-templatematStepLabel>Add Language</ng-template>

          <divclass="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">

                  <spanclass="mx-1"\*ngIf="genres.errors?.required"

                    >Genre cannot be empty</span

                  >

                  <spanclass="mx-1"\*ngIf="genres.errors?.validGenre"

                    >Invalid movie genre</span

                  >

                  <spanclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="movieForm.get('casts')!">

          <ng-templatematStepLabel>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"

            >

              <divclass="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>

              <divclass="col-lg-3 col-sm-6">

                <mat-form-field

                  class="d-block w-100 text-dark"

                  appearance="outline"

                >

                  <mat-label>Actor Role</mat-label>

                  <inputmatInputplaceholder="Hero"formControlName="role"/>

                  <mat-error

                    \*ngIf="cast.get('role')?.errors?.required&&cast.touched"

                  >

                    Actor role cannot be empty

                  </mat-error>

                </mat-form-field>

              </div>

              <divclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step[stepControl]="movieForm.get('crews')!">

          <ng-templatematStepLabel>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"

            >

              <divclass="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>

              <divclass="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>

              <divclass="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>

          <divclass="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>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <buttonmat-buttonmatStepperNexttype="button">Next</button>

          </div>

        </mat-step>

        <mat-step>

          <ng-templatematStepLabel>Review</ng-template>

          <pre>

              {{ movieForm.value | json }}

            </pre

          >

          <div>

            <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

            <button

              mat-button

              class="text-success"

              [disabled]="!movieForm.valid"

            >

              Submit

            </button>

          </div>

        </mat-step>

      </mat-horizontal-stepper>

    </form>

  </div>

</div>

**Movies:**

<divclass="px-lg-5 row g-0 my-5 justify-content-around position-relative">

  <divclass="col-3">

    <divclass="sorting text-center mb-3 align-self-center">

      <h2class="fs-2 text-center text-dark text-uppercase">Show</h2>

      <br/>

      <mat-form-fieldappearance="outline"class="d-block w-100">

        <mat-label>Show</mat-label>

        <mat-selectname="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/>

    <divclass="filter">

      <h2class="fs-2 text-center text-dark text-uppercase">Filters</h2>

      <br/>

      <mat-accordion

        class="example-headers-align"

        multi

        togglePosition="before"

      >

        <mat-expansion-panelclass="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>

          <divclass="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="btnbtn-sm"

                [for]="language"

                [ngClass]="{

                  'btn-warning': language == languageFilter,

                  'btn-info': language != languageFilter

                }"

                >{{ language }}</label

              >

            </div>

          </div>

        </mat-expansion-panel>

        <mat-expansion-panelclass="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>

          <divclass="row g-0 justify-content-around">

            <divclass="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="btnbtn-sm"

                [for]="genre"

                [ngClass]="{

                  'btn-warning': genre == genreFilter,

                  'btn-info': genre != genreFilter

                }"

                >{{ genre }}</label

              >

            </div>

          </div>

        </mat-expansion-panel>

      </mat-accordion>

    </div>

  </div>

  <divclass="col-8 movies">

    <divclass="now-showing">

      <mat-toolbar

        [color]="showText == 'Now Playing' ? 'primary' : 'accent'"

        class="rounded"

      >

        <divclass="row g-0 justify-content-around w-100 h-auto py-3">

          <divclass="col-sm-12 col-lg-3 text-center">{{ showText }}</div>

          <divclass="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>

      <h1class="text-center text-warning my-4"\*ngIf="!(moviesList$ | async)">

        No movies found

      </h1>

      <divclass="now-showing-movies my-4"\*ngIf="moviesList$ | async">

        <divclass="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

              <spanclass="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>

              <divclass="card-body text-center">

                <h2class="card-title fs-4 mb-0">

                  <a

                    class="stretched-link"

                    routerLink="../movie/{{ movie.id }}"

                    >{{ movie.name | uppercase }}</a

                  >

                </h2>

                <h3class="card-text text-muted">

                  {{ movie.language | titlecase }}

                </h3>

              </div>

            </div>

          </ng-container>

        </div>

      </div>

    </div>

  </div>

</div>

**My Bookings:**

<divclass="col-10 mx-auto my-4 h-80vh">

  <mat-toolbarcolor="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">

    <divclass="row g-0 col-12">

      <divclass="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="btnbtn-sm w-100 clearfix"

