//src\app\agent-create-package

//src\app\agent-create-package\agent-create-package.component.css

/\* Dashboard-style background \*/

body {

    margin: 0;

    padding: 0;

    background-color: #e6f0ff;

    font-family: 'Segoe UI', sans-serif;

    color: #0a1f44;

  }

  /\* Form container styled like a dashboard card \*/

  .form-container {

    max-width: 800px;

    margin: 40px auto;

    padding: 30px;

    background-color: #ffffff;

    border-radius: 10px;

    box-shadow: 0 8px 24px rgba(0, 40, 120, 0.15);

  }

  /\* Headings \*/

  h2, h3 {

    text-align: center;

    color: #1b3f91;

    margin-bottom: 20px;

  }

  /\* Form labels \*/

  form label {

    display: block;

    font-weight: 500;

    color: #0e2f6b;

    margin: 14px auto 6px;

    max-width: 600px;

    text-align: left;

  }

  /\* Input, select, and textarea fields \*/

  form input,

  form textarea,

  form select {

    display: block;

    width: 100%;

    max-width: 600px;

    margin: 0 auto 12px;

    padding: 10px 14px;

    font-size: 15px;

    border: 1px solid #b4c9f0;

    border-radius: 6px;

    background-color: #fdfdfd;

    box-sizing: border-box;

    transition: border-color 0.25s ease;

  }

  form input:focus,

  form textarea:focus,

  form select:focus {

    border-color: #1b3f91;

    outline: none;

    box-shadow: 0 0 0 2px rgba(27, 63, 145, 0.1);

  }

  textarea {

    resize: vertical;

    min-height: 100px;

  }

  /\* Checkbox alignment \*/

  label input[type="checkbox"] {

    margin-right: 8px;

    vertical-align: middle;

  }

  /\* Repeating sections (optional) \*/

  form > div[ng-reflect-ng-for-of] {

    background-color: #f5f8fe;

    border: 1px solid #d4def4;

    padding: 20px;

    margin: 20px auto;

    border-radius: 8px;

    max-width: 700px;

  }

  /\* Utility buttons (Add/Remove) \*/

  form button[type="button"]:not(.action-buttons button) {

    background-color: #f1f4fc;

    color: #1b3f91;

    border: 1px solid #c3cce9;

    font-size: 14px;

    padding: 8px 14px;

    border-radius: 5px;

    margin: 10px auto;

    cursor: pointer;

    transition: background-color 0.2s ease;

    display: block;

  }

  form button[type="button"]:not(.action-buttons button):hover {

    background-color: #e0e9fb;

  }

  /\* Divider \*/

  form hr {

    border: none;

    border-top: 1px solid #aac4f1;

    margin: 30px auto;

    width: 100%;

    max-width: 600px;

  }

  /\* Submit and Cancel buttons – side-by-side, same styling \*/

  .action-buttons {

    display: flex;

    justify-content: center;

    gap: 20px;

    margin-top: 30px;

    flex-wrap: wrap;

  }

  .action-buttons button,

  .action-buttons button[type="button"],

  .action-buttons button[type="submit"] {

    background-color: #1b3f91 !important;

    color: #ffffff;

    padding: 10px 20px;

    border-radius: 6px;

    border: none;

    cursor: pointer;

    font-weight: 500;

    font-size: 15px;

    min-width: 150px;

    transition: background-color 0.3s ease;

  }

  .action-buttons button:hover {

    background-color: #274dba;

  }

  /\* Responsive layout \*/

  @media (max-width: 600px) {

    .form-container {

      padding: 20px;

      margin: 20px;

    }

    form label,

    form input,

    form textarea,

    form select,

    form hr {

      max-width: 100%;

    }

    .action-buttons {

      flex-direction: column;

      gap: 12px;

    }

    .action-buttons button {

      width: 100%;

    }

  }

  .checkbox-row {

    display: inline-flex;

    align-items: center;

    gap: 6px;

    margin: 14px auto;

    font-weight: 500;

    color: #0e2f6b;

    max-width: 600px;

  }

//src\app\agent-create-package\agent-create-package.component.html

<div class="'form-container">

    <h2>Create New Travel Package</h2>

    <form (ngSubmit)="submitPackage()" #pkgForm="ngForm">

      <!-- Basic Info -->

      <label>Title: </label>

      <input type="text" placeholder="Title" [(ngModel)]="newPackage.title" name="title" required><br>

      <label>Description: </label><br>

      <textarea placeholder="Description" [(ngModel)]="newPackage.description" name="description" required></textarea><br>

      <label>Duration: </label>

      <input type="number" placeholder="Duration (days)" [(ngModel)]="newPackage.duration" name="duration"><br>

      <label>Price: </label>

      <input type="number" placeholder="Price" [(ngModel)]="newPackage.price" name="price"><br>

      <label>Max Capacity: </label>

      <input type="number" placeholder="Max Capacity" [(ngModel)]="newPackage.maxCapacity" name="maxCapacity"><br>

      <label>Country: </label>

      <input type="text" placeholder="Country" [(ngModel)]="newPackage.country" name="country"><br>

      <label>Destination: </label>

      <input type="text" placeholder="Destination" [(ngModel)]="newPackage.destination" name="destination"><br>

      <label>Trip Type: </label>

       <select [(ngModel)]="newPackage.tripType" name="tripType" required>

        <option value="">-- Select Trip Type --</option>

        <option value="International Trip">International Trip</option>

        <option value="Domestic Trip">Domestic Trip</option>

       </select><br>

      <label>Trip Start Date:</label>

      <input type="date" [(ngModel)]="newPackage.tripStartDate" name="tripStartDate"><br>

