Landing page cs

using CommunityToolkit.Maui.Extensions;

using CommunityToolkit.Maui.Views;

using Microsoft.Maui.Storage;

using road\_rescue.Driver\_UI;

namespace road\_rescue;

public partial class landingpage : ContentPage

{

public landingpage()

{

InitializeComponent();

}

private async void OnOtherHelpServicesTapped(object sender, TappedEventArgs e)

{

var popup = new OtherServicesPopup();

await this.ShowPopupAsync(popup);

}

private async void OnVulcanizingShopsTapped(object sender, TappedEventArgs e)

{

try

{

// 1. Initialize Supabase (if not already done)

await SupabaseService.InitializeAsync();

// 2. Get location

var location = await Geolocation.GetLocationAsync();

// 3. Navigate using the EXISTING client from SupabaseService

await Navigation.PushAsync(new Vulcanizing(location, SupabaseService.Client));

}

catch (Exception ex)

{

await DisplayAlert("Error", ex.Message, "OK");

}

}

private void ProfileButton\_Clicked(object sender, EventArgs e)

{

Shell.Current.FlyoutIsPresented = true;

}

private async void sosbutton\_Clicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new EmergencyRequestPage());

}

private async void ResetOnboarding\_Clicked(object sender, EventArgs e)

{

Preferences.Set("IsFirstLaunch", true);

await Shell.Current.GoToAsync("mainpage");

}

private void TapGestureRecognizer\_Tapped(object sender, TappedEventArgs e)

{

Shell.Current.FlyoutIsPresented = true;

}

private async void OnRepairShopTapped(object sender, TappedEventArgs e)

{

try

{

// 1. Initialize Supabase (if not already done)

await SupabaseService.InitializeAsync();

// 2. Get location

var location = await Geolocation.GetLocationAsync();

// 3. Navigate using the EXISTING client from SupabaseService

await Navigation.PushAsync(new repairShop(location, SupabaseService.Client));

}

catch (Exception ex)

{

await DisplayAlert("Error", ex.Message, "OK");

}

}

private async void OnGasStationTapped(object sender, TappedEventArgs e)

{

try

{

// 1. Initialize Supabase (if not already done)

await SupabaseService.InitializeAsync();

// 2. Get location

var location = await Geolocation.GetLocationAsync();

// 3. Navigate using the EXISTING client from SupabaseService

await Navigation.PushAsync(new gasStation(location, SupabaseService.Client));

}

catch (Exception ex)

{

await DisplayAlert("Error", ex.Message, "OK");

}

}

}

Repairshop xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="road\_rescue.Driver\_UI.repairShop"

Shell.NavBarIsVisible="False"

xmlns:local="clr-namespace:road\_rescue.Driver\_UI"

>

<ContentPage.Resources>

<local:RatingToStarsConverter x:Key="RatingToStars" />

</ContentPage.Resources>

<Grid RowDefinitions="Auto,Auto,\*">

<!-- Fixed Header (Search Bar) -->

<VerticalStackLayout Padding="10" Spacing="10" BackgroundColor="White" Grid.Row="0">

<Label Text="Nearest Repair Shop"

FontSize="24"

FontAttributes="Bold"

HorizontalOptions="Center"

Margin="0,10" />

<Border Stroke="Gray" StrokeThickness="1" Padding="0"

StrokeShape="RoundRectangle 100">

<SearchBar Placeholder="Find services near you.."

BackgroundColor="Transparent"

VerticalOptions="Center"

HeightRequest="40"/>

</Border>

</VerticalStackLayout>

<!-- Horizontal Scrollable Filter Buttons (Positioned below the search bar) -->

<ScrollView Orientation="Horizontal" Grid.Row="1" Margin="0,10">

<HorizontalStackLayout Spacing="6" Padding="5,0">

<Button Text="🛠 Repair"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🛞 Vulcanize" Clicked="Button\_Clicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="⛽ Gas"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚑 Hospital"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚨 Rescue"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="👮 Police"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🔥 Fire"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

