KONGU ENGINEERING COLLEGE

PERUNDURAI

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE : 22ITL42

COURSE NAME : WEB TECHNOLOGY LABORATORY

Online Movie Ticket Booking System

SUBMITTED BY

IMMANUVEL R

22ITR035

IT – A

OBJECTIVE:

To enhance convenience and accessibility by providing a user-friendly platform that allows customers to book movie tickets anytime and anywhere. It aims to streamline the booking process with secure payment methods, real-time updates on seat availability and showtimes, and automated operations to ensure efficiency and reduce errors.

TECHNOLOGY STACK:

HTML	-	HTML5
CSS	-	CSS3
BOOTSTRAP	-	Bootstrap 5.3. x
NODE JS	-	v20.12.2
MONGO DB	-	7.0.8
ANGULAR	-	18.0.3

DESCRIPTION:

The Online Movie Ticket Booking System is a web-based application designed to provide a convenient and efficient platform for booking movie tickets. It enables users to browse current and upcoming movie listings and select showtimes. The system integrates secure payment gateways to ensure safe transactions, offering G-pay payment options to accommodate user preferences.

Key features include user registration and login, detailed movie information with synopsis and showtime updates. Users receive booking confirmations via SMS for easy ticket collection..

The Online Movie Ticket Booking System aims to improve operational efficiency for theaters by automating the ticketing process, reducing manual errors, and providing valuable data insights for better decision-making. By offering a seamless and enjoyable booking experience, the system strives to increase customer satisfaction and loyalty.

Angular:

Angular is used to build the entire application. With several components and services, it is very easy to integrate all the requirements.

Components used:

- About

  To display the details of the Website.

- Admin

  To have additional privilages to see the booking details.

- Home

  Home page for navigation and displaying movie lists.

- Login

  For user login.

- Movie-booking-form

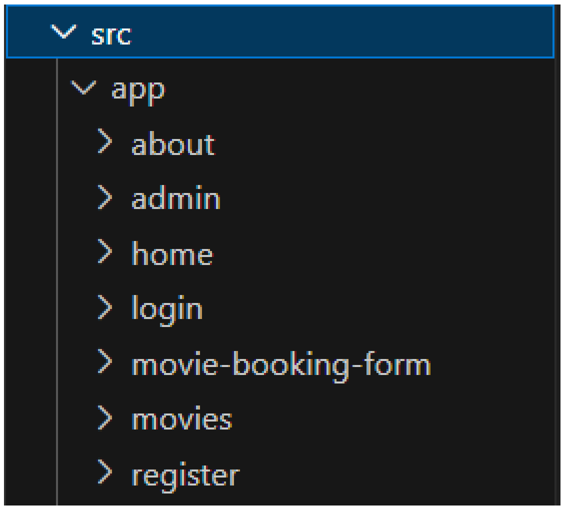
  Form used for booking movies , navigated through home page.

- Movies

  List of upcoming movies , displayed as cards.

- Register

  For user registration.



```
src
  app
    about
    admin
    home
    login
    movie-booking-form
    movies
    register
```

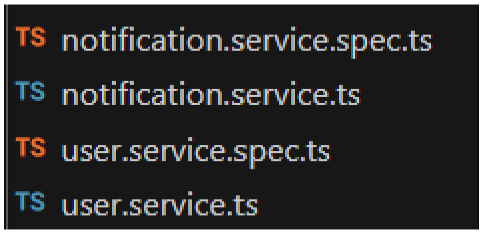
Services:

- Notification service

  To send notification to the user about the booking details.

- User service

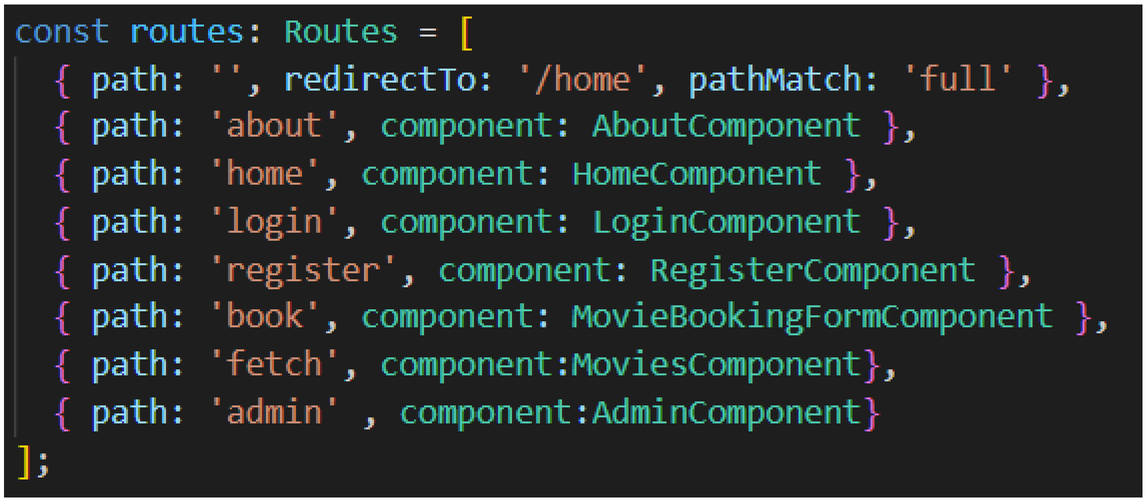
  To use the username of the currently logged in user across the app.



```
TS notification.service.spec.ts
TS notification.service.ts
TS user.service.spec.ts
TS user.service.ts
```