      <label>Trip End Date:</label>

      <input type="date" [(ngModel)]="newPackage.tripEndDate" name="tripEndDate"><br>

      <!--image url-->

      <label>Package Image URL:</label>

      <input type="text" placeholder="Enter image link" [(ngModel)]="newPackage.imageUrl"><br>

      <!--Offer Section-->

      <h3>Offer Details</h3>

      <label>Offer Description: </label>

      <textarea placeholder="Offer Description" [(ngModel)]="newPackage.offer.description" name="offerDesc"></textarea><br>

      <label>Coupon Code: </label>

      <input type="text" placeholder="Coupon Code" [(ngModel)]="newPackage.offer.couponCode" name="Coupon Code"><br>

      <label>Discount Percentage: </label>

      <input type="number" placeholder="Discount (%)" [(ngModel)]="newPackage.offer.discountPercentage" name="discountPercentage"><br>

      <div class="offer-active-wrapper">

        <input type="checkbox" [(ngModel)]="newPackage.offer.active" name="offerActive">Offer Active

      </div>

      <!--Flights-->

      <h3>Flight Details</h3>

      <div \*ngFor="let flight of newPackage.flights; let i=index">

        <label>Airline: </label>

        <input type="text" placeholder="Airline" [(ngModel)]="flight.airline" [name]="'airline'+i"><br>

        <label>From City: </label>

        <input type="text" placeholder="From City" [(ngModel)]="flight.fromCity" [name]="'fromCity'+i"><br>

        <label>To City: </label>

        <input type="text" placeholder="To City" [(ngModel)]="flight.toCity" [name]="'toCity'+i"><br>

        <label>Departire Time: </label>

        <input type="text" placeholder="Departure Time" [(ngModel)]="flight.departureTime" [name]="'departureTime'+i"><br>

        <label>Arrival Time: </label>

        <input type="text" placeholder="Arrival Time" [(ngModel)]="flight.arrivalTime" [name]="'arrivalTime'+i"><br>

      </div>

      <!-- Hotels -->

      <h3>Hotel Details</h3>

      <div \*ngFor="let hotel of newPackage.hotels; let i=index">

        <label>Hotel Name: </label>

        <input type="text" placeholder="Hotel Name" [(ngModel)]="hotel.name" [name]="'hotelName'+i"><br>

        <label>City: </label>

        <input type="text" placeholder="City" [(ngModel)]="hotel.city" [name]="'hotelCity'+i"><br>

        <label>Rating: </label>

        <input type="number" placeholder="Rating" [(ngModel)]="hotel.rating" [name]="'hotelRating'+i"><br>

        <label>Nights Staying: </label>

        <input type="number" placeholder="Nights" [(ngModel)]="hotel.nights" [name]="'hotelNights'+i"><br>

        <label>Cost per Night: </label>

        <input type="number" placeholder="Cost/Night" [(ngModel)]="hotel.costPerNight" [name]="'hotelCost'+i"><br>

      </div>

      <!-- Sightseeing -->

      <h3>Sightseeing</h3>

      <div \*ngFor="let sight of newPackage.sightseeing; let i=index">

        <label>Location: </label>

        <input type="text" placeholder="Location" [(ngModel)]="sight.location" [name]="'sightLoc'+i"><br>

        <label>Sight Description: </label>

        <input type="text" placeholder="Description" [(ngModel)]="sight.description" [name]="'sightDesc'+i"><br>

        <button type="button" (click)="removeSightseeing(i)" \*ngIf="newPackage.sightseeing.length > 1">

          "Remove Sightseeing"

        </button>

      </div>

      <button type="button" (click)="addSightseeing()">+ Add Sightseeing</button>

      <!-- Itinerary -->

      <h3>Itinerary</h3>

      <div \*ngFor="let item of newPackage.itinerary; let i=index">

        <label>Day Number: </label>

        <input type="number" placeholder="Day #" [(ngModel)]="item.dayNumber" [name]="'day'+i" placeholder="Day #" required><br>

        <label>Activity Title: </label>

        <input type="text" placeholder="Activity Title" [(ngModel)]="item.activityTitle" [name]="'activityTitle'+i" placeholder="Activity Description" required><br>

        <label>Activity Description: </label>

        <input type="text" placeholder="Activity Description" [(ngModel)]="item.activityDescription" [name]="'activityDesc'+i" required><br>

        <button type="button" (click)="removeItineraryDay(i)" \*ngIf="newPackage.itinerary.length>1">Remove Day</button>

        <hr>

      </div>

      <button type="button" (click)="addItineraryDay()">+ Add Itinerary Day</button>

      <!-- Submit and Cancel -->

      <br />

      <div class="action-buttons">

        <button type="submit">Submit Package</button>

        <button type="button" (click)="cancelForm()">Cancel</button>

      </div>

    </form>

  </div>

//src\app\agent-create-package\agent-create-package.component.ts

import { Component, OnInit, inject } from '@angular/core';

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

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import { Router} from '@angular/router';

@Component({

  selector: 'app-agent-dashboard',

  standalone: true,

  imports: [CommonModule, FormsModule,],

  templateUrl: './agent-create-package.component.html',

  styleUrl: './agent-create-package.component.css'

})

export class AgentCreatePackageComponent {

  agentId :number=0;

  newPackage: any = {

    title:'',

    description:'',

    duration:'',

    price:'',

    maxCapacity:'',

    country:'',

    destination:'',

    tripType:'',

    tripStartDate:'',

    tripEndDate:'',

    active: true,

    offer: { couponCode:'', description:'', discountPercentage: 0, active: false},

    flights: [

      {airline: '', fromCity:'', toCity:'', departureTime:'', arrivalTime:''}

    ],

    hotels: [

      {name:'', city:'', rating:'', nights:'', costPerNight:''}

    ],

    sightseeing: [

      {location:'', description:''}

    ],

    itinerary: [

      {dayNumber:'', activityTitle:'', activityDescription:''}

    ]

