**Designing the Models for Food Delivery App and Creating the Database**

**Project Setup:**

First, create a new ASP.NET Core MVC project with the name **FoodDeliveryApp** and add the following packages. Please execute the following commands in the Visual Studio Package Manager console to install the EF Core, Password Hashing, and Serilog Logging Packages.

**Install-Package Microsoft.EntityFrameworkCore.SqlServer**

**Install-Package Microsoft.EntityFrameworkCore.Tools**

**Install-Package Serilog.AspNetCore**

**Install-Package Serilog.Sinks.File**

**Install-Package Serilog.Settings.Configuration**

**Install-Package Serilog.Sinks.Async**

**Install-Package BCrypt.Net-Next**

**Creating Models:**

Models define the shape of the data that the application deals with. For example, in a food delivery application, we might have models like User, Order, Restaurant, and MenuItem. Models represent the domain entities and map to database tables (using Entity Framework Core). So, please create a folder named Models in the project root directory where we will create all our Models.

**RoleMaster Model:**

Create a class file named **RoleMaster.cs** within the **Models** folder, and then copy and paste the following code. This Model defines various roles (e.g., Customer, DeliveryPartner, RestaurantOwner, Admin, and SuperAdmin) that are essential for user role management. This helps in assigning appropriate permissions and functionalities to different user types. The navigation property lets you easily query all users belonging to a particular role.

**using Microsoft.EntityFrameworkCore;**

**namespace FoodDeliveryApp.Models**

**{**

**[Index(nameof(RoleName), IsUnique =true, Name = "IX\_RoleName\_Unique")]**

**public class RoleMaster**

**{**

**public int RoleMasterId { get; set; }**

**public string RoleName { get; set; } // e.g., "Customer", "DeliveryPartner", "RestaurantOwner", "Admin"**

**public string? Description { get; set; } // optional descriptive text**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<User> Users { get; set; }**

**}**

**}**

**User Model:**

Create a class file named **User.cs** within the **Models** folder, and then copy and paste the following code. The User Model represents application users, storing their credentials, personal details, and role associations. It is central to user authentication and authorization, as well as to maintaining a user’s order history and related data.

**using Microsoft.EntityFrameworkCore;**

**namespace FoodDeliveryApp.Models**

**{**

**[Index(nameof(Email), IsUnique =true)]**

**[Index(nameof(PhoneNumber), IsUnique = true)]**

**public class User**

**{**

**public int UserId { get; set; }**

**// Basic credentials**

**public string FirstName { get; set; }**

**public string LastName { get; set; }**

**public string PhoneNumber { get; set; }**

**public string Email { get; set; }**

**public string PasswordHash { get; set; }**

**//Role Information**

**public int RoleMasterId { get; set; }**

**public virtual RoleMaster RoleMaster { get; set; }**

**// Audit Information**

**public DateTime CreatedDate { get; set; }**

**public DateTime ModifiedDate { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<UserAddress>? UserAddresses { get; set; }**

**public virtual ICollection<Order>? Orders { get; set; } // For Customers**

**// Optional one-to-one relationships to specialized profiles**

**public virtual DeliveryPartnerProfile? DeliveryPartnerProfile { get; set; }**

**public virtual RestaurantOwnerProfile? RestaurantOwnerProfile { get; set; }**

**}**

**}**

**RestaurantOwnerProfile Model:**

Create a class file named **RestaurantOwnerProfile.cs** within the **Models** folder, and then copy and paste the following code. The RestaurantOwnerProfile model captures specific information about restaurant owners, such as business registration numbers and verification status. This allows the system to distinguish restaurant owners from other users and manage their restaurant-related details effectively.

**namespace FoodDeliveryApp.Models**

**{**

**public class RestaurantOwnerProfile**

**{**

**public int RestaurantOwnerProfileId { get; set; }**

**// 1:1 relationship to User**

**public int UserId { get; set; }**

**public virtual User User { get; set; }**

**// Role-specific fields for a restaurant owner**

**public string BusinessLicenseNumber { get; set; }**

**public string GSTIN { get; set; } // if relevant in your region**

**public string BusinessRegistrationNumber { get; set; }**

**public bool IsVerified { get; set; } // whether Admin verified the owner’s documents**

**// Bank details, payout info, etc.**

**public string AccountHolderName { get; set; }**

**public string BankName { get; set; }**

**public string AccountNumber { get; set; }**

**public string IFSCCode { get; set; }**

**}**

**}**

**DeliveryPartnerProfile Model:**

Create a class file named **DeliveryPartnerProfile.cs** within the **Models** folder, and then copy and paste the following code. The DeliveryPartnerProfile model holds details specific to delivery personnel, including vehicle information, ratings, and activity status. It helps track the performance and availability of delivery partners.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class DeliveryPartnerProfile**