Routing:

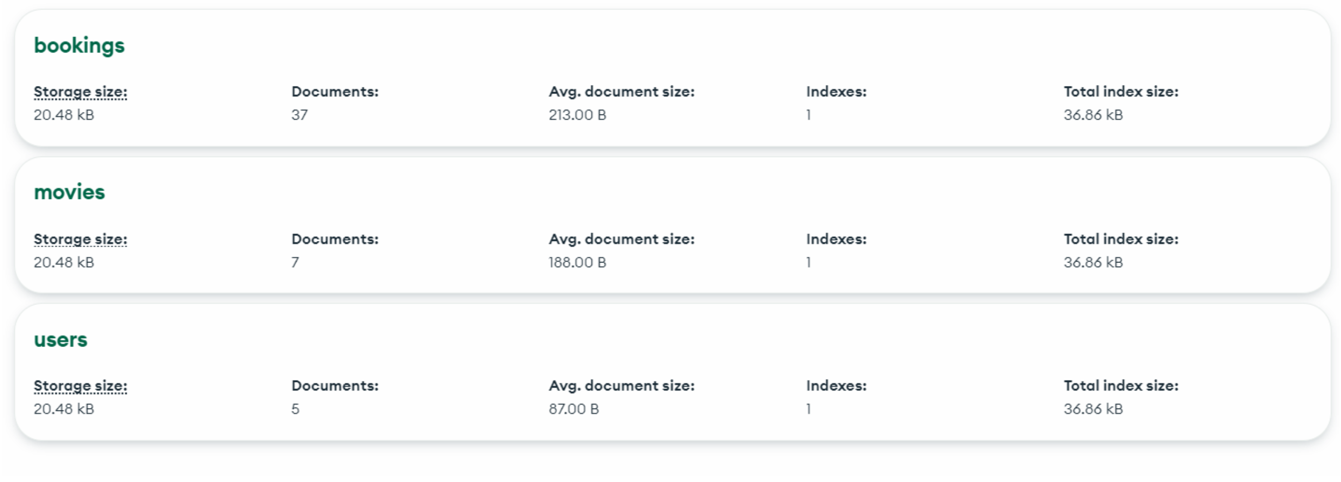
Several routing has been done in order to ensure smooth transition of pages.



```
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'about', component: AboutComponent },
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: 'book', component: MovieBookingFormComponent },
  { path: 'fetch', component:MoviesComponent},
  { path: 'admin' , component:AdminComponent}
];
```

Mongo DB:

Three collection are present under movieBookings Database .



	Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
bookings	20.48 kB	37	213.00 B	1	36.86 kB
movies	20.48 kB	7	188.00 B	1	36.86 kB
users	20.48 kB	5	87.00 B	1	36.86 kB

- Users is used for storing the user registered details and later used for

Login verification.

- Bookings is used for storing the booking details of the users. It is also used to display the admin , the list of movies and theatres booked.
- Movies is a list of upcoming movies .

Coding:

App-routing-module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AboutComponent } from './about/about.component';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { RegisterComponent } from './register/register.component';
import { MovieBookingFormComponent } from './movie-booking-form/movie-booking-form.component';
import { MoviesComponent } from './movies/movies.component';
import { AdminComponent } from './admin/admin.component';
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'about', component: AboutComponent },
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'register', component: RegisterComponent },
  { path: 'book', component: MovieBookingFormComponent },
  { path: 'fetch', component:MoviesComponent},
  { path: 'admin' , component:AdminComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.html

```
<title>Movie Mate</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        nav {
            background-color: #333;
            color: #fff;
            display: flex;
            justify-content: space-between;
            align-items: center;
            padding: 10px 20px;
            margin-bottom: 30px;
        }
        nav a {
            color: #fff;
            text-decoration: none;
            padding: 10px;
        }
        nav a:hover {
            background-color: #555;
        }
        .nav-left {
            display: flex;
            align-items: center;
        }
        .nav-right {
```

```
            display: flex;

            align-items: center;

        }

    </style>


<body>

    <nav>

        <div class="nav-left">

            <h1>Welcome to MovieMate</h1>

            <br><br><br>

            <a routerLink="about">About</a>

            <a routerLink="home">Home</a>

        </div>

        <div class="nav-right">

            <a routerLink="login">Login</a>

            <a routerLink="register">Register</a>

        </div>

    </nav>


    <router-outlet></router-outlet>
```

```
</body>
```

