src\app\customer-dashboard\customer-dashboard.component.css

.dashboard-container {

  padding: 20px;

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

}

h2 {

  text-align: center;

  margin-bottom: 25px;

  color: #343a40;

}

.package-grid {

  display: grid;

  grid-template-columns: repeat(auto-fill, minmax(340px, 1fr));

  gap: 20px;

}

.package-card {

  background: #f0f8ff;

  border-radius: 10px;

  overflow: hidden;

  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.12);

  transition: transform 0.2s ease, box-shadow 0.2s ease;

  padding: 20px; /\* Increased padding to prevent touching borders \*/

  box-sizing: border-box;

}

.package-card:hover {

  transform: scale(1.02);

  box-shadow: 0 6px 14px rgba(0, 0, 0, 0.18);

  background-color: #e0f7fa; /\* Cool hover color \*/

}

.package-card img {

  width: 100%;

  height: 180px;

  object-fit: cover;

  border-radius: 8px;

  margin-bottom: 15px;

}

.package-details {

  padding: 10px 0;

}

.package-details h3 {

  color: #007bff;

  margin: 0 0 10px 0;

}

.package-details p {

  margin: 6px 0;

  font-size: 14px;

  line-height: 1.5;

}

.buttons {

  display: flex;

  flex-wrap: wrap;

  gap: 10px;

  margin-top: 15px;

}

button {

  flex: 1 1 30%;

  background-color: #007bff;

  color: white;

  padding: 10px;

  border: none;

  border-radius: 6px;

  font-size: 14px;

  cursor: pointer;

  transition: background-color 0.2s ease;

}

button:hover {

  background-color: #17a2b8; /\* Teal hover color \*/

}

.search-box {

  display: flex;

  flex-wrap: wrap;

  gap: 10px;

  margin: 20px auto;

  max-width: 1000px;

  justify-content: center;

  padding: 20px;

  background-color: #f8f9fa;

  border-radius: 10px;

  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

}

.search-box input,

.search-box button {

  padding: 10px;

  font-size: 14px;

  border-radius: 6px;

  border: 1px solid #ccc;

  flex: 1 1 200px;

  min-width: 150px;

}

.search-box button {

  background-color: #007bff;

  color: white;

  border: none;

  cursor: pointer;

  transition: background-color 0.2s ease;

}

.search-box button:hover {

  background-color: #17a2b8;

}

.customer-dashboard {

  padding: 20px;

  max-width: 1200px;

  margin: auto;

}

.search-box {

  display: flex;

  flex-wrap: wrap;

  gap: 10px;

  margin: 20px auto;

  justify-content: center;

  padding: 10px;

  background-color: #f8f9fa;

  border-radius: 10px;

}

.search-box input,

.search-box button {

  padding: 8px;

  font-size: 14px;

  border-radius: 6px;

  border: 1px solid #ccc;

  min-width: 150px;

}

.search-box button {

  background-color: #007bff;

  color: white;

  border: none;

  cursor: pointer;

}

.clear-btn {

  background-color: #dc3545;

  color: white;

}

/\* for ratings css style \*/

/\* src/app/customer-dashboard/customer-dashboard.component.css \*/

/\* ... existing styles ... \*/

.stars {

  color: gold; /\* Default star color for empty stars \*/

  display: inline-block; /\* To keep stars together \*/

}

.star {

  font-size: 20px;

  /\* Adjust this if you want stars to be clickable, but not for display \*/

}

.star.filled {

  color: orange; /\* Color for filled stars \*/

}

/\* For half-filled stars \*/

.star.half-filled {

  position: relative;

  display: inline-block;

  color: gold; /\* Color for the "empty" part of the half star \*/

  overflow: hidden;

  width: 1em; /\* Adjust based on your font-size \*/

}

.star.half-filled:before {

  content: '\2605'; /\* Unicode star character \*/

  position: absolute;

  left: 0;

  width: 50%; /\* Make half of the star orange \*/

  overflow: hidden;

  color: orange; /\* Color for the "filled" part of the half star \*/

}

src\app\customer-dashboard\customer-dashboard.component.html

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

<div class="customer-dashboard">

  <h2 class="text-center">Available Travel Packages</h2>

  <div class="search-box">

    <input

      type="text"

      [(ngModel)]="searchText"

      (input)="searchPackages()"

      placeholder="Search title, destination, description, price"