**{**

**public int DeliveryPartnerProfileId { get; set; }**

**// 1:1 relationship to User**

**public int UserId { get; set; }**

**public virtual User User { get; set; }**

**// Role-specific fields**

**public string LicenseNumber { get; set; }**

**public string VehicleType { get; set; } // e.g., Bike, Car**

**public string VehicleRegistrationNumber { get; set; }**

**// Navigation for deliveries**

**public virtual ICollection<Delivery> Deliveries { get; set; }**

**// This will dynamically count the number of completed deliveries without storing it.**

**[NotMapped]**

**public int TotalDeliveries => Deliveries?.Count ?? 0;**

**// Navigation for reviews given to this delivery partner**

**public virtual ICollection<Review> Reviews { get; set; }**

**[NotMapped]**

**public decimal AverageRating => Reviews?.Any() ?? false ? (decimal)Reviews.Average(r => r.Rating) : 0;**

**}**

**}**

**UserAddress Model:**

Create a class file named **UserAddress.cs** within the **Models** folder, and then copy and paste the following code. The UserAddress model manages multiple addresses for users, allowing them to save home, work, or other frequently used delivery locations. This improves convenience and user experience during the ordering process.

**namespace FoodDeliveryApp.Models**

**{**

**public class UserAddress**

**{**

**public int UserAddressId { get; set; }**

**public int UserId { get; set; } //FK**

**public virtual User User { get; set; }**

**public string Label { get; set; } // e.g., "Home", "Work"**

**public string AddressLine1 { get; set; }**

**public string? AddressLine2 { get; set; }**

**public string City { get; set; }**

**public string State { get; set; }**

**public string ZipCode { get; set; }**

**public string? Landmark { get; set; }**

**public string? Latitude { get; set; } // optional for geo**

**public string? Longitude { get; set; } // optional for geo**

**}**

**}**

**Restaurant Model:**

Create a class file named **Restaurant.cs** within the **Models** folder, and then copy and paste the following code. The Restaurant model stores core information about restaurants, including contact details, approval status, and ratings. It is a key entity for listing and managing restaurants in the application.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class Restaurant**

**{**

**public int RestaurantId { get; set; }**

**// Foreign key to the "owner" if needed**

**public int OwnerId { get; set; }**

**public virtual User Owner { get; set; }**

**public string Name { get; set; }**

**public string? Description { get; set; }**

**public string PhoneNumber { get; set; }**

**public bool IsApproved { get; set; } // Admin approval for listing**

**// Possibly store operational timings**

**public TimeSpan OpeningTime { get; set; }**

**public TimeSpan ClosingTime { get; set; }**

**public string? LogoUrl { get; set; } // e.g., restaurant logo image**

**// Navigation**

**public virtual ICollection<RestaurantCuisine> RestaurantCuisines { get; set; } //Italian, Indian, Chinse**

**public virtual ICollection<MenuCategory> MenuCategories { get; set; } //Starters, Deserts, Main Courses**

**public virtual ICollection<Order> Orders { get; set; }**

**public virtual ICollection<RestaurantClosure> Closures { get; set; }**

**[NotMapped]**

**public bool IsOpen**

**{**

**get**

**{**

**var currentTime = DateTime.Now.TimeOfDay;**

**var today = DateTime.Today;**

**// Check if today falls within a closure period**

**bool isClosedToday = Closures?.Any(c => today >= c.StartDate.Date && today <= c.EndDate.Date) ?? false;**

**if (isClosedToday)**

**return false;**

**return currentTime >= OpeningTime && currentTime <= ClosingTime;**

**}**

**}**

**}**

**}**

**RestaurantAddress Model:**

Create a class file named **RestaurantAddress.cs** within the **Models** folder, and then copy and paste the following code. The RestaurantAddress model specifies the physical address details of each restaurant. It is used to link restaurants to their geographical location and supports functionalities such as distance-based delivery charges.

**namespace FoodDeliveryApp.Models**

**{**

**public class RestaurantAddress**

**{**

**public int RestaurantAddressId { get; set; }**

**public int RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**public string AddressLine1 { get; set; }**

**public string? AddressLine2 { get; set; }**

**public string City { get; set; }**

**public string State { get; set; }**

**public string ZipCode { get; set; }**

**public string? Landmark { get; set; }**

**public string? Latitude { get; set; }**

**public string? Longitude { get; set; }**

**}**

**}**

**RestaurantClosure Model**

Create a class file named **RestaurantClosure.cs** within the **Models** folder, and then copy and paste the following code. This model will hold information about when the restaurant is closed, the reasons, and for how many days. Once the restaurant is closed, it should not appear in the search results.

