# ShieldMyRide Project - Detailed Explanation of Attributes, Mappings, and Methods

## 1. Solution Structure Overview

The ShieldMyRide solution is an ASP.NET Core project structured into several key components: - **Program.cs**: Entry point, configures services and middleware. - **Context/MyDBContext.cs**: Entity Framework Core context, mapping entities to database tables. - **Authentication**: Contains ApplicationUser, UserRoles, Login, Register models, and controllers for handling authentication and authorization. - **Controllers**: Handle API endpoints (Claims, Payments, Policies, Quotes, Users, etc.). - **DTOs**: Data Transfer Objects are used to separate database entities from what is exposed to clients. - **Configuration (appsettings.json)**: Stores connection strings, JWT keys, and other environment settings.

## 2. Attributes & Mappings Explanation

ASP.NET Core with Entity Framework uses attributes to define how C# classes map to database tables and how validation works. - **[Key]**: Marks a property as the primary key. If removed, EF will try to infer a key but may fail. - **[Required]**: Ensures a property cannot be null. Removing it allows null values in the database, which may break business rules. - **[ForeignKey]**: Establishes a relationship between entities. If changed/removed, navigation properties may not work and EF could create extra tables. - **[MaxLength]/[StringLength]**: Defines the maximum size of a column. If not set, EF defaults to large text fields which can impact performance. - **[Table("Name")]**: Maps a class to a specific table name. If removed, EF uses the class name by default. - **[Column("Name")]**: Maps a property to a specific column name. Removing it uses the property name by default. - **[NotMapped]**: Excludes a property from being stored in the database. If removed, EF may try to persist it, causing migration issues.

## 3. Entity Framework Core - MyDBContext

The `MyDBContext` inherits from `DbContext` and defines `DbSet` properties for each entity. This tells EF Core which tables exist. - If a `DbSet` is missing, that entity will not be included in migrations or queries. - `OnModelCreating` can be used with Fluent API for advanced configuration (relationships, cascade delete, indexes). - Changing relationships here (One-to-Many vs Many-to-Many) directly impacts the database schema and how LINQ queries work.

## 4. Authentication & Roles

- **ApplicationUser**: Extends IdentityUser to include custom fields (e.g., Name, Role). - **UserRoles**: Defines constants like Admin, Officer, User for role-based authorization. - **AuthenticationController**: Issues JWT tokens. If token generation or validation is misconfigured, users cannot authenticate. - **[Authorize(Roles = "Admin")]**: Restricts endpoints. Removing it exposes sensitive APIs to all users.

## 5. Data Transfer Objects (DTOs)

DTOs decouple database models from API responses: - **GetDTO**: Used for fetching and displaying data. - **PostDTO**: Used for creating new records. - **UpdateDTO**: Used for editing existing records. If DTOs are not used, exposing EF entities directly can lead to over-posting attacks, data leaks, and tight coupling between DB schema and API.

## 6. Controllers & Methods

- **[HttpGet], [HttpPost], [HttpPut], [HttpDelete]**: Define which HTTP methods are supported. If mismatched, requests will fail. - **ModelState.IsValid**: Ensures validation before saving data. If skipped, invalid data enters the database. - **Dependency Injection**: Controllers receive services (DbContext, Repositories) via constructor. Without DI, code becomes tightly coupled and harder to test.

## 7. Configuration & Program.cs

- **appsettings.json**: Stores DB connection strings, JWT keys, logging configs. If misconfigured, app will fail at runtime. - **Program.cs**: Registers services and middleware: - `AddDbContext()`: Configures EF Core. - `AddIdentity()`: Configures authentication. - `UseAuthentication()` before `UseAuthorization()` is mandatory; switching order breaks login. - **Swagger**: Configured for API testing. Removing it won’t affect runtime but reduces dev visibility.