</HorizontalStackLayout>

</ScrollView>

<!-- Scrollable Shop List (below buttons) -->

<ScrollView Grid.Row="2">

<VerticalStackLayout Padding="10" Spacing="10">

<CollectionView ItemsSource="{Binding Places}">

<CollectionView.ItemTemplate>

<DataTemplate>

<Border Margin="10" Padding="15" StrokeThickness="1" Stroke="#E0E0E0" BackgroundColor="#ffffff">

<Grid RowDefinitions="Auto,\*,Auto" ColumnDefinitions="Auto,\*">

<Image Grid.RowSpan="2"

Source="derek.jpg"

WidthRequest="60"

HeightRequest="60"

Aspect="AspectFill"

VerticalOptions="Center"

HorizontalOptions="Start"

Margin="0,0,10,0">

<Image.Clip>

<EllipseGeometry Center="30,30" RadiusX="30" RadiusY="30"/>

</Image.Clip>

</Image>

<Label Grid.Column="1"

Text="{Binding name}"

FontSize="15"

VerticalOptions="Start"

TextColor="Black"

FontAttributes="Bold"

LineBreakMode="WordWrap"

Margin="0,0,0,5"/>

<StackLayout Grid.Row="2" Grid.Column="0"

Orientation="Vertical"

VerticalOptions="End"

HorizontalOptions="Start"

Spacing="4">

<Label Text="{Binding address}"

FontSize="14"

TextColor="Gray"

VerticalOptions="Center" />

<Label Text="{Binding plus\_code}"

FontSize="14"

TextColor="Gray"

VerticalOptions="Center" />

<StackLayout Orientation="Horizontal" Spacing="5">

<Label Text="{Binding rating, Converter={StaticResource RatingToStars}}"

FontSize="14"

TextColor="Goldenrod"

VerticalOptions="Center" />

<Label Text="{Binding rating, StringFormat='{0:F1}'}"

FontSize="14"

TextColor="Black"

VerticalOptions="Center"/>

</StackLayout>

</StackLayout>

<VerticalStackLayout Grid.Row="2" Grid.Column="1" HorizontalOptions="End" Spacing="10" Margin="0,15,0,0" WidthRequest="100">

<Button Text="Location" Command="{Binding Source={RelativeSource AncestorType={x:Type local:repairShop}},

Path=OpenMapsCommand}"

CommandParameter="{Binding maps\_link}"

BackgroundColor="#21499F" TextColor="White"

CornerRadius="20" HeightRequest="35" WidthRequest="100" Padding="0"/>

<Button Text="Message" Clicked="OnMessageClicked"

BackgroundColor="#34A853" TextColor="White"

CornerRadius="20" HeightRequest="35" WidthRequest="100" Padding="0"/>

</VerticalStackLayout>

</Grid>

</Border>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</VerticalStackLayout>

</ScrollView>

</Grid>

</ContentPage>

Repairshop cs

using Microsoft.Maui.Controls;

using Supabase.Postgrest.Attributes;

using System.Collections.ObjectModel;

using System.Globalization;

using System.Linq;

using System.Windows.Input;

namespace road\_rescue.Driver\_UI

{

public partial class repairShop : ContentPage

{

private readonly Supabase.Client \_supabase;

public Location CurrentLocation { get; set; }

public ObservableCollection<PlaceModel> Places { get; set; } = new ObservableCollection<PlaceModel>();

// Command for opening maps

public ICommand OpenMapsCommand { get; }

public repairShop(Location location, Supabase.Client supabase)

{

try

{

InitializeComponent();

BindingContext = this;

CurrentLocation = location;

\_supabase = supabase;

OpenMapsCommand = new Command<string>(async (url) =>

{

if (!string.IsNullOrEmpty(url))

{

try

{

await Launcher.OpenAsync(url);

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Could not open maps: {ex.Message}", "OK");

}

}

});

LoadRepairShops();

}

catch (Exception ex)

{

Console.WriteLine($"Error initializing repairShop: {ex}");

DisplayAlert("Error", "Failed to load repair shops", "OK");

}

}

private async void LoadRepairShops()

{

try

{

Places.Clear();

var placesResponse = await \_supabase

.From<PlaceModel>()

.Select("\*")

.Where(x => x.category == "repair\_shop")

.Get();

var ratingsResponse = await \_supabase

.From<RatingModel>()