    />

    <input

      type="number"

      [(ngModel)]="minPrice"

      (input)="searchPackages()"

      placeholder="Min Price"

    />

    <input

      type="number"

      [(ngModel)]="maxPrice"

      (input)="searchPackages()"

      placeholder="Max Price"

    />

    <button (click)="clearSearch()">Clear</button>

  </div>

  <div class="package-grid" \*ngIf="filteredPackages.length > 0">

    <div

      class="package-card"

      \*ngFor="let pack of filteredPackages"

      (click)="viewPackageDetails(pack.packageId)"

      style="cursor: pointer;"

    >

      <h3>{{ pack.title }}</h3>

      <p><strong>Location:</strong> {{ pack.destination }}</p>

      <p><strong>Description:</strong> {{ pack.description }}</p>

      <p><strong>Price:</strong> ₹{{ pack.price }}</p>

      <p><strong>Duration:</strong> {{ pack.duration }} days</p>

    </div>

  </div>

  <!-- 🚫 No Results Message

  <div \*ngIf="filteredPackages.length === 0">

    No packages found.

  </div> -->

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

  <div class="customer-dashboard">

    <h2 class="text-center">Available Travel Packages</h2>

    <!-- 🔍 Compact Search Section -->

    <div class="search-box">

      <input

        type="text"

        [(ngModel)]="searchText"

        placeholder="Search title,country,price,triptype etc.,"

      />

      <input

        type="number"

        [(ngModel)]="minPrice"

        placeholder="Min Price"

      />

      <input

        type="number"

        [(ngModel)]="maxPrice"

        placeholder="Max Price"

      />

      <input

        type="number"

        [(ngModel)]="exactDuration"

        placeholder="Exact Duration (days)"

      />

      <button (click)="searchPackages()">Search</button>

      <button class="clear-btn" (click)="clearSearch()">✕</button>

    </div>

    <!-- 📦 Packages Grid -->

    <div class="package-grid" \*ngIf="filteredPackages.length > 0">

      <div

        class="package-card"

        \*ngFor="let pack of filteredPackages"

        (click)="viewPackageDetails(pack.packageId)"

        style="cursor: pointer;"

      >

        <h3>{{ pack.title }}</h3>

        <p><strong>Location:</strong> {{ pack.destination }}</p>

        <p><strong>Description:</strong> {{ pack.description }}</p>

        <p>Country: {{ pack.country }}</p>

        <p><strong>Trip Type:</strong> {{ pack.tripType }}</p>

        <p><strong>Price:</strong> ₹{{ pack.price }}</p>

        <p><strong>Duration:</strong> {{ pack.duration }} days</p>

        <!-- for displaying average rating -->

        <p>

          <strong>Average Rating:</strong>

          <span \*ngIf="pack.averageRating !== undefined && pack.averageRating > 0">

            {{ pack.averageRating | number:'1.1-1' }}

            <span class="stars">

              <ng-container \*ngFor="let starType of getStarArray(pack.averageRating)">

                <span \*ngIf="starType === 1" class="star filled">&#9733;</span>

                <span \*ngIf="starType === 0.5" class="star half-filled">&#9733;</span> <span \*ngIf="starType === 0" class="star">&#9733;</span>

              </ng-container>

            </span>

          </span>

          <span \*ngIf="pack.averageRating === undefined || pack.averageRating === 0">

            No reviews yet.

          </span>

        </p>

      </div>

    </div>

    <!-- 🚫 No Results Message -->

    <div \*ngIf="filteredPackages.length === 0">

      No packages found.

    </div>

  </div>

src\app\customer-dashboard\customer-dashboard.component.ts

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

// import { TravelPackageService } from '../services/package.service';

// import { TravelPackage } from '../models/travel-package';

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

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

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

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

// @Component({

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

//   standalone: true,

//   imports: [CommonModule, FormsModule, DashboardNavbarComponent],

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

//   styleUrls: ['./customer-dashboard.component.css']

// })

// export class CustomerDashboardComponent implements OnInit {

//   packages: TravelPackage[] = [];

//   filteredPackages: TravelPackage[] = [];

//   searchText: string = '';

//   minPrice: number | null = null;

//   maxPrice: number | null = null;