**namespace FoodDeliveryApp.Models**

**{**

**public class RestaurantClosure**

**{**

**public int RestaurantClosureId { get; set; }**

**public int RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**public DateTime StartDate { get; set; } // Start of closure**

**public DateTime EndDate { get; set; } // End of closure**

**public string? Reason { get; set; }**

**}**

**}**

**Cuisine Model:**

Create a class file named **Cuisine.cs** within the **Models** folder, and then copy and paste the following code. The Cuisine model represents different cuisine types (e.g., Indian, Italian, Chinese, etc.) available in the application. This helps users filter and search for restaurants or menu items based on their preferred cuisine.

**namespace FoodDeliveryApp.Models**

**{**

**//A master table for cuisines like Indian, Chinese, Italian, etc.**

**public class Cuisine**

**{**

**public int CuisineId { get; set; }**

**public string CuisineName { get; set; }**

**// For icon or thumbnail if needed**

**public string? ImageUrl { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<RestaurantCuisine> RestaurantCuisines { get; set; }**

**}**

**}**

**RestaurantCuisine Model:**

Create a class file named **RestaurantCuisine.cs** within the **Models** folder, and then copy and paste the following code. The RestaurantCuisine model links restaurants with the cuisines they offer. It is used for mapping many-to-many relationships between restaurants and cuisines, facilitating easier categorization and searching.

**namespace FoodDeliveryApp.Models**

**{**

**// Links a Restaurant with multiple Cuisine entries.**

**public class RestaurantCuisine**

**{**

**public int RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**public int CuisineId { get; set; }**

**public virtual Cuisine Cuisine { get; set; }**

**}**

**}**

**MenuCategory Model:**

Create a class file named **MenuCategory.cs** within the **Models** folder, and then copy and paste the following code. The MenuCategory model organizes menu items into logical groups, such as Starters, Main Course, or Desserts, making it easier for customers to browse and select items.

**namespace FoodDeliveryApp.Models**

**{**

**// Many restaurants group menu items under categories, e.g., Starters, Main Course, Desserts, etc.**

**public class MenuCategory**

**{**

**public int MenuCategoryId { get; set; }**

**public string CategoryName { get; set; } //Starter, Main Course**

**public string? Description { get; set; }**

**public int DisplayOrder { get; set; }**

**public bool IsActive { get; set; }**

**// Restaurant Relationship**

**public int RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**// Navigation**

**public virtual ICollection<MenuItem> MenuItems { get; set; }**

**}**

**}**

**MenuItem Model:**

Create a class file named **MenuItem.cs** within the **Models** folder, and then copy and paste the following code. The MenuItem model represents individual dishes or food items available for order. It includes details like price, availability, and ratings, which are critical for displaying options to customers and enabling order placement.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**// Represents each individual dish or item in a category**

**public class MenuItem**

**{**

**public int MenuItemId { get; set; }**

**// Category Relationship**

**public int MenuCategoryId { get; set; }**

**public virtual MenuCategory MenuCategory { get; set; }**

**public string ItemName { get; set; }**

**public string? Description { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal Price { get; set; }**

**// Example attributes: Veg/Non-Veg, Available/Unavailable**

**public bool IsVeg { get; set; }**

**public bool IsAvailable { get; set; }**

**// Could store an image URL**

**public string? ImageUrl { get; set; }**

**}**

**}**

**OrderStatusMaster Model:**

Create a class file named **OrderStatusMaster.cs** within the **Models** folder, and then copy and paste the following code. The OrderStatusMaster model provides predefined statuses for orders (e.g., Placed, Preparing, Delivered. etc.), enabling the system to track and display the progress of an order throughout its lifecycle.

**namespace FoodDeliveryApp.Models**

**{**

**public class OrderStatusMaster**

**{**

**public int OrderStatusMasterId { get; set; }**

**public string StatusName { get; set; } // "Placed", "Confirmed", "Preparing", "OutForDelivery", "Delivered", "Cancelled"**

**public string? Description { get; set; } // optional**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<Order> Orders { get; set; }**

**}**

**}**

**Order Model:**

Create a class file named **Order.cs** within the **Models** folder, and then copy and paste the following code. The Order model represents a customer’s food order, linking the customer, selected restaurant, chosen delivery address, and current status. It also tracks financial details like subtotal, tax, and total amounts.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class Order**

**{**

**public int OrderId { get; set; }**

**// Customer who placed the order**

**public int CustomerId { get; set; }**

**public virtual User Customer { get; set; }**

**// Relationship to Restaurant**

**public int RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**// Which address (UserAddress) the user selected for delivery**