.Select("\*")

.Get();

var ratingsLookup = ratingsResponse.Models

.GroupBy(r => r.place\_id)

.ToDictionary(

g => g.Key,

g => g.Average(r => r.stars)

);

var sortedPlaces = placesResponse.Models

.Select(p => new

{

Place = p,

DistanceKm = CalculateDistance(

CurrentLocation.Latitude,

CurrentLocation.Longitude,

p.latitude,

p.longitude),

Rating = ratingsLookup.TryGetValue(p.place\_id, out var avg)

? avg

: 0

})

.OrderBy(x => x.DistanceKm)

.ToList();

foreach (var item in sortedPlaces)

{

item.Place.Distance = $"{item.DistanceKm:0.1} km away";

item.Place.rating = item.Rating;

Places.Add(item.Place);

}

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Failed to load data: {ex.Message}", "OK");

}

}

private double CalculateDistance(double lat1, double lon1, double lat2, double lon2)

{

const double R = 6371;

var dLat = ToRadians(lat2 - lat1);

var dLon = ToRadians(lon2 - lon1);

lat1 = ToRadians(lat1);

lat2 = ToRadians(lat2);

var a = Math.Sin(dLat / 2) \* Math.Sin(dLat / 2) +

Math.Sin(dLon / 2) \* Math.Sin(dLon / 2) \* Math.Cos(lat1) \* Math.Cos(lat2);

var c = 2 \* Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

return R \* c;

}

private double ToRadians(double angle) => Math.PI \* angle / 180.0;

private void OnBackClicked(object sender, EventArgs e) => Navigation.PopAsync();

private async void OnMessageClicked(object sender, EventArgs e)

=> await Navigation.PushAsync(new ChatPage());

private void OnLocationClicked(object sender, EventArgs e)

{

}

//private void OnBackClicked(object sender, EventArgs e)

//{

// Navigation.PopAsync();

//}

private void OnViewMapClicked(object sender, EventArgs e)

{

}

private void OnCallClicked(object sender, EventArgs e)

{

}

private void OnFavoriteClicked(object sender, EventArgs e)

{

}

private async void OnMoreInfoClicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new MapPage());

}

//private async void OnMessageClicked(object sender, EventArgs e)

//{

// await Navigation.PushAsync(new ChatPage());

//}

private void Button\_Clicked(object sender, EventArgs e)

{

}

}

public class RatingToStarsConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is double rating)

{

int fullStars = (int)Math.Floor(rating);

bool halfStar = (rating - fullStars >= 0.5);

string stars = new string('⭐', fullStars);

if (halfStar) stars += "⭐";

return stars.PadRight(5, '☆');

}

return "☆☆☆☆☆";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

=> throw new NotImplementedException();

}

[Table("places")]

public class PlaceModel : Supabase.Postgrest.Models.BaseModel

{

[PrimaryKey("place\_id")]

public string place\_id { get; set; }

public string name { get; set; }

public string category { get; set; }

public string address { get; set; }

public string plus\_code { get; set; }

public double latitude { get; set; }

public double longitude { get; set; }

public string maps\_link { get; set; }

public string Distance { get; set; }

public double rating { get; set; }

public override bool Equals(object obj) => base.Equals(obj);

public override int GetHashCode() => base.GetHashCode();

}

[Table("ratings")]

public class RatingModel : Supabase.Postgrest.Models.BaseModel

{

[PrimaryKey("rating\_id")]

public string rating\_id { get; set; }

public string driver\_id { get; set; }

public string place\_id { get; set; }

public double stars { get; set; }

public string comment { get; set; }

}

}

Gas statios xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="road\_rescue.Driver\_UI.gasStation"

Title="Nearest Gas Stations"

Shell.TabBarIsVisible="False"

xmlns:driver="clr-namespace:road\_rescue.Driver\_UI"

xmlns:local="clr-namespace:road\_rescue">

<ContentPage.Resources>

<local:RatingToStarsConverter x:Key="RatingToStars" />

</ContentPage.Resources>

<Grid RowDefinitions="Auto,Auto,\*">

<!-- Header with Search -->

<VerticalStackLayout Padding="10" Spacing="10" BackgroundColor="White" Grid.Row="0">

<Label Text="Nearest Gas Stations"

FontSize="24"

FontAttributes="Bold"

HorizontalOptions="Center"

Margin="0,10" />

<Border Stroke="Gray" StrokeThickness="1" Padding="0"

StrokeShape="RoundRectangle 100">

<SearchBar Placeholder="Find gas stations near you.."