            [for]="'booking-' + i"

            [ngClass]="{

              'btn-dark text-white': booking.id == selectedBookingId,

              'btn-info': booking.id != selectedBookingId

            }"

          >

            <spanclass="d-inline-block float-start">{{

              "Booking " + (i + 1)

            }}</span>

            <spanclass="d-inline-block float-end">{{ booking.bookedOn }}</span>

          </label>

        </div>

      </div>

      <divclass="col-10 px-2">

        <divclass="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>

          <divclass="col-8 px-2 details h-100 my-auto">

            <tableclass="table ms-3">

              <thead>

                <tr>

                  <th

                    scope="col"

                    colspan="2"

                    class="text-center fs-4 text-capitalize"

                  >

                    {{ auditoriumName$ | async }}

                  </th>

                </tr>

              </thead>

              <tbody>

                <tr>

                  <thscope="row">Movie Language</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ movieLanguage$ | async }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Total seats</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ noOfTickets$ | async }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Seat No.</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ (selectedSeats$ | async)?.join(", ") }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Price</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ amount$ | async | currency: "INR" }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Date</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ dateOfBooking$ | async | date: "mediumDate" }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Time</th>

                  <td>

                    <h1class="m-0 d-inline-block">

                      {{ formatTime((showTiming$ | async)!) }}

                    </h1>

                  </td>

                </tr>

                <tr>

                  <thscope="row">Status</th>

                  <td>

                    <spanclass="text-muted text-success">Confirmed</span>

                  </td>

                </tr>

              </tbody>

            </table>

          </div>

        </div>

      </div>

    </div>

  </mat-card>

</div>

**Profile:**

<divclass="h-80vh">

  <divclass="col-10 mx-auto my-4">

    <mat-toolbarcolor="primary"class="rounded text-center"

      >Your profile</mat-toolbar

    >

  </div>

  <mat-cardclass="col-10 mx-auto">

    <tableclass="table">

      <tbody>

        <tr>

          <thscope="row">Name</th>

          <td>{{ (user | async)?.name }}</td>

        </tr>

        <tr>

          <thscope="row">Email</th>

          <td>{{ (user | async)?.email }}</td>

        </tr>

        <tr>

          <thscope="row">Mobile</th>

          <td>{{ (user | async)?.mobile }}</td>

        </tr>

        <tr>

          <thscope="row">Gender</th>

          <td>{{ (user | async)?.gender }}</td>

        </tr>

      </tbody>

    </table>

  </mat-card>

</div>

**Register:**

<mainclass="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>

  <h1class="fs-2 py-4">Register Here</h1>

  <divclass="example-container">

    <form

      class="row w-100 g-0"

      [formGroup]="registerForm"

      (ngSubmit)="onSubmit()"

    >

      <!-- Email Field -->

      <divclass="col-12 mb-2 px-lg-1">

        <mat-form-fieldappearance="outline"class="d-block text-white">

          <mat-label>Enter your Email</mat-label>

          <inputmatInputtype="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>

      <divclass="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">

        <mat-form-fieldappearance="outline"class="d-block text-white">

          <mat-label>Enter your name</mat-label>

          <inputmatInputtype="text"formControlName="name"class="fs-5"/>

          <mat-error\*ngIf="nameErrors">{{ nameErrors }}</mat-error>

        </mat-form-field>

      </div>

      <divclass="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">

        <mat-form-fieldappearance="outline"class="d-block text-white">

          <mat-label>Enter your Mobile</mat-label>

          <spanmatPrefix>+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>

      <divclass="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">

        <mat-form-fieldappearance="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>

      <divclass="col-sm-12 col-md-12 col-lg-6 mb-2 px-lg-1">

        <mat-form-fieldappearance="outline"class="d-block text-white">

          <mat-label>Select Gender</mat-label>

          <mat-selectformControlName="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>

      <divclass="col-lg-12 col-md-12 mb-2">

        <mat-checkboxclass="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:**

<divclass="wrapper">

  <h1class="text-center mb-3 p-2 rounded border border-2 shadow-sm">

    Add Movie To Show

  </h1>

  <formclass="example-form p-4"[formGroup]="movieShowForm">

    <mat-form-fieldclass="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"