**public int UserAddressId { get; set; }**

**public virtual UserAddress UserAddress { get; set; }**

**// Timestamps**

**public DateTime OrderDate { get; set; }**

**// Status**

**public int OrderStatusMasterId { get; set; }**

**public virtual OrderStatusMaster OrderStatusMaster { get; set; }**

**// Totals**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal SubTotal { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal DeliveryFee { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal TaxAmount { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal Discount { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal TotalAmount { get; set; }**

**// Additional instructions**

**public string? Notes { get; set; }**

**// Order Items**

**public virtual ICollection<OrderItem> OrderItems { get; set; }**

**// Delivery Info**

**public virtual Delivery Delivery { get; set; }**

**// Payment Info**

**public virtual Payment Payment { get; set; }**

**//Multiple Offers can be applied to a single order**

**public virtual ICollection<OrderOffer> OrderOffers { get; set; }**

**}**

**}**

**OrderItem Model:**

Create a class file named **OrderItem.cs** within the Models folder, and then copy and paste the following code. The OrderItem model breaks down an order into individual menu items, showing what was ordered, in what quantity, and at what price. This is crucial for billing, reporting, and reordering features.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class OrderItem**

**{**

**public int OrderItemId { get; set; }**

**public int OrderId { get; set; }**

**public virtual Order Order { get; set; }**

**public int MenuItemId { get; set; }**

**public virtual MenuItem MenuItem { get; set; }**

**public int Quantity { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal UnitPrice { get; set; } // price at time of ordering**

**public decimal TotalPrice => Quantity \* UnitPrice;**

**}**

**}**

**DeliveryStatusMaster Model:**

Create a class file named **DeliveryStatusMaster.cs** within the **Models** folder, and then copy and paste the following code. The DeliveryStatusMaster model defines various statuses for deliveries (e.g., Assigned, PickedUp, EnRoute, Delivered), allowing the system to monitor and display the progress of a delivery in real-time.

**namespace FoodDeliveryApp.Models**

**{**

**public class DeliveryStatusMaster**

**{**

**public int DeliveryStatusMasterId { get; set; }**

**public string StatusName { get; set; } // e.g., Assigned, PickedUp, EnRoute, Delivered**

**public string? Description { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<Delivery> Deliveries { get; set; }**

**}**

**}**

**Delivery Model:**

Create a class file named **Delivery.cs** within the **Models** folder, and then copy and paste the following code. The Delivery model tracks the delivery partner assigned to an order, along with times and current delivery status. It helps manage and optimize the logistics of order fulfilment.

**namespace FoodDeliveryApp.Models**

**{**

**// Tracks the status and assignment of a delivery partner to an order.**

**public class Delivery**

**{**

**public int DeliveryId { get; set; }**

**public int OrderId { get; set; }**

**public Order Order { get; set; }**

**// The user who is delivering (DeliveryPartner)**

**public int DeliveryPartnerProfileId { get; set; }**

**public virtual DeliveryPartnerProfile DeliveryPartnerProfile { get; set; }**

**public DateTime? PickupTime { get; set; }**

**public DateTime? DeliveryTime { get; set; }**

**public int DeliveryStatusMasterId { get; set; }**

**public virtual DeliveryStatusMaster DeliveryStatusMaster { get; set; }**

**}**

**}**

**PaymentTypeMaster Model:**

Create a class file named **PaymentTypeMaster.cs** within the **Models** folder, and then copy and paste the following code. The PaymentTypeMaster model lists accepted payment methods (e.g., CreditCard, UPI, CashOnDelivery), providing flexibility and choice for customers during checkout.

**namespace FoodDeliveryApp.Models**

**{**

**public class PaymentTypeMaster**

**{**

**public int PaymentTypeMasterId { get; set; }**

**public string TypeName { get; set; } // "CreditCard", "UPI", "CashOnDelivery"**

**public string? Description { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<Payment> Payments { get; set; }**

**}**

**}**

**PaymentStatusMaster Model:**

Create a class file named **PaymentStatusMaster.cs** within the **Models** folder, and then copy and paste the following code. The PaymentStatusMaster keeps track of payment outcomes (e.g., Pending, Completed, Failed), ensuring a clear and consistent payment process.