BackgroundColor="Transparent"

VerticalOptions="Center"

HeightRequest="40"/>

</Border>

</VerticalStackLayout>

<!-- Category Chips -->

<ScrollView Orientation="Horizontal" Grid.Row="1" Margin="0,10">

<HorizontalStackLayout Spacing="6" Padding="5,0">

<Button Text="🛠 Repair" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🛞 Vulcanize" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="⛽ Gas" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚑 Hospital" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚨 Rescue" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="👮 Police" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🔥 Fire" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

</HorizontalStackLayout>

</ScrollView>

<!-- Gas Stations List -->

<ScrollView Grid.Row="2">

<VerticalStackLayout Padding="10" Spacing="10">

<CollectionView ItemsSource="{Binding Places}">

<CollectionView.ItemTemplate>

<DataTemplate>

<Border Margin="10" Padding="15" StrokeThickness="1" Stroke="#E0E0E0" BackgroundColor="#ffffff">

<Grid RowDefinitions="Auto,\*,Auto" ColumnDefinitions="Auto,\*">

<Image Grid.RowSpan="2"

Source="gas\_station.png"

WidthRequest="60"

HeightRequest="60"

Aspect="AspectFill"

VerticalOptions="Center"

HorizontalOptions="Start"

Margin="0,0,10,0">

<Image.Clip>

<EllipseGeometry Center="30,30" RadiusX="30" RadiusY="30"/>

</Image.Clip>

</Image>

<Label Grid.Column="1"

Text="{Binding name}"

FontSize="18"

VerticalOptions="Start"

TextColor="Black"

FontAttributes="Bold"

LineBreakMode="WordWrap"

Margin="0,0,0,5"/>

<StackLayout Grid.Row="2" Grid.Column="0"

Orientation="Vertical"

VerticalOptions="End"

HorizontalOptions="Start"

Spacing="2">

<Label Text="{Binding plus\_code}"

FontSize="14"

TextColor="Gray"

VerticalOptions="Center"

MaximumWidthRequest="200"/>

</StackLayout>

<VerticalStackLayout Grid.Row="2" Grid.Column="1" HorizontalOptions="End" Spacing="10" Margin="0,15,0,0" WidthRequest="100">

<Button Text="Locate" Command="{Binding Source={RelativeSource AncestorType={x:Type driver:gasStation}}, Path=OpenMapsCommand}"

CommandParameter="{Binding maps\_link}"

BackgroundColor="#21499F" TextColor="White"

CornerRadius="20" HeightRequest="35" WidthRequest="100" Padding="0"/>

</VerticalStackLayout>

</Grid>

</Border>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</VerticalStackLayout>

</ScrollView>

</Grid>

</ContentPage>

Gas station cs

using Microsoft.Maui.Controls;

using road\_rescue.Driver\_UI;

using Supabase.Postgrest.Attributes;

using System.Collections.ObjectModel;

using System.Globalization;

using System.Linq;

using System.Windows.Input;

namespace road\_rescue.Driver\_UI

{

public partial class gasStation : ContentPage

{

private readonly Supabase.Client \_supabase;

public Location CurrentLocation { get; set; }

public ObservableCollection<PlaceModel> Places { get; set; } = new ObservableCollection<PlaceModel>();

// Command for opening maps

public ICommand OpenMapsCommand { get; }

public gasStation(Location location, Supabase.Client supabase)

{

try

{

InitializeComponent();

BindingContext = this;

CurrentLocation = location;

\_supabase = supabase;

OpenMapsCommand = new Command<string>(async (url) =>

{

if (!string.IsNullOrEmpty(url))

{

try

{

await Launcher.OpenAsync(url);

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Could not open maps: {ex.Message}", "OK");

}

}

});

LoadGasStations();

}

catch (Exception ex)

{

Console.WriteLine($"Error initializing GasStation: {ex}");