//   constructor(

//     private packageService: TravelPackageService,

//     private router: Router

//   ) {}

//   ngOnInit(): void {

//     this.packageService.getAllPackages().subscribe({

//       next: (data: any) => {

//         this.packages = Array.isArray(data) ? data : data.data;

//         this.filteredPackages = [...this.packages];

//         console.log('Packages:', this.packages);

//       },

//       error: (err) => console.error("Failed to load packages", err)

//     });

//   }

//   searchPackages(): void {

//     const text = this.searchText.toLowerCase();

//     this.filteredPackages = this.packages.filter(pkg => {

//       const matchesText =

//         pkg.title?.toLowerCase().includes(text) ||

//         pkg.destination?.toLowerCase().includes(text) ||

//         pkg.description?.toLowerCase().includes(text) ||

//         pkg.price?.toString().includes(text);

//       const matchesPrice =

//         (this.minPrice === null || pkg.price >= this.minPrice) &&

//         (this.maxPrice === null || pkg.price <= this.maxPrice);

//       return matchesText && matchesPrice;

//     });

//   }

//   clearSearch(): void {

//     this.searchText = '';

//     this.minPrice = null;

//     this.maxPrice = null;

//     this.filteredPackages = [...this.packages];

//   }

//   viewPackageDetails(packageId: string): void {

//     this.router.navigate(['/package', packageId]);

//   }

// }

// src/app/customer-dashboard/customer-dashboard.component.ts

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

import { TravelPackageService } from '../services/package.service';

import { TravelPackage } from '../models/travel-package';

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

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

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

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

import { forkJoin, of } from 'rxjs';

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

@Component({

  selector: 'app-customer-dashboard',

  standalone: true,

  imports: [CommonModule, FormsModule, DashboardNavbarComponent],

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

  styleUrls: ['./customer-dashboard.component.css']

})

export class CustomerDashboardComponent implements OnInit {

  packages: TravelPackage[] = [];

  filteredPackages: TravelPackage[] = [];

  searchText: string = '';

  minPrice: number | null = null;

  maxPrice: number | null = null;

  exactDuration: number | null = null;

  constructor(

    private packageService: TravelPackageService,

    private router: Router

  ) {}

  ngOnInit(): void {

    this.loadPackagesWithRatings();

  }

  loadPackagesWithRatings(): void {

    this.packageService.getAllPackages().subscribe({

      next: (data: TravelPackage[]) => {

        this.packages = data;

        this.filteredPackages = [...this.packages];

        this.fetchAverageRatings(); // Call after packages are loaded

      },

      error: (err) => console.error("Failed to load packages", err)

    });

  }

  fetchAverageRatings(): void {

    if (this.packages.length === 0) {

      return;

    }

    const ratingRequests = this.packages.map(pkg =>

      // pkg.packageId is already number here, so no conversion needed

      this.packageService.getAverageRatingForPackage(pkg.packageId).pipe(

        map(avgRating => ({ packageId: pkg.packageId, averageRating: avgRating })),

        catchError(error => {

          console.error(`Error fetching average rating for package ${pkg.packageId}:`, error);

          return of({ packageId: pkg.packageId, averageRating: 0.0 }); // Default to 0.0 on error

        })

      )

    );

    forkJoin(ratingRequests).subscribe({

      next: (ratings: { packageId: number, averageRating: number }[]) => { // packageId is now number

        ratings.forEach(rating => {

          const pkg = this.packages.find(p => p.packageId === rating.packageId);

          if (pkg) {

            pkg.averageRating = rating.averageRating;

          }

        });

        this.searchPackages(); // Re-filter to ensure average ratings are applied to the displayed packages

      },

      error: (err) => console.error("Failed to fetch average ratings", err)

    });

  }

  searchPackages(): void {

    const text = this.searchText.toLowerCase();

    this.filteredPackages = this.packages.filter(pkg => {

      const matchesText =

        pkg.title?.toLowerCase().includes(text) ||

        pkg.destination?.toLowerCase().includes(text) ||

        pkg.description?.toLowerCase().includes(text) ||

        pkg.country?.toLowerCase().includes(text) ||

        pkg.tripType?.toLowerCase().includes(text) ||

        pkg.price?.toString().includes(text);

      const matchesPrice =

        (this.minPrice === null || pkg.price >= this.minPrice) &&

        (this.maxPrice === null || pkg.price <= this.maxPrice);

      const matchesDuration =

        this.exactDuration === null || pkg.duration === this.exactDuration;

      return matchesText && matchesPrice && matchesDuration;