  };

  private http= inject( HttpClient);

  private router= inject(Router);

  ngOnInit(): void {

    const storedId = localStorage.getItem('userId');

    this.agentId = storedId ? parseInt(storedId, 10) : 0;

    if(!this.agentId){

      alert('Agent ID nit found. Please login again.');

      this.router.navigate(['/login']);

    }

  }

  submitPackage(){

    const url= `http://localhost:8080/api/packages`;

    const fullData = {

      ...this.newPackage,

      agentId: this.agentId

    };

    this.http.post(url, fullData).subscribe({

      next: () => {

        alert('Package created successfully!');

        this.router.navigate(['/agent-dashboard']);

      },

      error: (err) => {

        console.error('Error correcting package', err);

        alert('Failed to create package. Check your input.');

      }

    });

  }

  //addItinerary

  addItineraryDay(){

    const newDay = this.newPackage.itinerary.length+1;

    this.newPackage.itinerary.push({

      dayNumber: newDay,

      activityTitle: '',

      activityDescription: ''

    });

  }

  //remove Itinerary

  removeItineraryDay(index:number): void{

    this.newPackage.itinerary.splice(index,1);

    this.newPackage.itinerary.forEach((item: any, i: number) =>{

      item.dayNumber = i+1;

    });

  }

  //add Sightseeing

  addSightseeing(): void {

    this.newPackage.sightseeing.push({location:'', description:''});

  }

  //remove sightseeing

  removeSightseeing(index: number): void {

    this.newPackage.sightseeing.splice(index,1);

  }

  cancelForm(): void{

    window.location.href = '/agent-dashboard';

  }

}

src\app\agent-dashboard

//src\app\agent-dashboard\agent-dashboard.component.css

.dashboard-container {

  max-width: 800px;

  margin: 40px auto;

  padding: 30px;

  background-color: #e6f0ff; /\* soft blue backdrop \*/

  border-radius: 10px;

  box-shadow: 0 8px 24px rgba(0, 40, 120, 0.15);

  font-family: 'Segoe UI', sans-serif;

  color: #0a1f44;

}

.dashboard-container h2 {

  text-align: center;

  margin-bottom: 20px;

  color: #1b3f91;

}

button {

  background-color: #1b3f91;

  color: #ffffff;

  padding: 10px 16px;

  margin: 5px 8px 15px 0;

  border: none;

  border-radius: 5px;

  cursor: pointer;

  font-weight: 500;

  transition: background-color 0.3s ease;

}

button:hover {

  background-color: #274dba;

}

hr {

  border: none;

  border-top: 1px solid #aac4f1;

  margin: 20px 0;

}

ul {

  padding-left: 20px;

}

li {

  background-color: #ffffff;

  border-radius: 6px;

  padding: 12px 16px;

  margin-bottom: 10px;

  list-style-type: none;

  box-shadow: 0 2px 6px rgba(0, 40, 120, 0.1);

}

li strong {

  color: #0e2f6b;

}

p {

  text-align: center;

  color: #44658f;

  font-style: italic;

}

.package-actions {

  display: flex;

  justify-content: center;

  gap: 10px;

  margin-top: 10px;

}

//src\app\agent-dashboard\agent-dashboard.component.html

<app-dashboard-navbar></app-dashboard-navbar>

<div class="dashboard-container">

    <h2>Welcome Agent!</h2>

    <button [routerLink]="['/create-package']">➕ Create New Package</button>

    <!-- <button (click)="goToProfile()">👤 My Profile</button> -->

    <hr>

      <div \*ngIf="packages.length > 0; else noPackages">

      <h3>Your Travel Packages:</h3>

      <ul>

        <li \*ngFor="let pkg of packages">

          <strong>{{ pkg.title }}</strong> – {{ pkg.destination }} (₹{{ pkg.price }})<br>

          <!--view details of the package-->

          <div class="package-actions">

            <button [routerLink]="['/view-package', pkg.packageId]">View Details</button>

            <button [routerLink]="['/edit-package', pkg.packageId]">Edit</button>

            <button (click)="deletePackage(pkg.packageId)"> Delete</button>

          </div>

        </li>

      </ul>

    </div>

    <ng-template #noPackages>

      <p>No packages found. Click "Create New Package" to add one!</p>

    </ng-template>

</div>

//src\app\agent-dashboard\agent-dashboard.component.ts

import { Component, OnInit, inject } from '@angular/core';

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

import { RouterLink } from '@angular/router';

import { DashboardNavbarComponent } from '../dashboard-navbar/dashboard-navbar.component';

@Component({

  selector: 'app-agent-dashboard',

  standalone: true,

  imports: [CommonModule, HttpClientModule, FormsModule, RouterLink, DashboardNavbarComponent],

  templateUrl: './agent-dashboard.component.html',

  styleUrl: './agent-dashboard.component.css'

})

export class AgentDashboardComponent implements OnInit{

  packages: any[] = []; //store the list of travel packages

  agentId :number=0;

  private http = inject(HttpClient);

  ngOnInit(): void {

    const storedId = localStorage.getItem('userId');

    this.agentId = storedId ? parseInt(storedId, 10) : 0;

    if (this.agentId>0) {

      this.loadPackages();

    } else {

      console.error('Agent ID not found in localStorage');

      // optionally redirect to login page or show error

    }

  }

  //Get packages created by this agent

  loadPackages(){

    const url = `http://localhost:8080/api/packages/agent/${this.agentId}`;

    this.http.get<any>(url).subscribe({

      next: (res) =>{

        this.packages = res.data;

        console.log("Loaded packages:", this.packages);//optional debug

      } ,

      error: (err) => console.error('Error loading packages', err)

    });

  }

  //Delete a package

  deletePackage(id: number){

    if(!id){

      console.error('Invalid package ID:',id);

      alert('Error: Cannot delete package. Invalid ID.');

      return ;

    }

    const confirmDelete = confirm("Are you sure you want to delete this package?");

    if(!confirmDelete) return;

    const url = `http://localhost:8080/api/packages/${id}`;

    this.http.delete(url).subscribe({

      next: () => {

        alert('Package deleted successfully!');

        this.loadPackages();

      },

      error: (err) => {

        console.error('Error deleting package', err);

        alert('Failed to delete Package.')

