# Week 1: PCBTracker Explanations

## Step 1: Setting Up Your Solution

**.NET SDK & MAUI Workload:** Tools to build .NET apps. Confirm installation with dotnet --list-sdks. MAUI workload enables cross-platform UI development.

**Creating a Solution (.sln):** Run dotnet new sln -n PCBTracker. A solution groups multiple projects for combined building.

## Step 2: Scaffolding Your Projects

**Domain Layer (classlib):** dotnet new classlib -n PCBTracker.Domain. Contains pure business entities: Board, Skid, User.

**Data Layer (classlib):** dotnet new classlib -n PCBTracker.Data. Adds EF Core for database access.

**UI Layer (MAUI):** dotnet new maui -n PCBTracker.UI. Default MAUI app template with MainPage, MauiProgram.cs.

**Wiring Projects:** UI → Data, Data → Domain via project references ensures correct build order and code visibility.

## Step 3: Defining Domain Entities

Define POCOs (Plain Old CLR Objects) with properties only:

- **Board:** BoardID, SerialNumber, PartNumber, BoardType, SkidID, PrepDate, ShipDate?, IsShipped, Navigation to Skid.

- **Skid:** SkidID, SkidName, ICollection.

- **User:** UserID, Username, PasswordHash, Role.

## Step 4: Configuring the Data Layer with EF Core

**Add EF Core Packages:** Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Design.

**AppDbContext:** Inherit from DbContext, add DbSet, DbSet, DbSet. OnModelCreating: unique index on SerialNumber.

**Connection String & LocalDB:** Data Source=(LocalDB)\MSSQLLocalDB; Initial Catalog=PCBTracking; Integrated Security=True; Register with DI in MauiProgram.cs.

## Step 5: Creating the Database with Migrations

**Install EF CLI:** dotnet tool install --global dotnet-ef.

**Design-Time DbContext Factory:** Implement IDesignTimeDbContextFactory for migrations.

**Add & Apply Initial Migration:** dotnet ef migrations add InitialCreate -p PCBTracker.Data -s PCBTracker.Data, then dotnet ef database update -p PCBTracker.Data -s PCBTracker.Data.

**Verify:** In SQL Server Object Explorer, see PCBTracking database with Boards, Skids, Users tables.

## Step 6: Seeding an Admin User

**Add BCrypt.Net-Next:** For password hashing, run dotnet add PCBTracker.UI package BCrypt.Net-Next.

**Migration & Seed on Startup:** In MauiProgram.cs, create a service scope, run db.Database.Migrate(), then if no users exist, add an Admin user with BCrypt.HashPassword("password").

**Verify:** Users table shows a row: Username=admin, PasswordHash=[BCrypt hash], Role=Admin.

## Dependency Injection (DI)

DI is a design pattern to decouple class dependencies. Inversion of Control means an external container builds classes. Constructor Injection passes dependencies via constructor. Register services with lifetimes: AddTransient (new each time), AddScoped (one per scope), AddSingleton (one per app). Benefits: testability, maintainability, lifecycle management. MAUI uses built-in DI via Microsoft.Extensions.DependencyInjection.