    });

  }

  clearSearch(): void {

    this.searchText = '';

    this.minPrice = null;

    this.maxPrice = null;

    this.exactDuration = null;

    this.filteredPackages = [...this.packages];

  }

  // Changed packageId type to number

  viewPackageDetails(packageId: number): void {

    this.router.navigate(['/package', packageId]);

  }

  getStarArray(rating: number | undefined): number[] {

    if (rating === undefined || rating === 0) {

      return [];

    }

    const fullStars = Math.floor(rating);

    const halfStar = rating % 1 >= 0.5 ? 1 : 0;

    const emptyStars = 5 - fullStars - halfStar;

    return [

      ...Array(fullStars).fill(1),

      ...(halfStar ? [0.5] : []),

      ...Array(emptyStars).fill(0)

    ];

  }

}

src\app\models

// review.model.ts

export interface Review {

    id?: number;

    userId: number;

    packageId: string;

    comment: string;

    rating: number;

    agentResponse?: string;

  }

src\app\models\travel-package.ts

export interface TravelPackage {

    // packageId: string;

    packageId: number,

    title: string;

    description: string;

    destination: string;

    price: number;

    duration: number;

    itinerary: string;

    features: string;

    startDate: string;

    endDate: string;

    imageUrl: string;

    termsAndConditions: string;

country: string; // ✅ Add this

  tripType: string;

location: string;

averageRating?: number;

  }

//src\app\package-details

//src\app\package-details\package-details.component.css

.package-container {

    max-width: 1000px;

    margin: auto;

    padding: 20px;

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

    background: #ffffff;

    border-radius: 10px;

    box-shadow: 0 2px 15px rgba(0,0,0,0.1);

  }

  .top-section {

    display: flex;

    gap: 20px;

    flex-wrap: wrap;

    margin-bottom: 20px;

  }

  .card {

    background: #f9f9f9;

    border-radius: 8px;

    padding: 12px;

    box-shadow: 0 1px 4px rgba(0,0,0,0.05);

    flex: 1;

    min-height: 160px;

  }

  .image-card {

    max-width: 400px;

  }

  .package-image {

    width: 100%;

    height: 250px;

    border-radius: 10px;

    object-fit: cover;

    background-color: #eee;

  }

  .details-card h2 {

    margin-bottom: 5px;

    color: #007bff;

  }

  .desc {

    font-size: 14px;

    color: #444;

    margin-bottom: 10px;

  }

  .info-grid {

    display: grid;

    grid-template-columns: 1fr 1fr;

    gap: 10px 20px;

    font-size: 14px;

    margin-bottom: 15px;

  }

  .action-buttons {

    display: flex;

    gap: 10px;

    flex-wrap: wrap;

  }

  .btn {

    padding: 8px 16px;

    border-radius: 5px;

    color: white;

    font-weight: bold;

    cursor: pointer;

    border: none;

  }

  .btn.book {

    background-color: #28a745;

  }

  .btn.review {

    background-color: #007bff;

  }

  .btn.insurance {

    background-color: #ffc107;

    color: #000;

  }

  /\* .section-grid {

    display: grid;

    grid-template-columns: repeat(3, 1fr);

    gap: 15px;

  }

  .section-grid {

    display: grid;

    grid-template-columns: repeat(3, 1fr);

    gap: 16px;

    margin-top: 20px;

  } \*/

  /\* .section-grid .card {

    background: #f9f9f9;

    border-radius: 8px;

    padding: 12px 16px;

    box-shadow: 0 1px 6px rgba(0, 0, 0, 0.05);

    min-height: 160px;

    display: flex;

    flex-direction: column;

    justify-content: space-between;

  }

  .section-grid .card h3 {

    margin-bottom: 8px;

    font-size: 16px;

    color: #007bff;

  }

  .section-grid .card ul {

    padding-left: 20px;

    margin: 0;

  }

  .section-grid .card li {

    font-size: 14px;

    margin-bottom: 6px;

  }

  .section-grid .card p {

    font-size: 14px;

    margin: 0;

  }

   \*/

   .section-grid {

    display: grid;

    grid-template-columns: repeat(3, 1fr);

    gap: 16px;

    margin-top: 20px;