          />

          <spanclass="me-1">{{ movie.name | uppercase }}</span> |

          <spanclass="me-1">{{ movie.language | titlecase }}</span> |

          <smallclass="me-1"

            >Release: {{ movie.release | date: "fullDate" }}</small

          >

          |

          <smallclass="me-1"

            >Added: {{ movie.addedOn | date: "fullDate" }}</small

          >

        </mat-option>

      </mat-select>

      <mat-error\*ngIf="movieIdErrors">{{ movieIdErrors }}</mat-error>

    </mat-form-field>

    <mat-form-fieldappearance="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"

      >

        <inputmatStartDateformControlName="start"placeholder="Start date"/>

        <inputmatEndDateformControlName="end"placeholder="End date"/>

      </mat-date-range-input>

      <mat-datepicker-togglematSuffix[for]="picker"></mat-datepicker-toggle>

      <mat-date-range-picker#picker></mat-date-range-picker>

      <mat-errorclass="d-inline-block mx-1"\*ngIf="startDateErrors">{{

        startDateErrors

      }}</mat-error>

      <mat-errorclass="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>

      <legendclass="lead">Price details</legend>

      <divclass="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>

          <inputtype="number"matInputformControlName="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>

          <inputtype="number"matInputformControlName="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>

          <inputtype="number"matInputformControlName="general"/>

          <mat-error\*ngIf="generalPriceErrors">{{

            generalPriceErrors

          }}</mat-error>

        </mat-form-field>

      </div>

    </fieldset>

  </form>

  <buttonmat-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:**

<divclass="wrapper p-5">

  <divclass="text-center mb-3 p-2 rounded border border-2 shadow-sm">

    <h1class="text-center d-inline-block my-2">Payment Gateway</h1>

    (<spanclass="text-danger d-inline-block my-2">do not refresh the page</span

    >)

  </div>

  <divclass="my-auto h-70vh row g-0 justify-content-around">

    <divclass="col-6 px-2 mx-auto order-2">

      <form[formGroup]="paymentForm"class="example-form p-4">

        <mat-form-fieldappearance="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>

        <divclass="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-selectformControlName="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-selectformControlName="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>

    <divclass="col-6 px-2 mx-auto order-1">

      <tableclass="table">

        <thead>

          <tr>

            <thscope="col"colspan="2">Booking Details</th>

          </tr>

        </thead>

        <tbody>

          <tr>

            <td>Auditorium</td>

            <thscope="row">{{ selectedMembers.auditoriumName }}</th>

          </tr>

          <tr>

            <td>Show</td>

            <thscope="row">

              {{ selectedMembers.showName }}

            </th>

          </tr>

          <tr>

            <td>Show Timing</td>

            <thscope="row">

              {{ formatTime(selectedMembers.showTime) }}

            </th>

          </tr>

          <tr>

            <td>Movie Name</td>

            <thscope="row">{{ selectedMembers.movieName | uppercase }}</th>

          </tr>

          <tr>

            <td>Movie Lang</td>

            <thscope="row">{{ selectedMembers.movieLanguage }}</th>

          </tr>

          <tr>

            <td>Date</td>

            <thscope="row">{{ selectedMembers.date | date: "mediumDate" }}</th>

          </tr>

        </tbody>

      </table>

    </div>

  </div>

</div>

**Screen:**

<divclass="screen-container w-100 py-3 bg-dark text-white">

  <spanclass="d-inline-block text-warning mx-auto text-center py-2 w-100"

    >NOTE: do not refresh the page</span

  >

  <divclass="my-auto row g-0 align-content-center">

    <divclass="ticket-details mb-3 row g-0 justify-content-between">

      <divclass="col-4 text-center">

        Movie: &nbsp;

        <spanclass="lead text-white">{{

          selectMembers.movieName | uppercase

        }}</span>

      </div>

      <divclass="col-4 text-center">

        Time: &nbsp;

        <spanclass="lead">{{ formatTime(selectMembers.showTime) }}</span>

      </div>

      <divclass="col-4 text-center">

        Date: &nbsp;

        <spanclass="lead">{{ selectMembers.date | date: "fullDate" }}</span>

      </div>

    </div>

    <divclass="ticket-details mb-3 row g-0 justify-content-between">

      <divclass="col-4 text-center">

        Seats Selected: &nbsp;