DisplayAlert("Error", "Failed to load gas stations", "OK");

}

}

private async void LoadGasStations()

{

try

{

Places.Clear();

// 1. Fetch all gas stations

var placesResponse = await \_supabase

.From<PlaceModel>()

.Select("\*")

.Where(x => x.category == "gas\_station")

.Get();

// 2. Fetch all ratings (single query is more efficient)

var ratingsResponse = await \_supabase

.From<RatingModel>()

.Select("\*")

.Get();

var ratingsLookup = ratingsResponse.Models

.GroupBy(r => r.place\_id)

.ToDictionary(

g => g.Key,

g => g.Average(r => r.stars)

);

// 3. Calculate distance and assign ratings

var sortedPlaces = placesResponse.Models

.Select(p => new

{

Place = p,

DistanceKm = CalculateDistance(

CurrentLocation.Latitude,

CurrentLocation.Longitude,

p.latitude,

p.longitude),

Rating = ratingsLookup.TryGetValue(p.place\_id, out var avg)

? avg

: 0 // Default to 0 if no ratings

})

.OrderBy(x => x.DistanceKm)

.ToList();

foreach (var item in sortedPlaces)

{

item.Place.Distance = $"{item.DistanceKm:0.1} km away";

item.Place.rating = item.Rating; // Assign the calculated average

Places.Add(item.Place);

}

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Failed to load data: {ex.Message}", "OK");

}

}

private double CalculateDistance(double lat1, double lon1, double lat2, double lon2)

{

const double R = 6371;

var dLat = ToRadians(lat2 - lat1);

var dLon = ToRadians(lon2 - lon1);

lat1 = ToRadians(lat1);

lat2 = ToRadians(lat2);

var a = Math.Sin(dLat / 2) \* Math.Sin(dLat / 2) +

Math.Sin(dLon / 2) \* Math.Sin(dLon / 2) \* Math.Cos(lat1) \* Math.Cos(lat2);

var c = 2 \* Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

return R \* c;

}

private double ToRadians(double angle) => Math.PI \* angle / 180.0;

private void OnBackClicked(object sender, EventArgs e) => Navigation.PopAsync();

private async void OnMessageClicked(object sender, EventArgs e)

=> await Navigation.PushAsync(new ChatPage());

private void OnLocationClicked(object sender, EventArgs e)

{

// Implement location functionality if needed

}

private void OnViewMapClicked(object sender, EventArgs e)

{

// Implement view map functionality if needed

}

private void OnCallClicked(object sender, EventArgs e)

{

// Implement call functionality if needed

}

private void OnFavoriteClicked(object sender, EventArgs e)

{

// Implement favorite functionality if needed

}

private async void OnMoreInfoClicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new MapPage());

}

private void OnChipClicked(object sender, EventArgs e)

{

// Implement chip functionality if needed

}

}

}

Vulcanizing xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="road\_rescue.Vulcanizing"

Shell.NavBarIsVisible="False"

xmlns:local="clr-namespace:road\_rescue"

>

<ContentPage.Resources>

<local:RatingToStarsConverter x:Key="RatingToStars" />

</ContentPage.Resources>

<Grid RowDefinitions="Auto,Auto,\*">

<!-- Fixed Header (Search Bar) -->

<VerticalStackLayout Padding="10" Spacing="10" BackgroundColor="White" Grid.Row="0">

<Label Text="Nearest Vulcanizing Shop"

FontSize="24"

FontAttributes="Bold"

HorizontalOptions="Center"

Margin="0,10" />

<Border Stroke="Gray" StrokeThickness="1" Padding="0"

StrokeShape="RoundRectangle 100">

<SearchBar Placeholder="Find services near you.."

