# Codex HRMS Clean Architecture – Updated Detailed Prompts (with .gitignore)

## PR #0 – .gitignore

## Build artifacts  
bin/  
obj/  
[Bb]uild/  
[Ll]ogs/  
\*.log  
\*.trace  
  
## Visual Studio / Rider  
.vs/  
\*.suo  
\*.user  
\*.userosscache  
\*.sln.docstates  
.idea/  
\*.DotSettings.user  
.ReSharper\*/  
\*\_Resharper.\*  
[Tt]est[Rr]esult\*/  
[Bb]enchmarkDotNet.Artifacts/  
  
## VS Code  
.vscode/  
  
## .NET tooling  
project.lock.json  
project.fragment.lock.json  
artifacts/  
\*.nupkg  
\*.snupkg  
  
## Packages / cache  
packages/  
\*.cache  
\*.tmp  
  
## Database / local data files  
\*.mdf  
\*.ldf  
\*.ndf  
\*.db  
\*.opendb  
\*.jfm  
  
## Symbols / binaries  
\*.pdb  
\*.mdb  
\*.exe  
\*.dll  
\*.so  
\*.dylib  
  
## OS junk  
.DS\_Store  
Thumbs.db  
  
## Node/Web (if used by UI)  
node\_modules/  
dist/  
coverage/  
  
## Publish / deploy output  
\*.PublishSettings  
\*.pubxml  
\*.pubxml.user  
.publish/  
publish/  
wwwroot/\*.map  
  
## IMPORTANT: Do not ignore these  
# - appsettings.json (keep, but never put real secrets)  
# - Migrations/ (EF Core migrations are code, track them)  
# - wwwroot/ (static assets for UI, track them)

## PR #1 – Solution Scaffold

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #1 – Solution & Projects Scaffold):  
1. Create a .NET 8 solution named HRMS.sln with six projects:  
 - HRMS.Models (Class Library)  
 - HRMS.DataAccess (Class Library)  
 - HRMS.Services (Class Library)  
 - HRMS.API (ASP.NET Core 8 Web API, minimal Program.cs)  
 - HRMS.UI (ASP.NET Core 8 MVC, minimal Program.cs + Views/Home/Index.cshtml placeholder)  
 - HRMS.Tests (xUnit, empty test class)  
2. Add project references:  
 - HRMS.API -> HRMS.Services, HRMS.Models  
 - HRMS.Services -> HRMS.DataAccess, HRMS.Models  
 - HRMS.DataAccess -> HRMS.Models  
 - HRMS.UI -> HRMS.Models  
 - HRMS.Tests -> HRMS.Services  
3. After scaffolding, run:  
 - dotnet restore HRMS.sln  
  
Output: Commit all new/modified files for PR #1.

## PR #2 – Entities & DTOs

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #2 – Entities & DTOs):  
1. In HRMS.Models project, add entities Department, Employee, LeaveBalance with Id, properties, and relationships.  
2. Add DTOs for Create/Update: CreateDepartmentDto, UpdateDepartmentDto, etc.  
3. Add pagination models: PagedRequest and PagedResult<T>.  
4. Follow standard naming and validation annotations.  
  
Output: Commit all new/modified files for PR #2.

## PR #3 – DataAccess & EF Core

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #3 – DataAccess & EF Core):  
1. Ensure EF Core packages are installed in HRMS.DataAccess:  
 - Install-Package Microsoft.EntityFrameworkCore -Project HRMS.DataAccess -Version 8.0.6  
 - Install-Package Microsoft.EntityFrameworkCore.SqlServer -Project HRMS.DataAccess -Version 8.0.6  
 - Install-Package Microsoft.EntityFrameworkCore.Design -Project HRMS.DataAccess -Version 8.0.6  
 - Install-Package Microsoft.EntityFrameworkCore.Tools -Project HRMS.DataAccess -Version 8.0.6  
2. Add AppDbContext with DbSets for Department, Employee, LeaveBalance.  
3. Add IGenericRepository<T> and GenericRepository<T>.  
4. Add EF Core configuration classes for each entity.  
5. Add DesignTimeDbContextFactory reading DefaultConnection from HRMS.API/appsettings.json.  
6. Create InitialCreate migration (ensure both .cs and .Designer.cs generated).  
7. Update HRMS.API Program.cs to register DbContext with SQL Server.  
8. Run NuGet restore and clean:  
 - dotnet restore HRMS.sln  
 - dotnet nuget locals all --clear  
  
Output: Commit all new/modified files for PR #3.

## PR #4 – Services Layer

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #4 – Services Layer):  
1. In HRMS.Services project, add DepartmentService, EmployeeService, LeaveBalanceService implementing CRUD with DTOs.  
2. Validate IDs (positive), emails (must contain @), and leave balances (non-negative).  
3. Services should call GenericRepository<T>.  
4. Update HRMS.API Program.cs to register services in DI container.  
5. Add smoke test in HRMS.Tests using InMemory EF to validate CRUD logic.  
  
Output: Commit all new/modified files for PR #4.

## PR #5 – API Controllers

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #5 – API Controllers):  
1. Ensure EF Core packages are installed in HRMS.API:  
 - Install-Package Microsoft.EntityFrameworkCore.SqlServer -Project HRMS.API -Version 8.0.6  
 - Install-Package Microsoft.EntityFrameworkCore.Design -Project HRMS.API -Version 8.0.6  
 - Install-Package Microsoft.EntityFrameworkCore.Tools -Project HRMS.API -Version 8.0.6  
2. Add DepartmentsController, EmployeesController, LeaveBalancesController.  
3. Implement endpoints: GET (paged list), GET by id, POST, PUT, DELETE.  
4. Translate service exceptions into RFC 7807 ProblemDetails.  
5. Enable Swagger in development.  
6. Add smoke tests with stubbed services.  
7. Run NuGet restore and clean:  
 - dotnet restore HRMS.sln  
 - dotnet nuget locals all --clear  
  
Output: Commit all new/modified files for PR #5.