        <spanclass="lead text-danger">{{ getSelectedSeats() }}</span>

      </div>

      <divclass="col-4 text-center">

        Amount: &nbsp;

        <spanclass="lead">{{ totalAmount }}</span>

      </div>

      <divclass="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-cardclass="bg-dark text-white text-center">

        <mat-listclass="bg-dark text-white">

          <ng-container

            \*ngFor="let item of tempSeatColumns.reverse(); index as i"

          >

            <h3class="text-danger m-1"\*ngIf="i == 7">

              <!-- EXECUTIVE &#8377;200.00 -->

              EXECUTIVE {{ generalPrice$ | async | currency: "INR" }}

            </h3>

            <h3class="text-danger m-1"\*ngIf="i == 3">

              SILVER {{ silverPrice$ | async | currency: "INR" }}

            </h3>

            <h3class="text-danger m-1"\*ngIf="i == 0">

              GOLD {{ goldPrice$ | async | currency: "INR" }}

            </h3>

            <mat-list-itemclass="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

                "

              >

                <divclass="col-1 text-center text-info fs-3">

                  {{ item }}

                </div>

                <divclass="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-templatematStepLabel>Select the Cinema Hall</ng-template>

        <mat-form-fieldclass="d-block w-100 mt-2">

          <mat-label>Cinema Hall Name</mat-label>

          <mat-selectclass="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>

          <buttonmat-buttonmatStepperNexttype="button">Next</button>

        </div>

      </mat-step>

      <mat-step

        [stepControl]="ticketsForm.get('showName')!"

        label="Select the Show Timing"

      >

        <mat-form-fieldclass="d-block w-100 mt-2">

          <mat-labelclass="text-dark">Show Timing</mat-label>

          <mat-selectclass="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>

          <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

          <buttonmat-buttonmatStepperNexttype="button">Next</button>

        </div>

      </mat-step>

      <mat-step

        [stepControl]="ticketsForm.get('date')!"

        label="Select the Date"

      >

        <mat-form-fieldclass="d-block w-100 mt-2">

          <mat-labelclass="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>

          <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

          <buttonmat-buttonmatStepperNexttype="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>

          <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

          <button

            mat-button

            matStepperNext

            [disabled]="selectedSeats! < 1"

            type="button"

          >

            next

          </button>

        </div>

      </mat-step>

      <mat-step>

        <ng-templatematStepLabel>Review</ng-template>

        <tableclass="table table-striped">

          <tr>

            <thscope="col">Auditorium</th>

            <td>{{ ticketsForm.get("auditoriumName")?.value! }}</td>

          </tr>

          <tr>

            <thscope="col">Show</th>

            <td>

              {{

                ticketsForm.get("showName")?.value! +

                  " | Time: " +

                  formatTime(selectMembers?.showTime!)

              }}

            </td>

          </tr>

          <tr>

            <thscope="col">Date</th>

            <td>{{ ticketsForm.get("date")?.value! | date: "fullDate" }}</td>

          </tr>

          <tr>

            <thscope="col">Seats</th>

            <td>{{ ticketsForm.get("seats")?.value! }}</td>

          </tr>

        </table>

        <div>

          <buttonmat-buttonmatStepperPrevioustype="button">Back</button>

          <buttonmat-buttoncolor="primary"[disabled]="!ticketsForm.valid">

            Proceed

          </button>

        </div>

      </mat-step>

    </mat-horizontal-stepper>

  </form>

</mat-card>

**Show Form:**

<divclass="wrapper">

  <h1class="text-center mb-3 p-2 rounded border border-2 shadow-sm">

    Add New Show

  </h1>

  <formclass="example-form p-4"[formGroup]="showForm">

    <mat-form-fieldclass="d-block w-100 text-dark"appearance="outline">

      <mat-label>Show Name</mat-label>

      <inputmatInputformControlName="name"list="show-name"/>

      <datalistid="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-fieldclass="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>

  <buttonmat-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**

<!DOCTYPEhtml>

<htmllang="en">

  <head>

    <metacharset="utf-8"/>

    <title>My Movie Plan</title>

    <basehref="/"/>

    <metaname="viewport"content="width=device-width, initial-scale=1"/>

    <linkrel="icon"type="image/x-icon"href="favicon.ico"/>

    <!-- Bootstrap Css -->

    <link

      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.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.min.css"

      crossorigin="anonymous"