  }

  .section-grid .card {

    background: #f9f9f9;

    border-radius: 6px;

    padding: 12px;

    box-shadow: 0 1px 4px rgba(0, 0, 0, 0.05);

    min-height: 200px;

    display: flex;

    flex-direction: column;

    justify-content: flex-start;

  }

  .section-grid .card h3 {

    margin: 0 0 4px 0; /\* Only one line space below heading \*/

    font-size: 15px;

    color: #007bff;

  }

  .section-grid .card ul {

    padding-left: 18px;

    margin: 0;

    list-style: none;

    flex-grow: 1;

  }

  .section-grid .card li {

    font-size: 13px;

    margin-bottom: 4px;

    line-height: 1.4;

  }

  .section-grid .card p {

    font-size: 13px;

    margin: 0;

    line-height: 1.4;

    flex-grow: 1;

  }

//src\app\package-details\package-details.component.html

<div class="package-container">

  <div class="top-section">

    <div class="card image-card">

      <img

        class="package-image"

        [src]="packageDetails.imageUrl || 'https://via.placeholder.com/400x250?text=Package+Image'"

        alt="Package Image"

      />

    </div>

    <div class="card details-card">

      <h2>{{ packageDetails.title }}</h2>

      <p class="desc">{{ packageDetails.description }}</p>

      <div class="info-grid">

        <div><strong>Destination:</strong> {{ packageDetails.destination }}</div>

        <div><strong>Country:</strong> {{ packageDetails.country }}</div>

        <div><strong>Trip Type:</strong> {{ packageDetails.tripType }}</div>

        <div><strong>Duration:</strong> {{ packageDetails.duration }} days</div>

        <div><strong>Price:</strong> ₹{{ packageDetails.price }}</div>

        <div><strong>Max Capacity:</strong> {{ packageDetails.maxCapacity }}</div>

        <div><strong>Start Date:</strong> {{ packageDetails.tripStartDate | date: 'MMM d, y' }}</div>

        <div><strong>End Date:</strong> {{ packageDetails.tripEndDate | date: 'MMM d, y' }}</div>

      </div>

      <div class="action-buttons">

        <button class="btn book" (click)="goToBooking()">Book Now</button>

        <button class="btn review" (click)="goToReview()">Give Review</button>

        <button class="btn insurance" (click)="addInsurance()">Insurance</button>

      </div>

    </div>

  </div>

  <div class="section-grid">

    <div class="card" \*ngIf="packageDetails.highlights?.length">

      <h3>Highlights</h3>

      <ul>

        <li \*ngFor="let h of packageDetails.highlights">🌟 {{ h }}</li>

      </ul>

    </div>

    <div class="card" \*ngIf="packageDetails.flights?.length">

      <h3>Flights</h3>

      <ul>

        <li \*ngFor="let f of packageDetails.flights">

          ✈️ {{ f.airline }} — {{ f.fromCity }} → {{ f.toCity }} ({{ f.departureTime }} - {{ f.arrivalTime }})

        </li>

      </ul>

    </div>

    <div class="card" \*ngIf="packageDetails.hotels?.length">

      <h3>Hotels</h3>

      <ul>

        <li \*ngFor="let h of packageDetails.hotels">

          🏨 {{ h.name }}, {{ h.city }} — {{ h.rating }}★ ({{ h.nights }} nights)

        </li>

      </ul>

    </div>

    <div class="card" \*ngIf="packageDetails.sightseeing?.length">

      <h3>Sightseeing</h3>

      <ul>

        <li \*ngFor="let s of packageDetails.sightseeing">

          📍 {{ s.location }} - {{ s.description }}

        </li>

      </ul>

    </div>

    <div class="card" \*ngIf="packageDetails.itinerary?.length">

      <h3>Itinerary</h3>

      <ul>

        <li \*ngFor="let i of packageDetails.itinerary">

          🗓️ Day {{ i.dayNumber }}: {{ i.activityTitle }} - {{ i.activityDescription }}

        </li>

      </ul>

    </div>

    <div class="card" \*ngIf="packageDetails.offer">

      <h3>Special Offer</h3>

      <p>

        🎁 <strong>{{ packageDetails.offer.couponCode }}</strong> —

        {{ packageDetails.offer.description }} ({{ packageDetails.offer.discountPercentage }}% OFF)