## PR #6 – MVC UI Scaffold

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #6 – MVC UI Scaffold):  
1. In HRMS.UI project, add Controllers, Views, Models folders.  
2. Add HomeController with Index() action.  
3. Create Views/Home/Index.cshtml with placeholder text "HRMS UI Home Page".  
4. Add Views/Shared/\_Layout.cshtml with Bootstrap navbar/footer.  
5. Configure \_ViewStart.cshtml to use layout.  
6. Add \_ViewImports.cshtml with TagHelpers and HRMS.Models namespace.  
  
Output: Commit all new/modified files for PR #6.

## PR #7 – MVC CRUD Pages

Safety rules:  
- Do NOT run restore/build/test; only create/modify text source files.  
- Respect .gitignore. Exclude bin/, obj/, .vs/, \*.dll, \*.exe, \*.pdb, \*.mdf, \*.ldf, \*.zip.  
  
Task (PR #7 – MVC CRUD Pages):  
1. Scaffold EmployeesController with Index, Details, Create, Edit, Delete.  
2. Scaffold DepartmentsController with Index, Details, Create, Edit, Delete.  
3. Add Views/Employees and Views/Departments with Bootstrap forms/tables.  
4. Use HttpClient to call HRMS.API for CRUD.  
5. Configure API base URL in appsettings.json.  
6. Update \_Layout.cshtml with navbar links for Employees and Departments.  
7. Run NuGet restore and clean:  
 - dotnet restore HRMS.sln  
 - dotnet nuget locals all --clear  
  
Output: Commit all new/modified files for PR #7.

## PR #8- Finalize Create Flow (API & UI) — Departments, Employees, LeaveBalances

Work against repository `zawmyohein1/codex-hrms-clean-architecture` on branch `codex/standardize-create/edit-flows-across-api-and-ui`.

Do NOT modify any other branch. Only commit to this branch.

Goals (fix all):

1) 500 error at CreatedAtAction (route mismatch after create).

2) Wrong/empty red alert banner on Create pages before submit.

3) Compile error in ApiControllerBase.HandleException(Exception).

4) Unit tests: controller constructors must align with ILogger<T> parameter.

Scope

- Minimal, targeted edits. Keep DTOs/services/structure unchanged.

Tasks

A) API (HRMS.API)

1) For controllers with Create + GetById (e.g., Departments, Employees, LeaveBalances):

- Add explicit GET-by-id route name:

[HttpGet("{id:int}", Name = "Get{EntityName}ById")]

- In POST Create, replace CreatedAtAction(...) with:

return CreatedAtRoute("Get{EntityName}ById", new { id = created.Id }, created);

- On validation/duplicate/business errors: return ProblemDetails (400/409). Wrap unexpected exceptions as 500 ProblemDetails.

2) Fix handler compile error

File: HRMS.API/Controllers/ApiControllerBase.cs

Method: protected IActionResult HandleException(Exception ex)

- Use a null-safe switch; replace any `or` with `||` and guard Message with `?.Contains(... ) ?? false`.

- Map: “not found” → 404, “exist/duplicate” → 409, ArgumentException → 400, other InvalidOperationException → 400, fallback → 500.

B) UI (HRMS.UI)

1) Controllers (Departments/Employees/LeaveBalances) – POST Create (and Edit if present):

- On success (2xx): TempData["Success"] = "<Entity> saved successfully."; RedirectToAction(Index).

- On failure: parse RFC7807 ProblemDetails (including ValidationProblemDetails.Errors) to ModelState; fallback to generic error; return View(dto).

2) Views

- In \*\*Create.cshtml / Edit.cshtml\*\* for the above entities: show the top error summary ONLY when `!ViewData.ModelState.IsValid` and include `<div asp-validation-summary="All"></div>`.

- Ensure `@section Scripts { <partial name="\_ValidationScriptsPartial" /> }` is present.

- In \*\*Index.cshtml\*\*: if TempData["Success"] has value, render a green success alert.

- If missing, add `Views/Shared/\_ValidationScriptsPartial.cshtml` (jquery-validate + unobtrusive).

C) Tests (HRMS.Tests)

1) File: HRMS.Tests/Controllers\_SmokeTests.cs

- Add `using Microsoft.Extensions.Logging;` and `using Microsoft.Extensions.Logging.Abstractions;`

- Instantiate controllers with `NullLogger<T>.Instance`:

new DepartmentsController(service, NullLogger<DepartmentsController>.Instance);

new EmployeesController(service, NullLogger<EmployeesController>.Instance);

new LeaveBalancesController(service, NullLogger<LeaveBalancesController>.Instance);

Commits

- 1) API: named GET-by-id + CreatedAtRoute + ProblemDetails + handler fix.

- 2) UI: TempData success + ProblemDetails→ModelState + views (no empty alerts, validation partial).

- 3) Tests: pass ILogger<T> with NullLogger<T>.Instance.

Verification

- Create valid entity → 201 from API; UI redirects to Index with green success banner; row visible.

- Invalid/duplicate → 400/409 ProblemDetails shown on Create page; no generic-only message.

- No 500 from Create; handler compiles and returns correct status codes.

- Tests compile and run without constructor errors.