App.component.spec.ts

```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'ticket-booking-system'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('ticket-booking-system');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
        expect(compiled.querySelector('h1')?.textContent).toContain('Hello, ticket-
booking-system');
  });
});
```

App.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { withFetch } from '@angular/common/http';
import { AppComponent } from './app.component';
import { AboutComponent } from './about/about.component';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { RegisterComponent } from './register/register.component';
import { MovieBookingFormComponent } from './movie-booking-form/movie-booking-form.component';
import { MoviesComponent } from './movies/movies.component'; // Ensure MoviesComponent is declared
import { ReactiveFormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { AdminComponent } from './admin/admin.component';
import { GooglePayButtonModule } from'@google-pay/button-angular';
```

```
@NgModule({
 declarations: [
  AppComponent,
  AboutComponent,
  HomeComponent,
  LoginComponent,
  RegisterComponent,
  MovieBookingFormComponent,
  MoviesComponent,
  AdminComponent
 ],
 imports: [
  BrowserModule,
  HttpClientModule,
  AppRoutingModule,
  ReactiveFormsModule,
  CommonModule,
  GooglePayButtonModule
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

App.component.ts

```
import { Component } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'ticket-booking-system';
}
```

Services:

Notification.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class NotificationService {
  private apiUrl = 'http://localhost:3090/send-notification';

  constructor(private http: HttpClient) {}

  sendNotification(phoneNumber: string, message: string) {
    return this.http.post(this.apiUrl, { phoneNumber, message });
  }
}
```

User.service.ts

```
// user.service.ts
import { Injectable } from '@angular/core';


@Injectable({
  providedIn: 'root'
})
export class UserService {
  private username: string = '';


  setUsername(username: string) {
    this.username = username;
    localStorage.setItem('username', username);
  }


  getUsername(): string {
    return this.username || localStorage.getItem('username') || '';
  }
```

```
 clearUsername() {
   this.username = '';
   localStorage.removeItem('username');
 }
}
```

Components

About.component.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About Our Movie Ticket Booking System</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f5f5f5;
```

```
}
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
    text-align: center;
    color: #333;
}
.image-container {
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
}
.image-container img {
    max-width: 100%;
    height: auto;
```

```
            border-radius: 5px;
        }
        p {
            line-height: 1.6;
            color: #555;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>About Our Movie Ticket Booking System</h1>
        <div class="image-container">
            <img src="../../assets/images/image.png" alt="Movie Ticket Booking System">
        </div>
        <p>Welcome to our state-of-the-art online movie ticket booking system! We are dedicated to providing you with a seamless and convenient experience when it comes to purchasing tickets for your favorite movies. Our user-friendly platform is designed to make the process of selecting movies, choosing showtimes, and securing your seats a breeze.</p>
        <p>At our core, we believe in delivering exceptional customer service and ensuring that your movie-going experience is nothing short of remarkable. Our
```

```
team of experts works tirelessly to curate a comprehensive selection of movies
across various genres, ensuring that there is something for everyone to enjoy.</p>

          <p>Our system is built with cutting-edge technology, ensuring a fast and
secure transaction process. Whether you're booking tickets from the comfort of
your home or on-the-go using your mobile device, our platform is optimized for a
smooth and hassle-free experience.</p>

          <p>We take pride in our partnerships with leading cinema chains, offering
you access to a wide range of theaters and showtimes across the country. With our
intuitive search and filtering options, you can easily find the perfect movie, theater,
and showtime that suits your preferences.</p>

            <p>At our online movie ticket booking system, we are committed to
continuously improving and enhancing our services to meet your evolving needs.
We value your feedback and suggestions, as they help us provide an even better
experience for movie enthusiasts like yourself.</p>

          <p>Thank you for choosing our platform, and we look forward to making
your movie-going experience truly unforgettable!</p>
   </div>
</body>
</html>
```

About.component.ts

```
import { Component } from '@angular/core';
```

```
@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styleUrl: './about.component.css'
})
export class AboutComponent {


}
```

Admin.component.html

```
<div class="container">
    <h2>Bookings</h2>
    <table class="table">
      <thead>
        <tr>
          <th>ID</th>
          <th>Movie Name</th>
          <th>Number of Tickets</th>
          <th>Date</th>
          <th>Email</th>
          <th>Phone Number</th>
          <th>Session Timing</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let booking of bookings">
          <td>{{ booking._id }}</td>
          <td>{{ booking.movieName }}</td>
          <td>{{ booking.numberOfTickets }}</td>
          <td>{{ booking.date }}</td>
          <td>{{ booking.email }}</td>
          <td>{{ booking.phoneNumber }}</td>
          <td>{{ booking.sessionTiming }}</td>
        </tr>
      </tbody>
    </table>
  </div>

  <div class="container">
    <h2>Movie Counts</h2>
```

```
    <ul>
        <li *ngFor="let movieCount of movieCounts">{{ movieCount._id }}: {{ movieCount.count }}</li>
    </ul>
  </div>

  <div class="container">
    <h2>Session Counts</h2>
    <ul>
      <li *ngFor="let sessionCount of sessionCounts">{{ sessionCount._id }}: {{ sessionCount.count }}</li>
    </ul>
  </div>
```