    /> -->

    <linkrel="preconnect"href="https://fonts.gstatic.com"/>

    <link

      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap"

      rel="stylesheet"

    />

    <link

      href="https://fonts.googleapis.com/icon?family=Material+Icons"

      rel="stylesheet"

    />

    <linkrel="preconnect"href="https://fonts.gstatic.com"/>

    <link

      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap"

      rel="stylesheet"

    />

    <link

      href="https://fonts.googleapis.com/icon?family=Material+Icons"

      rel="stylesheet"

    />

    <linkrel="preconnect"href="https://fonts.gstatic.com"/>

    <link

      href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap"

      rel="stylesheet"

    />

    <link

      href="https://fonts.googleapis.com/icon?family=Material+Icons"

      rel="stylesheet"

    />

  </head>

  <bodyclass="mat-typography">

    <app-root></app-root>

    <script

      src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"

      integrity="sha384-gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4"

      crossorigin="anonymous"

    ></script>

    <scripttype="text/javascript">

      constmyCarousel = document.querySelector("#movies-carousel");

      if (myCarousel) {

        constcarousel = newbootstrap.Carousel(myCarousel, {

          interval:2000,

          wrap:false,

        });

      }

    </script>

  </body>

</html>

**DOCKER FILE:**

FROMnode:current-alpine3.11 as build-stage

# RUN mkdir -p /app

WORKDIR /usr/src/app

COPYpackage.json package-lock.json ./

RUNnpm install

COPY . .

RUNnpm run build --prod

# CMD ["npm", "start"]

FROMnginx:latest-alpineas prod-stage

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

COPYnginx.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 PasswordEncoderpasswordEncoder;

@Override

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

// final UserEntitysuper\_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 MovieShowsRepositorymovieShow;

private final AuditoriumRepository auditorium;

@GetMapping({"/", "all"})

public List<AuditoriumEntity>findAllAuditoriums() {

return this.auditorium.findAll();

}

@GetMapping("{auditorium\_id}")

@PreAuthorize("hasAuthority('WRITE')")

public AuditoriumEntityfindAuditoriumById(@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 AuditoriumEntitysaveAuditorium(@RequestBody final AuditoriumEntity auditorium) {

return this.auditorium.save(auditorium);

}

@PutMapping("update")

@PreAuthorize("hasAuthority('UPDATE')")

public AuditoriumEntityupdateAuditorium(@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 ShowEntityfindShowById(@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 ShowEntitysaveShow(@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 ShowEntityupdateShow(@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 MovieShowsEntityfindMovieShowById(@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 MovieShowsEntitysaveMovieShow(@PathVariable final int auditorium\_id,

@PathVariable final int show\_id,

@RequestBody final MovieShowsEntitymovieShow) {

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 MovieShowsEntityupdateMovieShow(@PathVariable final int auditorium\_id,

@PathVariable final int show\_id,

@RequestBody final MovieShowsEntitymovieShow) {

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 MovieShowsEntitymovieShow = 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 BookingEntityfindBookingById(@PathVariable final int auditorium\_id,

@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@PathVariable final int booking\_id) {

final MovieShowsEntitymovieShow = 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 BookingEntitysaveBooking(@PathVariable final int auditorium\_id,

@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@RequestBody final BookingEntity booking) {

final MovieShowsEntitymoveShow = this.findMovieShowById(auditorium\_id, show\_id, movie\_show\_id);

booking.setUserId(this.service.getLoggedInUser().getId());

// booking.setUserId(this.service.findByMobile("8099531318").get().getId());

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 BookingEntityupdateBooking(@PathVariable final int auditorium\_id,

@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@RequestBody final BookingEntity booking) {

final MovieShowsEntitymoveShow = 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 TicketDetailsgetMovieDetails(@PathVariable final int booking\_id) {

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

final MovieShowsEntitymovieShow = 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 ShowEntityshowEntity = 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 BookingEntityfindById(@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 BookingDetailsEntityfindByDetailsId(@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 MovieRepositorymovieRepository;

private final MovieShowsRepositorymovieShowsRepository;

@GetMapping({"/", "all"})

public List<MovieEntity>findAll() {

return movieRepository.findAll();