**namespace FoodDeliveryApp.Models**

**{**

**public class PaymentStatusMaster**

**{**

**public int PaymentStatusMasterId { get; set; }**

**public string StatusName { get; set; } // "Pending", "Completed", "Failed"**

**public string? Description { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<Payment> Payments { get; set; }**

**}**

**}**

**Payment Model:**

Create a class file named **Payment.cs** within the **Models** folder, and then copy and paste the following code. The Payment model captures transaction details, such as the amount, payment type, and status. This is vital for financial record-keeping and auditing.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class Payment**

**{**

**public int PaymentId { get; set; }**

**public int OrderId { get; set; }**

**public virtual Order Order { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal Amount { get; set; }**

**public int PaymentTypeMasterId { get; set; }**

**public virtual PaymentTypeMaster PaymentTypeMaster { get; set; }**

**public int PaymentStatusMasterId { get; set; }**

**public virtual PaymentStatusMaster PaymentStatusMaster { get; set; }**

**public DateTime PaymentDate { get; set; }**

**}**

**}**

**Review Model:**

Create a class file named **Review.cs** within the **Models** folder, and then copy and paste the following code. The Review model enables customers to leave feedback and ratings for restaurants or menu items. This improves transparency and helps other users make informed decisions.

**namespace FoodDeliveryApp.Models**

**{**

**public class Review**

**{**

**public int ReviewId { get; set; }**

**// The user who wrote the review**

**public int UserId { get; set; }**

**public virtual User User { get; set; }**

**// Possibly link to either the Restaurant, a MenuItem, or a Delivery Partner**

**public int? RestaurantId { get; set; }**

**public virtual Restaurant Restaurant { get; set; }**

**public int? MenuItemId { get; set; }**

**public virtual MenuItem MenuItem { get; set; }**

**public int? DeliveryPartnerProfileId { get; set; }**

**public virtual DeliveryPartnerProfile? DeliveryPartnerProfile { get; set; }**

**public int Rating { get; set; } // 1 to 5**

**public string? Comment { get; set; }**

**public DateTime ReviewDate { get; set; }**

**}**

**}**

**Offer Model:**

Create a class file named **Offer.cs** within the **Models** folder, and then copy and paste the following code. The Offer model stores promotional offers and discounts. This helps attract customers and incentivize purchases by providing cost savings and special deals.

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class Offer**

**{**

**public int OfferId { get; set; }**

**public string OfferCode { get; set; } // e.g., "FIRST50"**

**public string Description { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal DiscountAmount { get; set; }**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal DiscountPercentage { get; set; }**

**public bool IsPercentage { get; set; } // True if discount is in %, False for fixed amount**

**public DateTime StartDate { get; set; }**

**public DateTime EndDate { get; set; }**

**// Offer Scope**

**public bool IsGlobal { get; set; } // If true, applies to all restaurants/orders**

**// Nullable FK for Restaurant-Specific Offers**

**public int? RestaurantId { get; set; }**

**public virtual Restaurant? Restaurant { get; set; }**

**public bool IsActive { get; set; }**

**// Navigation**

**public virtual ICollection<OrderOffer> OrderOffers { get; set; }**

**}**

**}**

**OrderOffer Model:**

Create a class file named **OrderOffer.cs** within the **Models** folder, and then copy and paste the following code. The OrderOffer model Tracking Offers Applied to Orders

**using System.ComponentModel.DataAnnotations.Schema;**

**namespace FoodDeliveryApp.Models**

**{**

**public class OrderOffer**

**{**

**public int OrderOfferId { get; set; }**

**// Foreign Keys**

**public int OrderId { get; set; }**

**public virtual Order Order { get; set; }**

**public int OfferId { get; set; }**

**public virtual Offer Offer { get; set; }**

**// Actual discount applied (to store historical values)**

**[Column(TypeName = "decimal(8,2)")]**

**public decimal DiscountApplied { get; set; }**

**}**

**}**

**DB Context:**

Create a class file named **ApplicationDbContext.cs** within the **Data** folder, and then copy and paste the following code.