Admin.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-admin',
  templateUrl: './admin.component.html',
  styleUrls: ['./admin.component.css']
})
export class AdminComponent implements OnInit {
  bookings: any[] = [];
  theatreCounts: any[] = [];
  movieCounts: any[] = [];
  sessionCounts: any[] = [];

  constructor(private http: HttpClient) {}

  ngOnInit(): void {
    this.loadBookings();
    this.loadTheatreCounts();
    this.loadMovieCounts();
    this.loadSessionCounts();
  }

  loadBookings(): void {
    this.http.get<any[]>('http://localhost:8090/api/bookings').subscribe(
      data => {
```

```
      this.bookings = data;
    },
    error => {
      console.error('Error loading bookings:', error);
    }
  );
}

loadTheatreCounts(): void {
  this.http.get<any[]>('http://localhost:8090/api/theatreCounts').subscribe(
    data => {
      this.theatreCounts = data;
    },
    error => {
      console.error('Error loading theatre counts:', error);
    }
  );
}

loadMovieCounts(): void {
  this.http.get<any[]>('http://localhost:8090/api/movieCounts').subscribe(
    data => {
      this.movieCounts = data;
    },
    error => {
      console.error('Error loading movie counts:', error);
    }
  );
}

loadSessionCounts(): void {
  this.http.get<any[]>('http://localhost:8090/api/sessionCounts').subscribe(
    data => {
      this.sessionCounts = data;
    },
    error => {
      console.error('Error loading session counts:', error);
    }
  );
}
```

```
}
```

home.component.html

```
<body>
        <h1 class="title" style="background-color: black;">Movie Ticket Booking System</h1>
          <h3 style="color: aliceblue;" class="username">Welcome, {{ username }}</h3>
    <marquee>
      <h2>
        <a [routerLink]="['/fetch']" style="text-decoration: none;">
            <p style="text-decoration: none; color: white; border: none; border-radius: 5px; padding: 10px 20px;">
            Find Upcoming Movies
          </p>
        </a>
      </h2>
    </marquee>
    <div class="container">
      <!-- Movie cards -->
      <div class="movie-card" *ngFor="let movie of movies">
```

```
        <img [src]="movie.image" alt="Movie Image">
        <div class="movie-details">
          <div class="movie-title">{{ movie.name }}</div>
          <div class="movie-description">{{ movie.description }}</div>
            <a [routerLink]="['/book']" [queryParams]="{ movieName: movie.name }"
class="cusbut">Book now</a>
        </div>
      </div>
    </div>
  </body>
```

Home.components.ts

```
import { Component, OnInit } from '@angular/core';
import { UserService } from '../user.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
```

```
export class HomeComponent implements OnInit {
  username:string='';
  movies = [
    {
      name: 'Ghili',
       description: 'Saravanavelu "Velu", a Kabaddi player, who goes to Madurai to
participate in a match, but instead rescues Dhanalakshmi from a gang leader who is
obsessed with her, Muthupandi.',
      image: '../../assets/images/ghili.jpeg'
    },
    {
      name: 'Mangatha',
       description: 'The plot of "Mangatha" revolves around Vinayak Mahadevan, a
cunning police officer with ambitions to seize a substantial amount of illicit betting
money from a wealthy entrepreneur.',
      image: '../../assets/images/Mankatha.jpeg'
    },
    {
      name: 'Aranmanai 4',
          description: 'A happy family of four is living their best life, who are
unfortunately soon found dead. Later, during the investigation, it was alleged that
```