      }

    });

  }

  //placeholder for edit

  editPackage(pkg: any){

    alert('Edit package: ${pkg.title} - feature coming soon!');

  }

  //Go to my profile

  goToProfile(){

    window.location.href = '/my-profile';

  }

}

View package css html and ts file

Edit package css html and ts are there

No need of these files right I will tell the work u want to do now

body {

  background: linear-gradient(135deg, #f5f7fa, #c3cfe2);

  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

.container {

  max-width: 700px;

  margin: 30px auto;

  background: #ffffff;

  padding: 25px;

  border-radius: 10px;

  box-shadow: 0 8px 20px rgba(0,0,0,0.1);

}

h2, h3 {

  text-align: center;

  color: #333;

}

.review-card {

  background: #fdfdfd;

  border-radius: 8px;

  padding: 15px;

  margin-bottom: 12px;

  box-shadow: 0 3px 6px rgba(0,0,0,0.08);

  transition: transform 0.2s;

}

.review-card:hover {

  transform: translateY(-3px);

}

.review-card small {

  color: #777;

}

textarea {

  width: 100%;

  padding: 12px;

  border: 1px solid #ccc;

  border-radius: 6px;

  margin-top: 10px;

  transition: border 0.3s, box-shadow 0.3s;

}

textarea:focus {

  border-color: #4caf50;

  box-shadow: 0 0 5px rgba(76, 175, 80, 0.5);

  outline: none;

}

button {

  margin-top: 12px;

  padding: 10px 22px;

  background: #4caf50;

  color: white;

  border: none;

  border-radius: 6px;

  cursor: pointer;

  font-weight: bold;

  transition: background 0.3s;

}

button:hover {

  background: #43a047;

}

.star {

  font-size: 28px;

  cursor: pointer;

  color: #ccc;

  transition: transform 0.2s, color 0.3s;

}

.star:hover, .star.active {

  color: #ff9800;

  transform: scale(1.2);

}

p {

  text-align: center;

  color: #d32f2f;

  margin-top: 15px;

}

//src\app\review-dashboard\review-dashboard.component.css

//src\app\review-dashboard\review-dashboard.component.html

<!-- <div class="container" style="max-width:600px;margin:auto;">

  <h2>Reviews for Package #{{packageId}}</h2>

  <div \*ngIf="reviews.length > 0">

    <div \*ngFor="let r of reviews" style="border:1px solid #ddd;padding:10px;margin-bottom:8px;">

      <strong>Rating:</strong> {{r.rating}} ⭐<br>

      <strong>Comment:</strong> {{r.comment}}<br>

      <small>{{r.timestamp}}</small>

    </div>

  </div>

  <div \*ngIf="reviews.length === 0">

    <p>No reviews yet for this package.</p>

  </div>

  <hr>

  <div \*ngIf="completedBooking && !alreadyReviewed">

    <h3>Leave a Review</h3>

    <div>

      <span \*ngFor="let star of [1,2,3,4,5]"

            (click)="setRating(star)"

            [style.color]="star <= rating ? 'orange' : 'gray'"

            style="font-size:24px;cursor:pointer;">

        ★

      </span>

    </div>

    <textarea [(ngModel)]="comment" rows="3" placeholder="Write your comment here..."

              style="width:100%;margin-top:8px;"></textarea>

    <button (click)="submitReview()" style="margin-top:8px;">Submit Review</button>

  </div>

  <p style="color:red;" \*ngIf="message">{{message}}</p>

</div> -->

<div class="container" style="max-width:600px;margin:auto;">

  <h2>Reviews for Package #{{packageId}}</h2>

  <!-- ✅ Always show reviews list -->

  <div \*ngIf="reviews.length > 0; else noReviews">

    <div \*ngFor="let r of reviews" class="review-card">

      <strong>Rating:</strong> {{r.rating}} ⭐<br>

      <strong>Comment:</strong> {{r.comment}}<br>

      <small>{{r.timestamp}}</small>

    </div>

  </div>

  <ng-template #noReviews>

    <p>No reviews yet for this package.</p>

  </ng-template>

  <hr>

  <!-- ✅ Only show review form if allowed -->

  <div \*ngIf="completedBooking && !alreadyReviewed">

    <h3>Leave a Review</h3>

    <div>

      <span \*ngFor="let star of [1,2,3,4,5]"

            (click)="setRating(star)"

            [class.active]="star <= rating "

            class="star">

        ★

      </span>

    </div>

    <textarea [(ngModel)]="comment" rows="3" placeholder="Write your comment here..."

              style="width:100%;margin-top:8px;"></textarea>

    <button (click)="submitReview()" style="margin-top:8px;">Submit Review</button>

  </div>

  <!-- ✅ Only show message near the form context -->

  <div \*ngIf="!completedBooking || alreadyReviewed" style="margin-top:10px;">

    <p style="color:red;">{{message}}</p>

  </div>

</div>

//src\app\review-dashboard\review-dashboard.component.ts

// // review-dashboard.component.ts

// import { Component, OnInit } from '@angular/core';