}

@GetMapping("{movie\_id}")

public MovieEntityfindById(@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 MovieEntitysaveMovie(@RequestBody final MovieEntity movie) {

return movieRepository.save(movie);

}

@PutMapping("update")

@PreAuthorize("hasAuthority('UPDATE')")

public MovieEntityupdateMovie(@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 MovieShowsEntitysave(@RequestBody MovieShowsEntitymovieShow) {

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 MovieShowsEntityfindMovieShowById(@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 BookedSeatsbookedSeats(@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 (BookingEntitybooking : 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 MovieShowsRepositorymovieShow;

private final MovieRepository movie;

private final UserService service;

private final BookingRepository booking;

@GetMapping("{show\_id}")

public ShowEntityfindShowById(@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 MovieShowsEntityfindMovieShowById(@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 MovieShowsEntitysaveMovieShow(@PathVariable final int show\_id,

@RequestBody final MovieShowsEntitymovieShow) {

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 MovieShowsEntityupdateMovieShow(@PathVariable final int show\_id,

@RequestBody final MovieShowsEntitymovieShow) {

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 MovieShowsEntitymovieShow = 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 BookingEntityfindBookingById(@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@PathVariable final int booking\_id) {

final MovieShowsEntitymovieShow = 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 BookingEntitysaveBooking(@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@RequestBody final BookingEntity booking) {

final MovieShowsEntitymoveShow = this.findMovieShowById(show\_id, movie\_show\_id);

// booking.setUserId(this.service.getLoggedInUser().getId());

booking.setUserId(this.service.findByMobile("8099531318").get().getId());

booking.setMovieShow(moveShow);

return this.booking.save(booking);

}

@PutMapping("{show\_id}/movie-show/{movie\_show\_id}/booking/update")

@PreAuthorize("hasAuthority('UPDATE')")

public BookingEntityupdateBooking(@PathVariable final int show\_id,

@PathVariable final int movie\_show\_id,

@RequestBody final BookingEntity booking) {

final MovieShowsEntitymoveShow = 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 UserEntityuser() {

return service.getLoggedInUser()

.setPassword(null);

}

@GetMapping("all")

@PreAuthorize("hasAuthority('WRITE')")

public List<UserEntity>allUsers() {

return service.findAll();

}

@PutMapping("update/{username}")

@PreAuthorize("hasAuthority('READ')")

public UserEntityupdateUser(@RequestBody final UserEntityuserEntity,

@PathVariable final String username) {

return service.update(userEntity, username);

}

@PostMapping("sign-up")

public HttpResponsesignUp(@RequestBody final UserEntityuserEntity) {

return service.register(userEntity);

}

@PutMapping("change-password")

@PreAuthorize("hasAuthority('READ')")

public HttpResponsechangePassword(@RequestBody final Credentials credentials) {

return service.changePassword(credentials);

}

@PutMapping("forgot-password")

public HttpResponseforgotPassword(@RequestBody final Credentials credentials) {

return service.forgotPassword(credentials);

}

@DeleteMapping("delete/{username}")

@PreAuthorize("hasAuthority('DELETE')")

public HttpResponsedelete(@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 AuditoriumEntitysetId(int id) {

this.id = id;

return this;

}

public AuditoriumEntitysetName(String name) {

this.name = name;

return this;

}

public AuditoriumEntitysetImage(String image) {

this.image = image;

return this;

}

public AuditoriumEntitysetEmail(String email) {

this.email = email;

return this;

}

public AuditoriumEntitysetCustomerCare(String customerCareNo) {

this.customerCareNo = customerCareNo;

return this;

}

public AuditoriumEntitysetAddress(String address) {

this.address = address;

return this;

}

public AuditoriumEntitysetSeatCapacity(int seatCapacity) {

this.seatCapacity = seatCapacity;

return this;

}

public AuditoriumEntitysetFacilities(List<String> facilities) {

this.facilities = facilities;

return this;

}

public AuditoriumEntitysetSafeties(List<String> safeties) {

this.safeties = safeties;

return this;

}

public AuditoriumEntitysetShows(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 BookingDetailsEntitybookingDetails;

@JsonIgnore

@ToString.Exclude

@EqualsAndHashCode.Exclude

@ManyToOne(targetEntity = MovieShowsEntity.class)

private MovieShowsEntitymovieShow;