BackgroundColor="Transparent"

VerticalOptions="Center"

HeightRequest="40"/>

</Border>

</VerticalStackLayout>

<!-- Horizontal Scrollable Filter Buttons (Positioned below the search bar) -->

<ScrollView Orientation="Horizontal" Grid.Row="1" Margin="0,10">

<HorizontalStackLayout Spacing="6" Padding="5,0">

<Button Text="🛠 Repair" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🛞 Vulcanize" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="⛽ Gas" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚑 Hospital" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🚨 Rescue" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="👮 Police" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

<Button Text="🔥 Fire" Clicked="OnChipClicked"

FontSize="12" Padding="10,5"

BackgroundColor="#f2f2f2" TextColor="Black"

BorderColor="Gray" BorderWidth="1"

CornerRadius="15" />

</HorizontalStackLayout>

</ScrollView>

<!-- Scrollable Shop List (below buttons) -->

<ScrollView Grid.Row="2">

<VerticalStackLayout Padding="10" Spacing="10">

<CollectionView ItemsSource="{Binding Places}">

<CollectionView.ItemTemplate>

<DataTemplate>

<Border Margin="10" Padding="15" StrokeThickness="1" Stroke="#E0E0E0" BackgroundColor="#ffffff">

<Grid RowDefinitions="Auto,\*,Auto" ColumnDefinitions="Auto,\*">

<Image Grid.RowSpan="2"

Source="derek.jpg"

WidthRequest="60"

HeightRequest="60"

Aspect="AspectFill"

VerticalOptions="Center"

HorizontalOptions="Start"

Margin="0,0,10,0">

<Image.Clip>

<EllipseGeometry Center="30,30" RadiusX="30" RadiusY="30"/>

</Image.Clip>

</Image>

<Label Grid.Column="1"

Text="{Binding name}"

FontSize="15"

VerticalOptions="Start"

TextColor="Black"

FontAttributes="Bold"

LineBreakMode="WordWrap"

Margin="0,0,0,5"/>

<StackLayout Grid.Row="2" Grid.Column="0"

Orientation="Vertical"

VerticalOptions="End"

HorizontalOptions="Start"

MaximumWidthRequest="200"

Spacing="4">

<Label Text="{Binding address}"

FontSize="14"

TextColor="Gray"

VerticalOptions="Center" />

<Label Text="{Binding plus\_code}"

FontSize="14"

TextColor="Gray"

VerticalOptions="Center" />

<StackLayout Orientation="Horizontal" Spacing="5">

<Label Text="{Binding rating, Converter={StaticResource RatingToStars}}"

FontSize="14"

TextColor="Goldenrod"

VerticalOptions="Center" />

<Label Text="{Binding rating, StringFormat='{0:F1}'}"

FontSize="14"

TextColor="Black"

VerticalOptions="Center"/>

</StackLayout>

</StackLayout>

<VerticalStackLayout Grid.Row="2" Grid.Column="1" HorizontalOptions="End" Spacing="10" Margin="0,15,0,0" WidthRequest="100">

<Button Text="Location" Command="{Binding Source={RelativeSource AncestorType={x:Type local:Vulcanizing}},

Path=OpenMapsCommand}"

CommandParameter="{Binding maps\_link}"

BackgroundColor="#21499F" TextColor="White"

CornerRadius="20" HeightRequest="35" WidthRequest="100" Padding="0"/>

<Button Text="Message" Clicked="OnMessageClicked"

BackgroundColor="#34A853" TextColor="White"

CornerRadius="20" HeightRequest="35" WidthRequest="100" Padding="0"/>

</VerticalStackLayout>

</Grid>

</Border>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</VerticalStackLayout>

</ScrollView>

</Grid>

</ContentPage>

using Microsoft.Maui.Controls;

using road\_rescue.Driver\_UI;

using Supabase.Postgrest.Attributes;

using System.Collections.ObjectModel;

using System.Globalization;

using System.Linq;

using System.Windows.Input;

namespace road\_rescue

{

public partial class Vulcanizing : ContentPage

{

private readonly Supabase.Client \_supabase;

public Location CurrentLocation { get; set; }

public ObservableCollection<PlaceModel> Places { get; set; } = new ObservableCollection<PlaceModel>();

// Command for opening maps

public ICommand OpenMapsCommand { get; }

public Vulcanizing(Location location, Supabase.Client supabase)

{

try

{

InitializeComponent();

BindingContext = this;

CurrentLocation = location;

\_supabase = supabase;

OpenMapsCommand = new Command<string>(async (url) =>

{

if (!string.IsNullOrEmpty(url))

{

try

{

await Launcher.OpenAsync(url);