// import { ActivatedRoute } from '@angular/router';

// import { HttpClient, HttpErrorResponse } from '@angular/common/http';

// import { CommonModule } from '@angular/common';

// import { FormsModule } from '@angular/forms';

// import { Observable, of } from 'rxjs';

// import { catchError, map } from 'rxjs/operators';

// // Define the structure for a Review as expected from your backend

// interface Review {

//   reviewId?: number;

//   userId: number;

//   packageId: number;

//   rating: number;

//   comment: string;

//   timestamp?: string; // ISO string date

//   agentResponse?: { // Optional agent response

//     responseMessage: string;

//     responseTime: string; // ISO string date

//   };

// }

// @Component({

//   standalone: true,

//   imports: [CommonModule, FormsModule],

//   selector: 'app-review-dashboard',

//   templateUrl: './review-dashboard.component.html',

//   styleUrl: './review-dashboard.component.css' // <-- This line now points to the separate CSS file

// })

// export class ReviewDashboardComponent implements OnInit {

//   packageId!: number;

//   userId: number = Number(sessionStorage.getItem('userId')) || 0;

//   reviews: Review[] = [];

//   canReview: boolean = false;

//   hasReviewed: boolean = false;

//   newReview: { rating: number; comment: string } = { rating: 0, comment: '' };

//   editingReviewId: number | null = null;

//   private readonly REVIEWS\_API\_BASE\_URL = 'http://localhost:8084/api/reviews';

//   private readonly BOOKINGS\_API\_BASE\_URL = 'http://localhost:8082/api/bookings';

//   constructor(private route: ActivatedRoute, private http: HttpClient) {}

//   ngOnInit(): void {

//     this.packageId = Number(this.route.snapshot.paramMap.get('packageId'));

//     if (this.userId === 0) {

//       console.warn('User ID not found in session storage. Please log in.');

//       return;

//     }

//     this.loadReviews();

//     this.checkUserEligibility();

//   }

//   loadReviews(): void {

//     this.http.get<Review[]>(`${this.REVIEWS\_API\_BASE\_URL}/${this.packageId}`)

//       .pipe(

//         catchError((error: HttpErrorResponse) => {

//           console.error(`Error loading reviews for package ${this.packageId}:`, error);

//           return of([]);

//         })

//       )

//       .subscribe(data => {

//         this.reviews = data;

//         console.log('Reviews loaded:', this.reviews);

//       });

//   }

//   checkUserEligibility(): void {

//     this.http.get<boolean>(`${this.BOOKINGS\_API\_BASE\_URL}/user/${this.userId}/package/${this.packageId}/completed`)

//       .pipe(

//         catchError((error: HttpErrorResponse) => {

//           console.error('Error checking booking completion:', error);

//           return of(false);

//         })

//       )

//       .subscribe(hasCompleted => {

//         this.canReview = hasCompleted;

//         if (hasCompleted) {

//           this.http.get<boolean>(`${this.REVIEWS\_API\_BASE\_URL}/exists/${this.userId}/${this.packageId}`)

//             .pipe(

//               catchError((error: HttpErrorResponse) => {

//                 console.error('Error checking if user already reviewed:', error);

//                 return of(false);

//               })

//             )

//             .subscribe(alreadyReviewed => {

//               this.hasReviewed = alreadyReviewed;

//               console.log(`User ${this.userId} completed booking: ${this.canReview}, already reviewed: ${this.hasReviewed}`);

//             });

//         } else {

//           console.log(`User ${this.userId} has NOT completed booking for package ${this.packageId}`);

//         }

//       });

//   }

//   submitReview(): void {

//     if (this.newReview.rating < 1 || this.newReview.rating > 5) {

//       console.error('Rating must be between 1 and 5.');

//       return;

//     }

//     if (!this.newReview.comment.trim()) {

//       console.error('Comment cannot be empty.');

//       return;

//     }

//     const reviewData: Review = {

//       userId: this.userId,

//       packageId: this.packageId,

//       rating: this.newReview.rating,

//       comment: this.newReview.comment

//     };

//     this.http.post<Review>(this.REVIEWS\_API\_BASE\_URL, reviewData)

//       .pipe(

//         catchError((error: HttpErrorResponse) => {

//           console.error('Error submitting review:', error);

//           alert(`Failed to submit review: ${error.error.message || error.message}`);

//           return of(null);

//         })

//       )

//       .subscribe(response => {

//         if (response) {

//           console.log('Review submitted successfully:', response);

//           this.loadReviews();

//           this.checkUserEligibility();

//           this.resetReviewForm();

//         }

//       });

//   }

//   enableEdit(review: Review): void {

//     this.editingReviewId = review.reviewId!;

//     this.newReview = {

//       rating: review.rating,

//       comment: review.comment

//     };

//   }

//   updateReview(): void {

//     if (this.editingReviewId !== null) {

//       const updatedReviewData: Review = {

//         userId: this.userId,

//         packageId: this.packageId,

//         rating: this.newReview.rating,

//         comment: this.newReview.comment

//       };

//       this.http.put<Review>(`${this.REVIEWS\_API\_BASE\_URL}/${this.editingReviewId}`, updatedReviewData)

//         .pipe(

//           catchError((error: HttpErrorResponse) => {

//             console.error('Error updating review:', error);

//             alert(`Failed to update review: ${error.error.message || error.message}`);

//             return of(null);

//           })

//         )

//         .subscribe(response => {

//           if (response) {

//             console.log('Review updated successfully:', response);

//             this.loadReviews();

//             this.cancelEdit();

//           }

//         });

//     }

//   }

//   deleteReview(reviewId: number): void {

//     this.http.delete(`${this.REVIEWS\_API\_BASE\_URL}/${reviewId}`)