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

PaymentEntity payment, String userId, MovieShowsEntitymovieShow) {

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 BookingEntitysetMovieShow(MovieShowsEntitymovieShow) {

this.movieShow = movieShow;

return this;

}

public BookingEntitysetId(int id) {

this.id = id;

return this;

}

public BookingEntitysetAmount(double amount) {

this.amount = amount;

return this;

}

public BookingEntitysetTotalSeats(int totalSeats) {

this.totalSeats = totalSeats;

return this;

}

public BookingEntitysetStatus(Date bookedOn) {

this.bookedOn = bookedOn;

return this;

}

public BookingEntitysetDateOfBooking(Date dateOfBooking) {

this.dateOfBooking = dateOfBooking;

return this;

}

public BookingEntitysetSeatNumbers(List<String>seatNumbers) {

this.seatNumbers = seatNumbers;

return this;

}

public BookingEntitysetPayment(PaymentEntity payment) {

this.payment = payment;

return this;

}

public BookingEntitysetUserId(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 MovieEntitysetId(int id) {

this.id = id;

return this;

}

public MovieEntitysetName(String name) {

this.name = name;

return this;

}

public MovieEntitysetImage(String image) {

this.image = image;

return this;

}

public MovieEntitysetBgImage(String bgImage) {

this.bgImage = bgImage;

return this;

}

public MovieEntitysetStory(String story) {

this.story = story;

return this;

}

public MovieEntitysetYear(String year) {

this.year = year;

return this;

}

public MovieEntitysetDuration(String duration) {

this.duration = duration;

return this;

}

public MovieEntitysetCaption(String caption) {

this.caption = caption;

return this;

}

public MovieEntitysetAddedOn(Date addedOn) {

this.addedOn = addedOn;

return this;

}

public MovieEntitysetRelease(Date release) {

this.release = release;

return this;

}

public MovieEntitysetLanguages(String language) {

this.language = language;

return this;

}

public MovieEntitysetGenres(List<String> genres) {

this.genres = genres;

return this;

}

public MovieEntitysetCasts(List<ActorEntity> casts) {

this.casts = casts;

return this;

}

public MovieEntitysetCrews(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 MovieShowsEntitysetId(int id) {

this.id = id;

return this;

}

public MovieShowsEntitysetStart(Date start) {

this.start = start;

return this;

}

public MovieShowsEntitysetEnd(Date end) {

this.end = end;

return this;

}

public MovieShowsEntitysetShow(ShowEntity show) {

this.show = show;

return this;

}

public MovieShowsEntitysetMovieId(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 PaymentEntitysetId(int id) {

this.id = id;

return this;

}

public PaymentEntitysetAmount(double amount) {

this.amount = amount;

return this;

}

public PaymentEntitysetPaymentDate(Date paymentDate) {

this.paymentDate = paymentDate;

return this;

}

public PaymentEntitysetCardNumber(String cardNumber) {

this.cardNumber = cardNumber;

return this;

}

public PaymentEntitysetCardExpiryMonth(String cardExpiryMonth) {

this.cardExpiryMonth = cardExpiryMonth;

return this;

}

public PaymentEntitysetCardExpiryYear(String cardExpiryYear) {

this.cardExpiryYear = cardExpiryYear;

return this;

}

public PaymentEntitysetCardCVV(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 ShowEntitysetId(int id) {

this.id = id;

return this;

}

public ShowEntitysetName(String name) {

this.name = name;

return this;

}

public ShowEntitysetStartTime(String startTime) {

this.startTime = startTime;

return this;

}

public ShowEntitysetAuditorium(AuditoriumEntity auditorium) {

this.auditorium = auditorium;

return this;

}

public ShowEntitysetMovieShows(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 UserRoleuserRole;

public UserEntity(String name, String email, String mobile, String gender, String password, Boolean terms,

Boolean isAccountNonExpired, Boolean isAccountNonLocked,

Boolean isCredentialsNonExpired, Boolean isEnabled, UserRoleuserRole) {

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 UserEntitysetId(String id) {

this.id = id;

return this;

}

public UserEntitysetName(String name) {

this.name = name;

return this;

}

public UserEntitysetEmail(String email) {

this.email = email;

return this;