      </p>

    </div>

  </div>

</div>

//src\app\package-details\package-details.component.ts

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

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

 import { TravelPackageService } from '../services/package.service';

import { TravelPackage } from '../models/travel-package';

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

@Component({ selector: 'app-package-details',

  standalone: true,

   imports: [CommonModule],

    templateUrl: './package-details.component.html',

    styleUrl: './package-details.component.css' })

  export class PackageDetailsComponent implements OnInit {

     packageId!: number;

    // travelPackage!: TravelPackage;

   packageDetails:any;

constructor( private route: ActivatedRoute,

   private packageService: TravelPackageService,

   private router: Router ) {}

   ngOnInit(): void {

    // Convert packageId from route snapshot (string) to number

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

    if (idParam) {

        this.packageId = Number(idParam);

        console.log('Opened Package ID:', this.packageId);

        this.packageService.getPackageById(this.packageId).subscribe({

            next: (response: any) => {

                console.log('Fetched package details:', response);

                // Corrected assignment: handle both direct object and 'data' wrapper

                this.packageDetails = response.data || response;

            },

            error: (err) => {

                console.error('Failed to fetch package details', err);

                this.packageDetails = undefined; // Clear details on error

            }

        });

    } else {

        console.error('Package ID not found in route parameters.');

        this.packageDetails = undefined;

        // Optionally, navigate away or show an error message

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

    }

  }

    // Ensure these methods pass a number for packageId

  goToReview(): void {

    if (this.packageId) {

      this.router.navigate(['/review-dashboard', this.packageId]);

    }

  }

  goToBooking(): void {

    if (this.packageId) {

      this.router.navigate(['/booking', this.packageId]);

    }

  }

  addInsurance(): void {

    if (this.packageId) {

      this.router.navigate(['/insurance', this.packageId]);

    }

  }

  }

//src\app\review-dashboard

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

.review-dashboard {

    width: 70%;

    margin: auto;

    font-family: Arial, sans-serif;

  }

  .review {

    background: #f9f9f9;

    border-radius: 10px;

    padding: 15px;

    margin-bottom: 20px;

  }

  .review-header {

    display: flex;

    justify-content: space-between;

    align-items: center;

  }

  .stars {

    color: gold;

  }

  .star {

    font-size: 20px;

    cursor: pointer;

  }

  .star.filled {

    color: orange;

  }

  .comment {

    font-size: 16px;

    margin: 10px 0;

  }

  .agent-response {

    background-color: #eef5ff;

    padding: 10px;

    border-left: 4px solid #0077ff;

    margin-top: 10px;

  }

  .review-form {

    background-color: #fff3cd;

    border: 1px solid #ffeeba;

    padding: 20px;

    border-radius: 10px;

  }

  textarea {

    width: 100%;

    height: 80px;

    padding: 8px;

    margin-top: 8px;

  }

  button {

    margin-top: 10px;

    margin-right: 8px;

    padding: 6px 14px;

    border: none;

    border-radius: 5px;

    cursor: pointer;

  }

  button:hover {

    background-color: #0077ff;

    color: white;

  }

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

<div class="review-dashboard">

  <h2>Package Reviews</h2>

  <!-- Review List -->

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

    <div class="review-header">

      <div class="stars">

        <span \*ngFor="let star of [1,2,3,4,5]" class="star" [class.filled]="review.rating >= star">&#9733;</span>

      </div>

      <p><strong>User {{ review.userId }}</strong> on {{ review.timestamp | date:'short' }}</p>

    </div>

    <p class="comment">{{ review.comment }}</p>

    <!-- Agent Response -->

    <div \*ngIf="review.agentResponse" class="agent-response">

      <p><strong>Agent Response:</strong> {{ review.agentResponse.responseMessage }}</p>

    </div>

    <!-- User actions -->

    <div \*ngIf="review.userId === userId">

      <button (click)="enableEdit(review)">Edit</button>

      <button (click)="deleteReview(review.reviewId)">Delete</button>

    </div>

  </div>

  <!-- Add or Edit Review -->

  <div class="review-form" \*ngIf="canReview || editingReviewId !== null">

    <h3>{{ editingReviewId ? 'Update' : 'Add' }} Your Review</h3>

    <label>Rating:</label>

    <div class="stars">

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

            [class.filled]="newReview.rating >= star"