**using FoodDeliveryApp.Models;**

**using Microsoft.EntityFrameworkCore;**

**namespace FoodDeliveryApp.Data**

**{**

**public class ApplicationDbContext : DbContext**

**{**

**public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)**

**: base(options)**

**{**

**}**

**public DbSet<RoleMaster> RoleMasters { get; set; }**

**public DbSet<OrderStatusMaster> OrderStatusMasters { get; set; }**

**public DbSet<DeliveryStatusMaster> DeliveryStatusMasters { get; set; }**

**public DbSet<PaymentStatusMaster> PaymentStatusMasters { get; set; }**

**public DbSet<PaymentTypeMaster> PaymentTypeMasters { get; set; }**

**public DbSet<User> Users { get; set; }**

**public DbSet<DeliveryPartnerProfile> DeliveryPartnerProfiles { get; set; }**

**public DbSet<RestaurantOwnerProfile> RestaurantOwnerProfiles { get; set; }**

**public DbSet<UserAddress> UserAddresses { get; set; }**

**public DbSet<Restaurant> Restaurants { get; set; }**

**public DbSet<RestaurantAddress> RestaurantAddresses { get; set; }**

**public DbSet<RestaurantClosure> RestaurantClosures { get; set; }**

**public DbSet<Cuisine> Cuisines { get; set; }**

**public DbSet<RestaurantCuisine> RestaurantCuisines { get; set; }**

**public DbSet<MenuCategory> MenuCategories { get; set; }**

**public DbSet<MenuItem> MenuItems { get; set; }**

**public DbSet<Order> Orders { get; set; }**

**public DbSet<OrderItem> OrderItems { get; set; }**

**public DbSet<Delivery> Deliveries { get; set; }**

**public DbSet<Payment> Payments { get; set; }**

**public DbSet<Review> Reviews { get; set; }**

**public DbSet<Offer> Offers { get; set; }**

**public DbSet<OrderOffer> OrderOffers { get; set; }**

**protected override void OnModelCreating(ModelBuilder modelBuilder)**

**{**

**base.OnModelCreating(modelBuilder);**

**// Configure composite keys for many-to-many bridging tables**

**modelBuilder.Entity<RestaurantCuisine>()**

**.HasKey(rc => new { rc.RestaurantId, rc.CuisineId });**

**// Configure relationships to prevent multiple cascade paths**

**modelBuilder.Entity<Order>()**

**.HasOne(o => o.Customer)**

**.WithMany(u => u.Orders)**

**.HasForeignKey(o => o.CustomerId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascade deletion**

**modelBuilder.Entity<Order>()**

**.HasOne(o => o.Restaurant)**

**.WithMany(r => r.Orders)**

**.HasForeignKey(o => o.RestaurantId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascade deletion**

**modelBuilder.Entity<Delivery>()**

**.HasOne(d => d.Order)**

**.WithOne(o => o.Delivery)**

**.HasForeignKey<Delivery>(d => d.OrderId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascading delete**

**modelBuilder.Entity<Delivery>()**

**.HasOne(d => d.DeliveryPartnerProfile)**

**.WithMany(u => u.Deliveries)**

**.HasForeignKey(d => d.DeliveryPartnerProfileId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascading delete**

**modelBuilder.Entity<Order>()**

**.HasOne(o => o.Payment)**

**.WithOne(p => p.Order)**

**.HasForeignKey<Payment>(p => p.OrderId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascading delete**

**modelBuilder.Entity<Order>()**

**.HasOne(o => o.UserAddress)**

**.WithMany()**

**.HasForeignKey(o => o.UserAddressId)**

**.OnDelete(DeleteBehavior.Restrict); // Prevent cascading delete**

**// Master table seeding**

**// 1. RoleMaster**

**modelBuilder.Entity<RoleMaster>().HasData(**

**new RoleMaster { RoleMasterId = 1, RoleName = "SuperAdmin", Description = "Super Admin", IsActive = true },**

**new RoleMaster { RoleMasterId = 2, RoleName = "Admin", Description = "System administrator", IsActive = true },**

**new RoleMaster { RoleMasterId = 3, RoleName = "Customer", Description = "Regular user placing orders", IsActive = true },**

**new RoleMaster { RoleMasterId = 4, RoleName = "DeliveryPartner", Description = "Delivery personnel", IsActive = true },**

**new RoleMaster { RoleMasterId = 5, RoleName = "RestaurantOwner", Description = "Restaurant owner", IsActive = true }**

**);**

**// 2. OrderStatusMaster**

**modelBuilder.Entity<OrderStatusMaster>().HasData(**

**new OrderStatusMaster { OrderStatusMasterId = 1, StatusName = "Placed", Description = "Order received", IsActive = true },**

**new OrderStatusMaster { OrderStatusMasterId = 2, StatusName = "Confirmed", Description = "Restaurant confirmed the order", IsActive = true },**

**new OrderStatusMaster { OrderStatusMasterId = 3, StatusName = "Preparing", Description = "Food is being prepared", IsActive = true },**

**new OrderStatusMaster { OrderStatusMasterId = 4, StatusName = "OutForDelivery", Description = "Rider is out for delivery", IsActive = true },**

**new OrderStatusMaster { OrderStatusMasterId = 5, StatusName = "Delivered", Description = "Order delivered", IsActive = true },**

**new OrderStatusMaster { OrderStatusMasterId = 6, StatusName = "Cancelled", Description = "Order cancelled", IsActive = true }**

**);**

**// 3. DeliveryStatusMaster**

**modelBuilder.Entity<DeliveryStatusMaster>().HasData(**

**new DeliveryStatusMaster { DeliveryStatusMasterId = 1, StatusName = "Assigned", Description = "Delivery assigned to partner", IsActive = true },**