```
the husband went to the forest after a spat with his wife and died while she hung
herself.',
      image: '../../assets/images/Aranmanai.jpeg'
    },
    {
      name: 'Spiderman far from home',
        description: 'Nick Fury and Maria Hill investigate an unnatural storm and
encounter the Earth Elemental. Quentin Beck, arrives and defeats the creature.',
      image: '../../assets/images/spiderman far from home.jpeg'
    },
    {
      name: 'Manjummel Boys',
       description: 'The movie narrates the gripping story of a group of friends from
Fort Kochi on a journey to Kodaikanal. Their visit to the Guna Caves takes a
perilous turn when one of them gets trapped, leading to a thrilling rescue mission.',
      image: '../../assets/images/Manjummel Boys.jpeg'
    },
    {
      name: 'Bahubali',
       description: 'A child from the Mahishmati kingdom is raised by tribal people
and one day learns about his royal heritage, his father\'s bravery in battle and a
mission to overthrow the incumbent ruler.',
      image: '../../assets/images/Bahubali.jpg'
    },
    {
      name: 'Dada',
        description: 'Dada is loosely based on true events of Ganesh K. Babu\'s friend\'s life and focuses on a young man\'s journey of adoring the role of fatherhood and a single parent',
      image: '../../assets/images/Dada.jpg'
    },
    {
      name: 'Komban',
        description: 'An angry young man tries to give up his violent ways after his marriage runs into trouble. But his nemesis will not rest until he is killed.',
      image: '../../assets/images/Komban.jpg'
    },
    {
      name: 'Mersal',
        description: 'It is the second out of the three collaboration between Vijay and Atlee, before they began working with others. The film follows two brothers who were separated at birth.',
      image: '../../assets/images/mersal.jpg'
```

```
  },
  {
    name: 'Osthi',
      description: 'Osthe tells the story of a fearless police officer, "Osthi" Velan
(Silambarasan), and his troubled relationship with his stepfather and half-brother as
he embarks on a mission to take down a corrupt politician.',
    image: '../../assets/images/Osthi.jpg'
  },
  {
    name: 'Sarkar',
      description: 'Sundar Ramaswamy is a wealthy and ruthless NRI corporate
raider in the US, referred to as a corporate monster due to his ruthless business
dealings. His imminent arrival in Chennai scares companies since they think they
are a target for him.',
    image: '../../assets/images/Sarkaar.jpg'
  },
  {
    name: 'Soorari Potru',
      description: 'Inspired by the book \'Simply Fly\', the film tells the story of
Nedumaaran Rajangam known to friends as Maara, the son of a teacher, who sets
out to make the common man fly .',
    image: '../../assets/images/Soorarai+pootru.jpg'
  },
  {
    name: 'Thupakki',
    description: 'It revolves around Jagadish, an intelligence officer in the Indian Army, who seeks to track down, destroy and deactivate a sleeper cell, after witnessing and barely escaping a bomb blast executed by the cell.',
    image: '../../assets/images/Thupakki.jpg'
  },
  {
    name: 'Vaaranam Aayiram',
      description: 'The film illustrates Suriya, an Indian Army Major who learns about his father Krishnan\'s death when he is on a rescue mission. He then starts reminiscing about the bond he shared with Krishnan and the stories related to him.',
    image: '../../assets/images/vaaranam_aayiram.jpg'
  }
];

constructor(private userService: UserService) {}
ngOnInit(): void {
  this.username = this.userService.getUsername();
}
```

```
}
```

Login.component.html

```
<div class="container vh-75">
  <div class="row justify-content-center align-items-center">
    <div class="col-md-6">
      <div class="form-container bg-light p-5 rounded">
        <h2 class="text-center mb-4">Login</h2>
        <form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
          <div class="form-group">
            <label for="username" class="form-label">Username</label>
                              <input type="text" class="form-control" id="username"
formControlName="username" required>
                              <div *ngIf="username?.invalid && (username?.dirty ||
username?.touched)" class="text-danger">
                            <small *ngIf="username?.errors?.['required']">Username is
required</small>
            </div>
          </div>
          <div class="form-group">
            <label for="password" class="form-label">Password</label>
            <input type="password" class="form-control" id="password" formControlName="password" required>
            <div *ngIf="password?.invalid && (password?.dirty || password?.touched)" class="text-danger">
              <small *ngIf="password?.errors?.['required']">Password is required</small>
            </div>
          </div>
          <button type="submit" class="btn btn-primary btn-block" [disabled]="loginForm.invalid">Login</button>
          <p class="mt-3 text-center text-danger" *ngIf="errorMessage">{{ errorMessage }}</p>
        </form>
      </div>
    </div>
  </div>
</div>
```

Login.component.ts

```
// LoginComponent (login.ts)
import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
import { UserService } from '../user.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {
  loginForm: FormGroup;
  errorMessage: string = '';
  constructor(private fb: FormBuilder, private http: HttpClient, private router:
Router, private userService: UserService) {
    this.loginForm = this.fb.group({
      username: ['', Validators.required],
      password: ['', Validators.required]
    });
  }

  get username() {
    return this.loginForm.get('username');
  }

  get password() {
    return this.loginForm.get('password');
  }

  onSubmit() {
    if (this.loginForm.valid) {
      this.errorMessage = ''; // Clear any previous error message
```

```
                              this.http.post<any>('http://localhost:8090/api/login',
this.loginForm.value).subscribe(
    response => {
      if (response.success) {
        // Store the username using UserService
        this.userService.setUsername(this.loginForm.value.username);

        // Login successful, navigate based on user role
        if (response.isAdmin) {
          this.router.navigate(['/admin']);
        } else {
          this.router.navigate(['/home']);
        }
      } else {
        // Login failed, display error message
        this.errorMessage = 'Invalid credentials';
      }
    },
    error => {
      console.error('Login error:', error);
```