            (click)="newReview.rating = star">&#9733;</span>

    </div>

    <textarea [(ngModel)]="newReview.comment" name="comment" placeholder="Write your comment..."></textarea>

    <br/>

    <button (click)="editingReviewId ? updateReview() : submitReview()">

      {{ editingReviewId ? 'Update' : 'Submit' }}

    </button>

    <button \*ngIf="editingReviewId" (click)="cancelEdit()">Cancel</button>

  </div>

</div>

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

// review-dashboard.component.ts

// =============================

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

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

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

import { ReviewService } from '../services/review.service';

import { Review } from '../models/review.model';

import { FormBuilder, FormGroup, Validators, ReactiveFormsModule } from '@angular/forms';

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

@Component({

  selector: 'app-review-dashboard',

  standalone: true,

  imports: [CommonModule, ReactiveFormsModule,FormsModule],

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

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

})

export class ReviewDashboardComponent implements OnInit {

  packageId!: number;

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

  newReview={

    rating:0,

    comment:'',

    userId:0,

    packageId:0

   };

  reviews: any[] = [];

  editingReviewId: number | null = null;

  hasAlreadyReviewed: boolean = false;

  canReview: boolean = false;

  constructor(private route: ActivatedRoute, private reviewService: ReviewService) {}

  ngOnInit(): void {

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

    this.loadReviews();

    this.checkUserEligibility();

  }

  loadReviews() {

    this.reviewService.getReviewsByPackage(this.packageId).subscribe(data => {

      this.reviews = data;

    });

  }

  checkUserEligibility() {

    this.reviewService.hasCompletedBooking(this.userId, this.packageId).subscribe(hasCompleted => {

      if (hasCompleted) {

        this.reviewService.hasAlreadyReviewed(this.userId, this.packageId).subscribe(already => {

          this.hasAlreadyReviewed = already;

          this.canReview = !already;

        });

      }

    });

  }

  submitReview() {

    this.reviewService.postReview({

      userId: this.userId,

      packageId: this.packageId,

      rating: this.newReview.rating,

      comment: this.newReview.comment

    }).subscribe(() => {

      this.loadReviews();

      this.canReview = false;

      this.newReview = {

        rating: 0,

        comment: '',

        userId: this.userId,

        packageId: this.packageId

      };

    });

  }

  enableEdit(review: any) {

    this.editingReviewId = review.reviewId;

    this.newReview = {

      rating: review.rating,

      comment: review.comment,

      userId: this.userId,

      packageId: this.packageId

    };

     }

  updateReview() {

    if (this.editingReviewId != null) {

      this.reviewService.updateReview(this.editingReviewId, this.newReview).subscribe(() => {

        this.loadReviews();

        this.editingReviewId = null;

        this.newReview = {

          rating: 0,

          comment: '',

          userId: this.userId,

          packageId: this.packageId

        };

        });

    }

  }

  deleteReview(reviewId: number) {

    this.reviewService.deleteReview(reviewId).subscribe(() => {

      this.loadReviews();

      this.canReview = true;

    });

  }

  cancelEdit() {

    this.editingReviewId = null;

    this.newReview = {

      rating: 0,

      comment: '',

      userId: this.userId,

      packageId: this.packageId

    };

    }

}

//src\app\services

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

// src/app/services/package.service.ts

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

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

import { Observable } from 'rxjs';

import { map } from 'rxjs/operators';

import { TravelPackage } from '../models/travel-package';

@Injectable({

  providedIn: 'root'

})

export class TravelPackageService {

  private baseUrl = 'http://localhost:8086/api/bookings/packages'; // Assuming this is correct for your package management service

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

  constructor(private http: HttpClient) {}

  getAllPackages(): Observable<TravelPackage[]> { // Type as TravelPackage[] for consistency

    return this.http.get<any>(`${this.baseUrl}`).pipe( // Use any first, then map to TravelPackage[]

      map(response => Array.isArray(response) ? response : response.data)

    );

  }

  // CORRECTED: Changed packageId type to number

  getPackageById(packageId: number): Observable<TravelPackage> { // Type as TravelPackage

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

      map(response => response.data || response) // Robustly get the package data

    );

  }

  // Method to get average rating for a package from the RatingsAndReviews service

  getAverageRatingForPackage(packageId: number): Observable<number> {

    // The backend endpoint is /api/reviews/{packageId}/average-rating

    return this.http.get<number>(`${this.reviewBaseUrl}/${packageId}/average-rating`);