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Could not open maps: {ex.Message}", "OK");

}

}

});

LoadVulcanizingShops();

}

catch (Exception ex)

{

Console.WriteLine($"Error initializing Vulcanizing: {ex}");

DisplayAlert("Error", "Failed to load vulcanizing shops", "OK");

}

}

private async void LoadVulcanizingShops()

{

try

{

Places.Clear();

// 1. Fetch all vulcanizing shops

var placesResponse = await \_supabase

.From<PlaceModel>()

.Select("\*")

.Where(x => x.category == "vulcanizing")

.Get();

// 2. Fetch all ratings

var ratingsResponse = await \_supabase

.From<RatingModel>()

.Select("\*")

.Get();

var ratingsLookup = ratingsResponse.Models

.GroupBy(r => r.place\_id)

.ToDictionary(

g => g.Key,

g => g.Average(r => r.stars)

);

// 3. Calculate distance and assign ratings

var sortedPlaces = placesResponse.Models

.Select(p => new

{

Place = p,

DistanceKm = CalculateDistance(

CurrentLocation.Latitude,

CurrentLocation.Longitude,

p.latitude,

p.longitude),

Rating = ratingsLookup.TryGetValue(p.place\_id, out var avg)

? avg

: 0 // Default to 0 if no ratings

})

.OrderBy(x => x.DistanceKm)

.ToList();

foreach (var item in sortedPlaces)

{

item.Place.Distance = $"{item.DistanceKm:0.1} km away";

item.Place.rating = item.Rating; // Assign the calculated average

Places.Add(item.Place);

}

}

catch (Exception ex)

{

await DisplayAlert("Error", $"Failed to load data: {ex.Message}", "OK");

}

}

private double CalculateDistance(double lat1, double lon1, double lat2, double lon2)

{

const double R = 6371;

var dLat = ToRadians(lat2 - lat1);

var dLon = ToRadians(lon2 - lon1);

lat1 = ToRadians(lat1);

lat2 = ToRadians(lat2);

var a = Math.Sin(dLat / 2) \* Math.Sin(dLat / 2) +

Math.Sin(dLon / 2) \* Math.Sin(dLon / 2) \* Math.Cos(lat1) \* Math.Cos(lat2);

var c = 2 \* Math.Atan2(Math.Sqrt(a), Math.Sqrt(1 - a));

return R \* c;

}

private double ToRadians(double angle) => Math.PI \* angle / 180.0;

private void OnBackClicked(object sender, EventArgs e) => Navigation.PopAsync();

private async void OnMessageClicked(object sender, EventArgs e)

=> await Navigation.PushAsync(new ChatPage());

private void OnLocationClicked(object sender, EventArgs e)

{

// Implement location functionality if needed

}

private void OnViewMapClicked(object sender, EventArgs e)

{

// Implement view map functionality if needed

}

private void OnCallClicked(object sender, EventArgs e)

{

// Implement call functionality if needed

}

private void OnFavoriteClicked(object sender, EventArgs e)

{

// Implement favorite functionality if needed

}

private async void OnMoreInfoClicked(object sender, EventArgs e)

{

await Navigation.PushAsync(new MapPage());

}

private void OnChipClicked(object sender, EventArgs e)

{

// Implement chip functionality if needed

}

}

public class RatingToStarsConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is double rating)

{

int fullStars = (int)Math.Floor(rating);

bool halfStar = (rating - fullStars >= 0.5);

string stars = new string('⭐', fullStars);

if (halfStar) stars += "⭐";

return stars.PadRight(5, '☆');

}

return "☆☆☆☆☆";

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

=> throw new NotImplementedException();

}

[Table("places")]

public class PlaceModel : Supabase.Postgrest.Models.BaseModel

{

[PrimaryKey("place\_id")]

public string place\_id { get; set; }

public string name { get; set; }

public string category { get; set; }

public string address { get; set; }

public string plus\_code { get; set; }

public double latitude { get; set; }

public double longitude { get; set; }

public string maps\_link { get; set; }

public string Distance { get; set; }

public double rating { get; set; }

public override bool Equals(object obj) => base.Equals(obj);

public override int GetHashCode() => base.GetHashCode();

}

}