**new DeliveryStatusMaster { DeliveryStatusMasterId = 2, StatusName = "PickedUp", Description = "Delivery partner has picked up the order", IsActive = true },**

**new DeliveryStatusMaster { DeliveryStatusMasterId = 3, StatusName = "EnRoute", Description = "Order is on the way", IsActive = true },**

**new DeliveryStatusMaster { DeliveryStatusMasterId = 4, StatusName = "Delivered", Description = "Order delivered to the customer", IsActive = true }**

**);**

**// 4. PaymentStatusMaster**

**modelBuilder.Entity<PaymentStatusMaster>().HasData(**

**new PaymentStatusMaster { PaymentStatusMasterId = 1, StatusName = "Pending", Description = "Awaiting payment confirmation", IsActive = true },**

**new PaymentStatusMaster { PaymentStatusMasterId = 2, StatusName = "Completed", Description = "Payment received", IsActive = true },**

**new PaymentStatusMaster { PaymentStatusMasterId = 3, StatusName = "Failed", Description = "Payment failed", IsActive = true }**

**);**

**// 5. PaymentTypeMaster**

**modelBuilder.Entity<PaymentTypeMaster>().HasData(**

**new PaymentTypeMaster { PaymentTypeMasterId = 1, TypeName = "CreditCard", Description = "Pay using credit card", IsActive = true },**

**new PaymentTypeMaster { PaymentTypeMasterId = 2, TypeName = "UPI", Description = "Unified Payments Interface", IsActive = true },**

**new PaymentTypeMaster { PaymentTypeMasterId = 3, TypeName = "CashOnDelivery", Description = "Pay with cash upon delivery", IsActive = true }**

**);**

**// 6. Seed Cuisine data**

**modelBuilder.Entity<Cuisine>().HasData(**

**new Cuisine { CuisineId = 1, CuisineName = "Indian", IsActive = true },**

**new Cuisine { CuisineId = 2, CuisineName = "Chinese", IsActive = true },**

**new Cuisine { CuisineId = 3, CuisineName = "Italian", IsActive = true },**

**new Cuisine { CuisineId = 4, CuisineName = "Mexican", IsActive = true },**

**new Cuisine { CuisineId = 5, CuisineName = "American", IsActive = true },**

**new Cuisine { CuisineId = 6, CuisineName = "Thai", IsActive = true }**

**);**

**}**

**}**

**}**

**Modify AppSettings.json File:**

Please modify the **appsettings.json** file as follows. It contains configuration settings for the application. The ConnectionStrings section holds the SQL Server connection string (DefaultConnection) that the DbContext uses to connect to the database.

**{**

**"ConnectionStrings": {**

**"DefaultConnection": "Server=LAPTOP-6P5NK25R\\SQLSERVER2022DEV;Database=FoodDeliveryAppDB;Trusted\_Connection=True;TrustServerCertificate=True;"**

