Data Model (5%)

- Design a database schema with ≥ 3 interconnected entities
    - Models/Vehicle.cs
    - Models/Reservation.cs
    - Models/BusinessInquiry.cs
- Implement at least one many-to-many relationship
    - Models/VehicleFeature.cs
- Use Entity Framework Core for DB operations
    - Data/ApplicationDbContext.c
- Data validation via annotations + server-side
    - Attributes in Models/Vehicle.cs, Models/Reservation.cs, Models/BusinessInquiry.cs, Models/Feature.cs

Application Functionalities (50%)

- CRUD for all entities
    - Controllers/VehiclesController.cs
    - Controllers/ReservationsController.cs
    - Controllers/AdminController.cs
    - Views/Shared/_FeatureAssignModal.cshtml
- ≥ 1 complex workflow
    - Controllers/HomeController.cs (Reserve, ConfirmReservation, Checkout)

    - wwwroot/js/FeatureAssign.js

- User authentication & authorization
    - [Authorize(Roles="Admin")] in VehiclesController.cs, AdminController.cs, ReservationsController.cs, FeatureController.cs
- Dashboard with summary statistics
    - Views/Home/Statistics.cshtml

REST Web API (15%)

- Design & implement RESTful endpoints
    - /Feature/GetAll
    - /Vehicles/AssignFeature, /Vehicles/RemoveFeature
- Proper HTTP methods
    - All above implemented as [HttpGet] for JSON reads/assigns

JavaScript and AJAX (15%)

- ≥ 2 JS-powered features
    - wwwroot/js/PaymentEstimator.js
    - wwwroot/js/BusinessInquiry.js
- AJAX calls for data updates
    - wwwroot/js/FeatureAssign.js

User Interface and UX (10%)

- Use a CSS framework for responsive design

    - Views/Shared/_Layout.cshtml
- Custom theme / visual customizations
    - wwwroot/css/custom-theme.css

- Intuitive navigation
    - Home, Vehicles, About Us, Admin, Profile, Login/Logout)
    - Views/Admin/Index.cshtml