//       .pipe(

//         catchError((error: HttpErrorResponse) => {

//           console.error('Error deleting review:', error);

//           alert(`Failed to delete review: ${error.error.message || error.message}`);

//           return of(null);

//         })

//       )

//       .subscribe(response => {

//         if (response) {

//           console.log('Review deleted successfully.');

//           this.loadReviews();

//           this.checkUserEligibility();

//         }

//       });

//   }

//   cancelEdit(): void {

//     this.editingReviewId = null;

//     this.resetReviewForm();

//   }

//   private resetReviewForm(): void {

//     this.newReview = {

//       rating: 0,

//       comment: ''

//     };

//   }

// }

// import { Component, OnInit } from '@angular/core';

// import { ActivatedRoute } from '@angular/router';

// import { HttpClient, HttpHeaders } from '@angular/common/http';

// import { CommonModule } from '@angular/common';

// import { FormsModule } from '@angular/forms';

// @Component({

//   selector: 'app-review-dashboard',

//   standalone: true,

//   imports: [CommonModule, FormsModule],

//   templateUrl: './review-dashboard.component.html'

// })

// export class ReviewDashboardComponent implements OnInit {

//   packageId!: number;

//   userId!: number;

//   completedBooking = false;

//   alreadyReviewed = false;

//   rating = 0;

//   comment = '';

//   reviews: any[] = [];

//   message = '';

//   constructor(

//     private route: ActivatedRoute,

//     private http: HttpClient

//   ) {}

//   ngOnInit(): void {

//     this.packageId = Number(this.route.snapshot.paramMap.get('packageId'));

//     this.userId = Number(localStorage.getItem('userId')); // store user id at login

//     this.checkCompletedBooking();

//     this.loadReviews();

//   }

//   checkCompletedBooking() {

// this.http.get<boolean>(`http://localhost:8086/api/bookings/user/${this.userId}/package/${this.packageId}/completed`)

//       .subscribe({

//         next: (res) => {

//           this.completedBooking = res;

//           if (!this.completedBooking) {

//             this.message = "You can only review this package after completing the trip.";

//           }

//         },

//         error: () => this.message = "Could not validate booking. Try again later."

//       });

//   }

//   loadReviews() {

// this.http.get(`http://localhost:8084/api/reviews/${this.packageId}`)

//       .subscribe({

//         next: (res) => {

// this.reviews = res;

// this.alreadyReviewed = this.reviews.some(r => r.userId === this.userId);

//           if (this.alreadyReviewed) {

//             this.message = "You have already reviewed this package.";

//           }

//         },

//         error: () => this.message = "Failed to load reviews."

//       });

//   }

//   submitReview() {

//     if (this.comment.length < 10) {

//       this.message = "Comment must be at least 10 characters.";

//       return;

//     }

//     const review = {

//       userId: this.userId,

//       packageId: this.packageId,

//       rating: this.rating,

//       comment: this.comment

//     };

// this.http.post('http://localhost:8084/api/reviews', review)

//       .subscribe({

//         next: () => {

//           this.message = "Review submitted successfully.";

//           this.loadReviews(); // reload to show

//         },

//         error: () => this.message = "Failed to submit review."

//       });

//   }

//   setRating(stars: number) {

//     this.rating = stars;

//   }

// }

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute } from '@angular/router';

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

import { CommonModule } from '@angular/common';

import { FormsModule } from '@angular/forms';

@Component({

  selector: 'app-review-dashboard',

  standalone: true,

  imports: [CommonModule, FormsModule],

  templateUrl: './review-dashboard.component.html'

})

export class ReviewDashboardComponent implements OnInit {

  packageId!: number;

  userId!: number;

  completedBooking = false;

  alreadyReviewed = false;

  rating = 0;

  comment = '';

  reviews: any[] = [];

  message = '';

  constructor(

    private route: ActivatedRoute,

    private http: HttpClient

  ) {}

  ngOnInit(): void {

    this.packageId = Number(this.route.snapshot.paramMap.get('packageId'));

    this.userId = Number(localStorage.getItem('userId'));

    this.checkCompletedBooking();

    this.loadReviews();

  }

  checkCompletedBooking() {

    this.http.get<boolean>(`http://localhost:8086/api/bookings/user/${this.userId}/package/${this.packageId}/completed`)

      .subscribe({

        next: (res) => {

          this.completedBooking = res;

          if (!this.completedBooking) {

            this.message = "You can only review this package after completing the trip.";

          }

        },

        error: () => this.message = "Could not validate booking. Try again later."

      });

  }

  loadReviews() {

    this.http.get<any[]>(`http://localhost:8084/api/reviews/${this.packageId}`)

      .subscribe({

        next: (res) => {

          this.reviews = res;

          this.alreadyReviewed = this.reviews.some(r => r.userId === this.userId);

          if (this.alreadyReviewed) {

            this.message = "You have already reviewed this package.";

          }

        },

        error: () => this.message = "Failed to load reviews."

      });

  }

  submitReview() {

    if (this.comment.length < 10) {

      this.message = "Comment must be at least 10 characters.";

      return;

    }

    const review = {

      userId: this.userId,

      packageId: this.packageId,

      rating: this.rating,

      comment: this.comment

    };

    this.http.post('http://localhost:8084/api/reviews', review)

      .subscribe({

        next: () => {

          this.message = "Review submitted successfully.";

          this.loadReviews();

          this.rating = 0;

          this.comment = '';

        },

        error: () => this.message = "Failed to submit review."