**},**

**"AllowedHosts": "\*",**

**"Serilog": {**

**"MinimumLevel": {**

**"Default": "Information",**

**"Override": {**

**"Microsoft": "Error",**

**"System": "Error"**

**}**

**},**

**"WriteTo": [**

**{**

**// Wrap sinks with the Async sink to enable asynchronous logging.**

**"Name": "Async",**

**"Args": {**

**"configure": [**

**{**

**// Asynchronously log to a file with rolling, retention, and file size limit settings.**

**"Name": "File",**

**"Args": {**

**// File sink configuration**

**"path": "logs/MyAppLog-.txt",**

**"rollingInterval": "Day",**

**"retainedFileCountLimit": 30,**

**"fileSizeLimitBytes": 10485760, // 10 MB per file (10 \* 1024 \* 1024 bytes)**

**"rollOnFileSizeLimit": true,**

**"outputTemplate": "{Timestamp:yyyy-MM-dd HH:mm:ss} [{Level:u3}] [{SourceContext}] {Message:lj}{NewLine}{Exception}"**

**}**

**}**

**]**

**}**

**}**

**]**

**},**

**"EmailSettings": {**

**"SmtpServer": "smtp.gmail.com",**

**"SmtpPort": "587",**

**"SenderName": "My Food Delivery App",**

**"SenderEmail": "YOUR\_GMAIL\_ID",**

**"Password": "YOUR\_GMAIL\_APP\_PASSWORD"**

**},**

**"GoogleMaps": {**

**"ApiKey": "YOUR\_GOOGLE\_MAP\_API\_KEY"**

**}**

**}**

**How to get the YOUR\_GOOGLE\_MAPS\_API\_KEY?**

* **Sign In to Google Cloud Console:** Go to Google Cloud Console: [**https://console.cloud.google.com/**](https://console.cloud.google.com/)andsign in with your Google account.
* **Create or Select a Project:** Click on the project dropdown at the top and select an existing project or click New Project to create one. When creating a new project, give it a name, select an organization (if applicable), and click Create.
* **Enable Billing (if not already enabled):** Google Maps Platform requires a billing account even for free usage. Navigate to the Billing section in the Cloud Console and set up a billing account if you haven’t done so already.
* **Enable the Required API:** In your project, go to the Navigation menu (☰) > APIs & Services > Library. Search for "**Maps Embed API**" (or any other Maps API you need, like "Maps JavaScript API"). Click on the API and then click the "Enable" button.
* **Navigate to APIs & Services → Library.** Search for **Maps JavaScript API**. Click on it and then click the Enable button.
* **Navigate to APIs & Services → Library.** Search for **Geocoding API** and click on it. If it isn’t already enabled, click Enable.
* **Create Credentials (API Key):** Go to APIs & Services > Credentials. Click on "+ Create Credentials" and select "API key". A dialog will appear with your new API key. Copy this key for use in your application.
* **Restrict Your API Key (Recommended):** Click the "Edit API key" icon next to your key. Under Key restrictions, choose the restrictions that best suit your project (e.g., restrict by HTTP referrers or IP addresses, and select the specific APIs like "Maps Embed API"). Click Save to apply your restrictions. Now, replace "YOUR\_GOOGLE\_MAPS\_API\_KEY" in your code with the API key you obtained.

**Modify the Program Class:**

The Program.cs class serves as the entry point for the ASP.NET Core application. It registers services such as controllers with views, the ApplicationDbContext (using SQL Server with the connection string from appsettings.json), and cookie-based authentication. It also sets up middleware components (HTTPS redirection, static files, routing, authentication, and authorization) and maps default routes and starts the application. So, please modify the Program class as follows.

**using FoodDeliveryApp.Data;**

**using Microsoft.AspNetCore.Authentication.Cookies;**

**using Microsoft.EntityFrameworkCore;**

**using Serilog;**

**namespace FoodDeliveryApp**

**{**

**public class Program**

**{**

**public static void Main(string[] args)**

**{**

**var builder = WebApplication.CreateBuilder(args);**

**// Add services to the container.**

**builder.Services.AddControllersWithViews();**

**// Register in-memory caching service for caching data in RAM**

**builder.Services.AddMemoryCache();**

**// Clear the default logging providers.**

**builder.Logging.ClearProviders();**

**// Configure the host to use Serilog as the logging provider.**

**builder.Host.UseSerilog((context, services, configuration) =>**

**{**

**// Reads configuration settings for Serilog from appsettings.json.**

**configuration.ReadFrom.Configuration(context.Configuration);**

**});**

**// Configure SQL Server DbContext**

**builder.Services.AddDbContext<ApplicationDbContext>(options =>**

**options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"))**

**);**

**// Add Cookie Authentication**

**builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)**

**.AddCookie(options =>**

**{**

**// Wherever you want to redirect when unauthenticated and unauthorized**

**options.LoginPath = "/Account/Login";**

**options.AccessDeniedPath = "/Account/AccessDenied";**

**options.ExpireTimeSpan = TimeSpan.FromMinutes(60);**

**options.SlidingExpiration = true;**

**});**

**var app = builder.Build();**

**// Configure the HTTP request pipeline.**

**if (!app.Environment.IsDevelopment())**

**{**

**app.UseExceptionHandler("/Home/Error");**

**app.UseHsts();**

**}**

**app.UseHttpsRedirection();**

**app.UseStaticFiles();**

**app.UseRouting();**

**// Enable Authentication & Authorization**

**app.UseAuthentication();**

**app.UseAuthorization();**

**app.MapControllerRoute(**

**name: "default",**

**pattern: "{controller=Home}/{action=Index}/{id?}");**

**app.Run();**

**}**

**}**

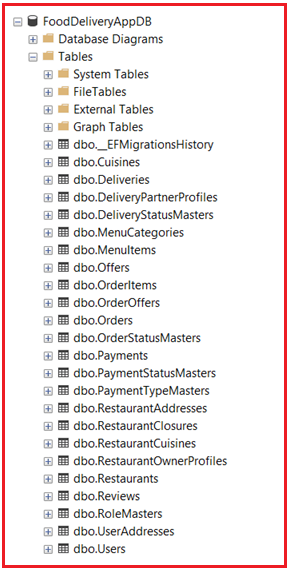
**}**

**Database Migrations**

Open the Package Manager Console in Visual Studio, and then execute the following commands:

* Add-Migration Mig1
* Update-Database

This should create the FoodDeliveryAppDB with the required database tables, as shown in the image below.

****