```
          this.errorMessage = 'Invalid credentials. Please try again!!';
        }
      );
    }
  }
}
```

Movie-booking-form

```
<div class="form-container">
  <form [formGroup]="bookingForm" (ngSubmit)="onSubmit()">
    <p>
      <label for="movieName">Movie Name</label>
      <input type="text" id="movieName" formControlName="movieName" readonly>
      <ng-container *ngIf="getFormControl('movieName')?.touched">
        <div *ngIf="getFormControl('movieName')?.errors?.['required']" class="error-message">Movie name is required</div>
      </ng-container>
    </p>


    <p>
```

```
     <label for="cinemaLocation">Cinema Location</label>
     <select id="cinemaLocation" formControlName="cinemaLocation">
      <option value="" style="color: gray;">Select Cinema Location</option>
         <option *ngFor="let location of theaterLocations" [value]="location">{{
location }}</option>
     </select>
     <ng-container *ngIf="getFormControl('cinemaLocation')?.touched">
               <div  *ngIf="getFormControl('cinemaLocation')?.errors?.['required']"
class="error-message">Cinema location is required</div>
     </ng-container>
   </p>

   <p>
     <label for="numberOfTickets">Number of Tickets</label>
                         <input   type="number"   id="numberOfTickets"
formControlName="numberOfTickets">
     <ng-container *ngIf="getFormControl('numberOfTickets')?.touched">
              <div  *ngIf="getFormControl('numberOfTickets')?.errors?.['required']"
class="error-message">Number of tickets is required</div>
                 <div  *ngIf="getFormControl('numberOfTickets')?.errors?.['min']"
class="error-message">Should be at least 1</div>
```

```
      </ng-container>
    </p>

    <p>
      <label for="date">Date</label>
      <input type="date" id="date" formControlName="date">
      <ng-container *ngIf="getFormControl('date')?.touched">
            <div *ngIf="getFormControl('date')?.errors?.['required']" class="error-
message">Date is required</div>
          <div *ngIf="getFormControl('date')?.errors?.['invalidDate']" class="error-
message">Date must be today or in the future</div>
      </ng-container>
    </p>

    <p>
      <label for="sessionTiming">Session Timing</label>
      <select id="sessionTiming" formControlName="sessionTiming">
        <option value="">Select Session Timing</option>
         <option *ngFor="let timing of sessionTimings" [value]="timing">{{ timing
}}</option>
      </select>
      <ng-container *ngIf="getFormControl('sessionTiming')?.touched">
                    <div   *ngIf="getFormControl('sessionTiming')?.errors?.['required']"
class="error-message">Session timing is required</div>
      </ng-container>
    </p>


    <p>
      <label for="email">Email</label>
      <input type="email" id="email" formControlName="email">
      <ng-container *ngIf="getFormControl('email')?.touched">
               <div  *ngIf="getFormControl('email')?.errors?.['required']"  class="error-
message">This field is required</div>
                  <div  *ngIf="getFormControl('email')?.errors?.['email']"  class="error-
message">Invalid email format</div>
      </ng-container>
    </p>


    <p>
      <label for="phoneNumber">Phone Number</label>
         <input  type="text"  id="phoneNumber"  formControlName="phoneNumber"
placeholder="+1234567890">
```

```
      <ng-container *ngIf="getFormControl('phoneNumber')?.touched">
        <div *ngIf="getFormControl('phoneNumber')?.errors?.['required']"
class="error-message">Phone number is required</div>
        <div *ngIf="getFormControl('phoneNumber')?.errors?.['pattern']"
class="error-message">Invalid phone number format</div>
      </ng-container>
    </p>