  }

}

//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 Routes type from Angular Router

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

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

import { LoginComponent } from './login/login.component';

import { LandingComponent } from './landing/landing.component';

import { RegisterComponent } from './register/register.component';

import { AdminUserManageComponent } from './admin-user-manage/admin-user-manage.component';

import { CustomerHelpComponent } from './customer-help/customer-help.component';

import { CustomerInsuranceComponent } from './customer-insurance/customer-insurance.component';

import { LandingNavbarComponent } from './landing-navbar/landing-navbar.component';

import { CustomerDashboardComponent } from './customer-dashboard/customer-dashboard.component';

import { PackageDetailsComponent } from './package-details/package-details.component'; // <--- ADD THIS LINE

import { ReviewDashboardComponent } from './review-dashboard/review-dashboard.component'; // <--- ADD THIS LINE

export const routes: Routes = [

  { path: '', component: LandingComponent },

  { path: 'register', component: RegisterComponent },

  { path: 'login', component: LoginComponent },

  { path: 'packages/:packageId', component: PackageDetailsComponent },     // <--- ADD THIS LINE

  { path: 'review-dashboard/:packageId', component: ReviewDashboardComponent }, // <--- ADD/ENSURE THIS LINE

  // Dashboard placeholders

  { path: 'admin-dashboard', loadComponent: () => import('./admin-dashboard/admin-dashboard.component').then(m => m.AdminDashboardComponent) },

  { path: 'agent-dashboard', loadComponent: () => import('./agent-dashboard/agent-dashboard.component').then(m => m.AgentDashboardComponent) },

  { path: 'customer-dashboard', loadComponent: () => import('./customer-dashboard/customer-dashboard.component').then(m => m.CustomerDashboardComponent) },

  { path: 'my-profile', loadComponent: () => import('./my-profile/my-profile.component').then(m => m.MyProfileComponent) },

  { path: 'admin/users', loadComponent: () => import('./admin-user-manage/admin-user-manage.component').then(m => m.AdminUserManageComponent) },

  { path: 'update-profile', loadComponent: () => import('./update-profile/update-profile.component').then(m => m.UpdateProfileComponent) },

  { path: 'admin-user-manage' , loadComponent: () => import('./admin-user-manage/admin-user-manage.component').then(m =>AdminUserManageComponent)},

  { path: 'admin/agent-packages/:id', loadComponent: () => import('./admin-agent-packages/admin-agent-packages.component').then(m => m.AdminAgentPackagesComponent)},

  { path: 'forgot-password', loadComponent: () => import('./forgot-password/forgot-password.component').then(m => m.ForgotPasswordComponent)},

  //For creating new package by agent

  { path: 'create-package', loadComponent:() => import('./agent-create-package/agent-create-package.component').then(m=> m.AgentCreatePackageComponent)},

  //for landing navbar

  { path: 'about', loadComponent: () => import('./about/about.component').then(m => m.AboutComponent) },

{ path: 'contact-us', loadComponent: () => import('./contact-us/contact-us.component').then(m => m.ContactUsComponent) },

  //For package editing/updating in agent dashboard

  { path: 'edit-package/:id', loadComponent:() => import('./edit-package/edit-package.component').then(m => m.EditPackageComponent)},

  //For viewing the details of the package by the agent(retrieve)

  { path: 'view-package/:id', loadComponent:() => import('./view-package/view-package.component').then(m => m.ViewPackageComponent)},

  //for help and insurance

  {

    path: 'customer-help',

    loadComponent: () =>

      import('./customer-help/customer-help.component').then(

        m => m.CustomerHelpComponent

      ),

  },

  {

    path: 'customer-insurance',

    loadComponent: () =>

      import('./customer-insurance/customer-insurance.component').then(

        m => m.CustomerInsuranceComponent

      ),

  },

  //For slecting insurance after choosing package

  {

    path: 'insurance-selection',

    loadComponent: () =>

      import('./insurance-selection/insurance-selection.component').then(

        m => m.InsuranceSelectionComponent

      ),

  },

];

//src\app\app.config.ts

import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';

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

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

import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {

  providers: [provideZoneChangeDetection({ eventCoalescing: true }), provideRouter(routes),provideHttpClient() ]

};

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.