}

public UserEntitysetMobile(String mobile) {

this.mobile = mobile;

return this;

}

public UserEntitysetGender(String gender) {

this.gender = gender;

return this;

}

public UserEntitysetPassword(String password) {

this.password = password;

return this;

}

public UserEntitysetActive(Boolean active) {

terms = active;

return this;

}

public UserEntitysetAccountNonExpired(Boolean accountNonExpired) {

isAccountNonExpired = accountNonExpired;

return this;

}

public UserEntitysetAccountNonLocked(Boolean accountNonLocked) {

isAccountNonLocked = accountNonLocked;

return this;

}

public UserEntitysetCredentialsNonExpired(Boolean credentialsNonExpired) {

isCredentialsNonExpired = credentialsNonExpired;

return this;

}

public UserEntitysetEnabled(Boolean enabled) {

isEnabled = enabled;

return this;

}

public UserEntitysetUserRole(UserRoleuserRole) {

this.userRole = userRole;

return this;

}

public UserEntitysetTerms(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.ResponseEntityExceptionHandler;

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 HttpStatushttpStatus = HttpStatus.valueOf(statusCode);

final HttpResponsehttpResponse =

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.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

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 JWTUtiljwtUtil;

@Autowired

private ApplicationUserDetailsServiceuserDetailsService;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChainfilterChain) 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) {

UserDetailsuserDetails

= userDetailsService.loadUserByUsername(userName);

if (jwtUtil.validateToken(token, userDetails)) {

UsernamePasswordAuthenticationTokenauthenticationToken

= new UsernamePasswordAuthenticationToken(userDetails,

null, userDetails.getAuthorities());

authenticationToken.setDetails(

new WebAuthenticationDetailsSource().buildDetails(request)

);

SecurityContextHolder.getContext().setAuthentication(authenticationToken);

}

}

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 HttpResponsehttpResponse =

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 enumUserPermission {

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 enumUserRole {

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\_showsms WHERE ms.show\_start> CURRENT\_DATE", nativeQuery = true)

List<MovieShowsEntity>findAllUpComing();

@Query(value = "SELECT DISTINCT ON(ms.movie\_id) \* FROM movie\_showsms 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\_showsms WHERE ms.show\_end>= CURRENT\_DATE", nativeQuery = true)

List<MovieShowsEntity>findAllNowPlayingAndUpComing();

@Query(value = "SELECT DISTINCT ON(ms.movie\_id) \* FROM movie\_showsms WHERE ms.show\_end< CURRENT\_DATE", nativeQuery = true)

List<MovieShowsEntity>findAllNotPlaying();

// @Query("FROM MovieShowsEntityms LEFT JOIN ms.bookings b WHERE ms.id = ?1 AND b.dateOfBooking = ?2")

// @Query(value = "SELECT \* FROM movie\_showsms 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\_showsms 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\_showsms 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\_showsms INNER JOIN bookings b ON b.id = :bookingId", nativeQuery = true)

// MovieShowsEntityfindByBookingId(@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.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

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 ApplicationUserDetailsServiceuserDetailsService;

@Autowired

private PasswordEncoderpasswordEncoder;

@Autowired

private JWTFilterjwtFilter;

@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 DaoAuthenticationProviderauthenticationProvider() {

DaoAuthenticationProviderauthenticationProvider = new DaoAuthenticationProvider();

authenticationProvider.setPasswordEncoder(passwordEncoder);

authenticationProvider.setUserDetailsService(userDetailsService);

return authenticationProvider;

}

@Bean

CorsConfigurationSourcecorsConfigurationSource() {

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 AuthenticationManagerauthenticationManager() 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 booleanisAccountNonExpired() {

return user.getIsAccountNonExpired();

}

@Override

public booleanisAccountNonLocked() {

return user.getIsAccountNonLocked();

}

@Override

public booleanisCredentialsNonExpired() {

return user.getIsCredentialsNonExpired();

}

@Override

public booleanisEnabled() {

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.UsernameNotFoundException;

import org.springframework.stereotype.Service;

@Service

public class ApplicationUserDetailsService implements UserDetailsService {

@Autowired

private UserService service;

@Override

public UserDetailsloadUserByUsername(final String username) throws UsernameNotFoundException {

final UserEntityuserEntity = 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.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilderconfigure(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() {

}

}