    <p>
      <button type="submit" [disabled]="!bookingForm.valid">Book
Tickets</button>
    </p>
  </form>
  <google-pay-button
      environment="TEST"
      buttonSizeMode="fill"
      [paymentRequest]="paymentRequest"
      (loadpaymentdata)="onLoadPaymentData($event)"
      ></google-pay-button>
  <p *ngIf="msg" style="color: aliceblue;">{{ msg }}</p>
  <p *ngIf="bookingConfirmed">
  </p>
  <p *ngIf="msg1" style="color: aliceblue;">{{ msg1 }}</p>
  <p *ngIf="totalPrice !== null" style="color: aliceblue;">Total Price: {{ totalPrice
}} INR</p>
</div>
```

Movie-booking-form.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators, AbstractControl } from
'@angular/forms';
import { ActivatedRoute } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { GooglePayButtonComponent } from '@google-pay/button-angular';
import { GooglePayButtonModule } from '@google-pay/button-angular';
import { Gather } from 'twilio/lib/twiml/VoiceResponse';

@Component({
  selector: 'app-movie-booking-form',
  templateUrl: './movie-booking-form.component.html',
  styleUrls: ['./movie-booking-form.component.css']
})
```

```
export class MovieBookingFormComponent implements OnInit {
  bookingForm: FormGroup;
  msg: string = '';
  msg1: string = '';
  bookingConfirmed: boolean = false;
  sessionTimings: string[] = ['10:00 AM', '1:00 PM', '4:00 PM', '7:00 PM'];
  theaterLocations: string[] = [];
  totalPrice: number = 0;
  ticketPrice: number = 120; // Price per ticket

  movieTheaterMapping: { [key: string]: string[] } = {
    'Ghili': ['Shanthi Theartre', 'Nayanthara Theartre', 'Apsara Theatrre'],
    'Mangatha': ['Shanthi Theartre', 'Cosmos', 'Guna'],
    'Aranmanai 4': ['Nayanthara Theartre', 'Sivakumar', 'Cosmos'],
      'Spiderman far from home': ['Apsara Theatrre', 'PVR,Salem', 'Inox,Salem',
'Santhoshkumar Theatre'],
    'Manjummel Boys': ['Raja theatre', 'Inox,Salem'],
    'Dada': ['Shanthi Theartre'],
    'Komban': ['Raja theatre', 'Santhoshkumar Theatre'],
  };
  paymentRequest!: google.payments.api.PaymentDataRequest;



  constructor(
    private fb: FormBuilder,
    private http: HttpClient,
    private route: ActivatedRoute
  ) {
    this.bookingForm = this.fb.group({
      movieName: ['', [Validators.required]],
      cinemaLocation: ['', [Validators.required]],
      numberOfTickets: ['', [Validators.required, Validators.min(1)]],
      date: ['', [Validators.required, this.futureDateValidator]],
      email: ['', [Validators.required, Validators.email]],
              phoneNumber:  ['',  [Validators.required,  Validators.pattern('^\\+?[1-
9]\\d{1,14}$')]],
      sessionTiming: ['', [Validators.required]]
    });
```

```
}

ngOnInit(): void {
  this.route.queryParams.subscribe(params => {
    const movieName = params['movieName'];
    if (movieName) {
      this.bookingForm.patchValue({ movieName });
      this.updateTheaterLocations(movieName);
    }
  });

  this.bookingForm.get('numberOfTickets')?.valueChanges.subscribe(value => {
    this.calculateTotalPrice(value);



  this.paymentRequest = {

    apiVersion:2,
    apiVersionMinor:0,
    allowedPaymentMethods:[
  {
   type:'CARD',
   parameters:{
     allowedAuthMethods:['PAN_ONLY','CRYPTOGRAM_3DS'],
     allowedCardNetworks:['MASTERCARD','VISA'],
   },
   tokenizationSpecification:{
     type:'PAYMENT_GATEWAY',
     parameters:{
       gateway:'example',
       gatewayMerchantId:'exampleGatewayMerchantId',
     },
   },
  },
],
merchantInfo:{
  merchantId:'17613812255336763067',
  merchantName: 'Demo only(you will not be charged)',
},
transactionInfo:{
      totalPriceStatus:'FINAL',

      totalPriceLabel:'Total',

      totalPrice:this.ticketPrice.toFixed(2),

      currencyCode:'USD',

      countryCode:'US',

    },

  };

  });



}

updateTheaterLocations(movieName: string): void {

  this.theaterLocations = this.movieTheaterMapping[movieName] || [];

}

calculateTotalPrice(ticketCount: number): void {

  this.totalPrice = ticketCount * this.ticketPrice;

}
```

```
onSubmit(): void {
  if (this.bookingForm.valid) {
    this.storeBookingDetails();
    this.msg = 'Booking confirmed!';
    this.bookingConfirmed = true;
  } else {
    this.msg = 'Please correct the errors in the form.';
    this.bookingConfirmed = false;
  }
}

storeBookingDetails(): void {
  const bookingDetails = this.bookingForm.value;

  this.http.post('http://localhost:8090/api/bookings', bookingDetails).subscribe(
    response => console.log('Booking details stored in MongoDB:', response),
    error => console.error('Error storing booking details:', error)
  );

  this.sendBookingReport();
  }

  futureDateValidator(control: AbstractControl): { [key: string]: boolean } | null {
    const inputDate = new Date(control.value);
    const today = new Date();
    today.setHours(0, 0, 0, 0);
    return inputDate >= today ? null : { 'invalidDate': true };
  }

  getFormControl(name: string): AbstractControl | null {
    return this.bookingForm.get(name);
  }

  sendBookingReport(): void {
    const bookingDetails = this.bookingForm.value;
     const message = `Booking confirmed! Movie: ${bookingDetails.movieName},
Location: ${bookingDetails.cinemaLocation}, Tickets:
${bookingDetails.numberOfTickets}, Date: ${bookingDetails.date}, Session:
${bookingDetails.sessionTiming}`;
```

```
        this.http.post('http://localhost:8090/api/sendReport', { phoneNumber: bookingDetails.phoneNumber, message }).subscribe(
      response => this.msg1 = 'Message sent successfully !!',
      error => this.msg1 = 'Message sent successfully !!'
    );
  }



  async onLoadPaymentData(event: Event){
        const paymentData = (event as CustomEvent<google.payments.api.PaymentData>).detail;
  }


}
```