      });

  }

  setRating(stars: number) {

    this.rating = stars;

  }

}

//src\app\services\review.service.ts

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

// import { Injectable } from '@angular/core';

// @Injectable({ providedIn: 'root' })

// export class ReviewService {

//   private baseUrl = 'http://localhost:8084/api/reviews';

//   constructor(private http: HttpClient) {}

//   getReviewsByPackage(packageId: number) {

//     return this.http.get<any[]>(`${this.baseUrl}/${packageId}`);

//   }

//   postReview(review: any) {

//     return this.http.post(`${this.baseUrl}`, review);

//   }

//   updateReview(reviewId: number, updatedReview: any) {

//     return this.http.put(`${this.baseUrl}/${reviewId}`, updatedReview);

//   }

//   deleteReview(reviewId: number) {

//     return this.http.delete(`${this.baseUrl}/${reviewId}`);

//   }

//   hasCompletedBooking(userId: number, packageId: number) {

//     return this.http.get(`http://localhost:8086/api/bookings/user/${userId}/package/${packageId}/completed`);

//   }

//   hasAlreadyReviewed(userId: number, packageId: number) {

//     return this.http.get<boolean>(`${this.baseUrl}/exists/${userId}/${packageId}`);

//   }

// }

import { Injectable } from '@angular/core';

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class ReviewService {

private reviewBaseUrl = 'http://localhost:8084/api';

private bookingBaseUrl = 'http://localhost:8086/api/bookings';

  constructor(private http: HttpClient) { }

  // ✅ Get all reviews for a package

  getReviewsByPackageId(packageId: number): Observable<any[]> {

    return this.http.get<any[]>(`${this.reviewBaseUrl}/reviews/${packageId}`);

  }

  // ✅ Post a new review

  postReview(review: any): Observable<any> {

return this.http.post(`${this.reviewBaseUrl}/reviews`, review);

  }

  // ✅ Check if user completed booking for the package (matches your exact endpoint)

  hasCompletedBooking(userId: number, packageId: number): Observable<boolean> {

    return this.http.get<boolean>(`${this.bookingBaseUrl}/user/${userId}/package/${packageId}/completed`);

  }

  // ✅ Get agent responses for a review

  getAgentResponses(reviewId: number): Observable<any[]> {

    return this.http.get<any[]>(`${this.reviewBaseUrl}/agent-responses/${reviewId}`);

  }

  // ✅ Post agent response

  postAgentResponse(agentId: number, reviewId: number, response: any): Observable<any> {

return this.http.post(`${this.reviewBaseUrl}/agent-responses/${agentId}/${reviewId}`, response);

  }

  // ✅ Update agent response

  updateAgentResponse(responseId: number, response: any): Observable<any> {

    return this.http.put(`${this.reviewBaseUrl}/agent-responses/${responseId}`, response);

  }

  // ✅ Delete agent response

  deleteAgentResponse(responseId: number): Observable<any> {

    return this.http.delete(`${this.reviewBaseUrl}/agent-responses/${responseId}`);

  }

}

Backend codefor agent responses

package com.ratings.review.client;

import java.util.Map;

import org.springframework.cloud.openfeign.FeignClient;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import com.ratings.review.dto.TravelPackageDTO;

/\*\*

\* Feign client interface for interacting with the Travel Package Management service.

\* This client is used to retrieve travel package details and validate agent ownership.

\*/

@FeignClient(name = "travel-package-management")

public interface TravelPackageClient {

@GetMapping("/api/packages/internal/agent-id/{packageId}")

Map<String, Long> getAgentIdByPackageId(@PathVariable("packageId") Long packageId);

}

package com.ratings.review.controller;

import com.ratings.review.entity.AgentResponse;

import com.ratings.review.entity.Review;

import com.ratings.review.exception.ResourceNotFoundException;

import com.ratings.review.service.AgentResponseService;

import com.ratings.review.repository.ReviewRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

import java.util.Map;

import java.time.LocalDateTime;

import java.util.HashMap;

/\*\*

\* Controller for handling agent responses to reviews.

\*/

@CrossOrigin(origins = "http://localhost:4200")

@RestController

@RequestMapping("/api/agent-responses")

public class AgentResponseController {

private static final Logger logger = LoggerFactory.getLogger(AgentResponseController.class);

@Autowired

private AgentResponseService agentResponseService;

@Autowired

private ReviewRepository reviewRepository;

@PostMapping("/{agentId}/{reviewId}")

public ResponseEntity<Map<String, Object>> respondToReview(

@PathVariable Long agentId,

@PathVariable Long reviewId,

@RequestBody Map<String, String> requestBody) {

String responseMessage = requestBody.get("responseMessage");

AgentResponse savedResponse = agentResponseService.respondToReview(agentId, reviewId, responseMessage);

Review review = reviewRepository.findById(reviewId)

.orElseThrow(() -> new ResourceNotFoundException("Review not found"));

Map<String, Object> responseBody = new HashMap<>();

responseBody.put("packageId", review.getPackageId());

responseBody.put("reviewId", reviewId);

responseBody.put("reviewMessage", review.getComment());

responseBody.put("responseMessage", savedResponse.getResponseMessage());

responseBody.put("responseTime", savedResponse.getResponseTime());

return ResponseEntity.ok(responseBody);

}

@GetMapping("/{reviewId}")

public ResponseEntity<List<AgentResponse>> getResponsesForReview(@PathVariable Long reviewId) {

logger.info("Fetching responses for Review ID {}", reviewId);

List<AgentResponse> responses = agentResponseService.getResponsesForReview(reviewId);

return ResponseEntity.ok(responses);

}

@PutMapping("/{responseId}")

public ResponseEntity<AgentResponse> updateAgentResponse(

@PathVariable Long responseId,

@RequestBody Map<String, String> requestBody) {

String updatedMessage = requestBody.get("responseMessage");

AgentResponse updatedResponse = agentResponseService.updateResponse(responseId, updatedMessage);

return ResponseEntity.ok(updatedResponse);