Movies.component.html

```
<body>


<div class="container">
  <div class="row">
    <div class="col-md-4" *ngFor="let movie of movies">
```

```
      <div class="card card-animate">
        <div class="card-body">
          <h5 class="card-title">{{ movie.name }}</h5>
          <p class="card-text">Release Date: {{ movie.releaseDate }}</p>
          <p class="card-text">{{ movie.description }}</p>
        </div>
      </div>
    </div>
  </div>
</div>

</body>
```

Movies.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-movies',
  templateUrl: './movies.component.html',
  styleUrls: ['./movies.component.css']
})
export class MoviesComponent implements OnInit {
  movies: any[] = [];

  constructor(private http: HttpClient) { }

  ngOnInit(): void {
    this.fetchMovies();
  }

  fetchMovies() {
    this.http.get<any[]>('http://localhost:8090/api/movies')
      .subscribe(
        (response) => {
          this.movies = response;
        },
        (error) => {
          console.error('Error fetching movies:', error);
        }
    );
  }
}
```

Register.component.html

```
<body>
  <div class="container vh-75">
    <div class="row justify-content-center align-items-center vh-100">
      <div class="col-md-6">
        <div class="form-container">
          <h2 class="text-center mb-4">Register</h2>
          <form [formGroup]="registerForm" (ngSubmit)="onSubmit()">
            <div class="form-group">
              <label for="username">Username</label><br>
                <input type="text" formControlName="username" class="form-control"
id="username">
                                <p *ngIf="registerForm.get('username')?.touched &&
registerForm.get('username')?.invalid" class="error-message">
                Username must be at least 5 characters long.
              </p>
            </div><br>
      <div class="form-group">
       <label for="password">Password</label><br>
       <div class="input-group">
          <input type="password" formControlName="password" class="form-control" id="password">
       </div>
          <p *ngIf="registerForm.get('password')?.touched && registerForm.get('password')?.invalid" class="error-message">
          Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, one number, and one special character.
       </p>
      </div><br>
      <div class="form-group">
       <label for="confirmPassword">Confirm Password</label><br>
          <input type="password" formControlName="confirmPassword" class="form-control" id="confirmPassword">
          <p *ngIf="registerForm.get('confirmPassword')?.touched && registerForm.get('confirmPassword')?.invalid" class="error-message">
        Passwords do not match.
       </p>
      </div><br><br>
```

```
                        <button type="submit" class="btn btn-primary btn-block" [disabled]="registerForm.invalid">Register</button>
                  <p id="message" class="mt-3 text-center" *ngIf="errorMessage">{{ errorMessage }}</p>
          </form><br>
                      <p class="mt-3 text-center">Already have an account? <a routerLink="/login" class="cusbut">Login</a></p>
        </div>
      </div>
    </div>
  </div>
</body>
```

Register.component.ts

```
import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';

@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent {
  registerForm: FormGroup;
  passwordFieldType: string = 'password';
  errorMessage: string | null = null;

   constructor(private fb: FormBuilder, private http: HttpClient, private router: Router) {
    this.registerForm = this.fb.group({
      username: ['', [Validators.required, Validators.minLength(5)]],
      password: ['', [Validators.required, Validators.minLength(8)]],
      confirmPassword: ['', [Validators.required]]
    }, { validator: this.matchPasswords('password', 'confirmPassword') });
  }

  matchPasswords(password: string, confirmPassword: string) {
    return (formGroup: FormGroup) => {
      const passControl = formGroup.controls[password];
      const confirmPassControl = formGroup.controls[confirmPassword];

      if (confirmPassControl.errors && !confirmPassControl.errors['mustMatch']) {
        return;
      }

      if (passControl.value !== confirmPassControl.value) {
        confirmPassControl.setErrors({ mustMatch: true });
      } else {
        confirmPassControl.setErrors(null);
      }
    };
  }

  togglePasswordVisibility() {
        this.passwordFieldType = this.passwordFieldType === 'password' ? 'text' :
'password';
  }

  onSubmit() {
    if (this.registerForm.valid) {
                                          this.http.post('http://localhost:8090/api/register',
this.registerForm.value).subscribe(
        response => {
          console.log('Registration successful', response);
          this.router.navigate(['/login']);
        },
        error => {
          console.error('Registration failed', error);
          if (error.status === 400) {
            this.errorMessage = 'User already exists';
          } else {
            this.errorMessage = 'An error occurred. Please try again later.';
          }
        }
      );
    }
  }
}
```