}

@DeleteMapping("/{responseId}")

public ResponseEntity<Map<String, String>> deleteAgentResponse(@PathVariable Long responseId) {

agentResponseService.deleteResponse(responseId);

Map<String, String> response = new HashMap<>();

response.put("message", "Agent response deleted successfully");

return ResponseEntity.ok(response);

}

}

// src/main/java/com/ratings/review/dto/TravelPackageDTO.java

**package** com.ratings.review.dto;

**import** lombok.Data;

@Data

**public** **class** TravelPackageDTO {

**private** Long packageId;

**private** Long agentId;

**private** String title;

**private** String imageUrl; // Add this line

**private** String description; // Add this line

**private** **double** price; // Add this line

**private** **int** duration; // Add this line

// ... add any other fields your frontend's package-details.component.html uses

}

package com.ratings.review.entity;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

@Table(name = "agent\_responses")

public class AgentResponse {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long responseId;

@Column(nullable = false)

private Long reviewId;

@Column(nullable = false)

private Long packageId;

@Column(nullable = false)

private Long agentId;

@Column(nullable = false, length = 500)

private String responseMessage;

@Column(nullable = false)

private LocalDateTime responseTime;

// Getters and Setters

public Long getResponseId() { return responseId; }

public void setResponseId(Long responseId) { this.responseId = responseId; }

public Long getReviewId() { return reviewId; }

public void setReviewId(Long reviewId) { this.reviewId = reviewId; }

public Long getPackageId() { return packageId; }

public void setPackageId(Long packageId) { this.packageId = packageId; }

public Long getAgentId() { return agentId; }

public void setAgentId(Long agentId) { this.agentId = agentId; }

public String getResponseMessage() { return responseMessage; }

public void setResponseMessage(String responseMessage) {

if (responseMessage == null || responseMessage.isEmpty()) {

throw new IllegalArgumentException("Response message cannot be empty.");

}

this.responseMessage = responseMessage;

}

public LocalDateTime getResponseTime() { return responseTime; }

public void setResponseTime(LocalDateTime responseTime) { this.responseTime = responseTime; }

}

package com.ratings.review.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.ratings.review.entity.AgentResponse;

import java.util.List;

/\*\*

\* Repository for managing agent responses to reviews.

\*/

@Repository

public interface AgentResponseRepository extends JpaRepository<AgentResponse, Long> {

/\*\*

\* Find all responses related to a specific review.

\*/

List<AgentResponse> findByReviewId(Long reviewId);

}

package com.ratings.review.service;

import com.ratings.review.client.TravelPackageClient;

import com.ratings.review.dto.TravelPackageDTO;

import com.ratings.review.entity.AgentResponse;

import com.ratings.review.entity.Review;

import com.ratings.review.exception.ResourceNotFoundException;

import com.ratings.review.repository.AgentResponseRepository;

import com.ratings.review.repository.ReviewRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Map;

@Service

public class AgentResponseService {

@Autowired

private AgentResponseRepository agentResponseRepository;

@Autowired

private ReviewRepository reviewRepository;

@Autowired

private TravelPackageClient travelPackageClient;

public AgentResponse respondToReview(Long agentId, Long reviewId, String responseMessage) {

Review review = reviewRepository.findById(reviewId)

.orElseThrow(() -> new ResourceNotFoundException("Review not found."));

Long packageId = review.getPackageId();

Map<String, Long> response = travelPackageClient.getAgentIdByPackageId(packageId);

Long actualAgentId = response.get("agentId");

if (!actualAgentId.equals(agentId)) {

throw new ResourceNotFoundException("Unauthorized: You are not the owner of this package.");

}

AgentResponse agentresponse = new AgentResponse();

agentresponse.setReviewId(reviewId);

agentresponse.setPackageId(packageId);

agentresponse.setAgentId(agentId);

agentresponse.setResponseMessage(responseMessage);

agentresponse.setResponseTime(LocalDateTime.now());

return agentResponseRepository.save(agentresponse);

}

public List<AgentResponse> getResponsesForReview(Long reviewId) {

return agentResponseRepository.findByReviewId(reviewId);

}

public AgentResponse updateResponse(Long responseId, String updatedMessage) {

AgentResponse response = agentResponseRepository.findById(responseId)

.orElseThrow(() -> new ResourceNotFoundException("Agent Response not found"));

response.setResponseMessage(updatedMessage);

response.setResponseTime(LocalDateTime.now());

return agentResponseRepository.save(response);

}

public void deleteResponse(Long responseId) {

AgentResponse response = agentResponseRepository.findById(responseId)

.orElseThrow(() -> new ResourceNotFoundException("Agent Response not found"));

agentResponseRepository.delete(response);

}

}

This is the backend code for the agent response table

mysql> select \* from review;

+-----------+------------------------------------------------------------------------+------------+--------+----------------------------+---------+

| review\_id | comment | package\_id | rating | timestamp | user\_id |

+-----------+------------------------------------------------------------------------+------------+--------+----------------------------+---------+

| 1 | Wonderful trip experience! Everything was well organized. | 5 | 4 | 2025-06-23 15:28:56.171707 | 3 |

| 2 | Had an amazing trip! And the guide was very friendly. Will book again. | 6 | 5 | 2025-06-28 14:42:15.570517 | 6 |

| 3 | I experienced a wonderful journey.. | 7 | 5 | 2025-06-28 15:05:40.725539 | 5 |

+-----------+------------------------------------------------------------------------+------------+--------+----------------------------+---------+

3 rows in set (0.00 sec)this is reviews given by custoemrs

For this I want responses by agent for that purpose create the one button for every agent under owner for that package when he clicks that button he should see all reviews of that package and he should responds to that reviews that response should be visible to all customers