**Fastest MVP Build Order + Copy‑Paste Prompts**

**0) Foundation (Day 0)**

**Goal:** Create solution, wire logging, EF Core, auth skeleton.

**Deliverables:** Solution structure, Serilog, HealthChecks, DbContext, initial migration, OIDC sign‑in/out.

**Prompts to use:**

1. **Solution skeleton**

Create an ASP.NET Core .NET 8 solution with projects Domain, Application, Infrastructure, Web (Blazor Server + MVC). Add Serilog, HealthChecks, and a shared Directory.Build.props with nullable enabled. Show the .sln, project references, Program.cs wiring, and appsettings stubs.

1. **OpenIddict OIDC integration**

Wire OIDC auth using my OpenIddict server at https://user-auth.unece.org. Use cookie + AddOpenIdConnect (code flow). Map email and roles, enrich from DB on OnTokenValidated, and implement proper single logout (end-session + post\_logout\_redirect\_uri). Include secure cookie options and environment-specific appsettings.

1. **EF Core + baseline DbContext**

Add EF Core (SQL Server) with a single EoDbContext. Create the base tables: ApplicationUser, ApplicationRole, ApplicationUserRole, ActivityLog. Include rowversion on auditable entities, configure snake\_case table names, and generate the first migration + instructions to apply.

**1) Domain Model (Core Entities)**

**Goal:** Get core entities and enums ready for compilation.

**Deliverables:** Entities + value objects + configurations.

**Prompt:**

Define the Domain entities/enums for MVP: Request, RequestType, WorkflowTemplate, WorkflowStepTemplate, WorkflowInstance, WorkflowState, TaskItem, Attachment, SlaRule, EscalationRule, CommentThread, Comment, CommentReaction, Mention. Include key properties, relationships, and rowversion where needed. Provide EF Core configurations (Fluent API) in Infrastructure. Include enums: EntityType, AssignmentMode, WorkflowStateCode, TaskProgressStatus, CommentVisibility.

**2) Database Migrations & Seed**

**Goal:** DB is scaffolded with seed data.

**Prompt:**

Create EF Core migrations for all entities and seed initial data: two RequestType records (Consultant Request, ICT Support), one simple WorkflowTemplate each with 2–3 WorkflowStepTemplates, and a couple of roles (Requester, Reviewer, ProcessOwner, Admin). Provide the migration, seeder, and commands to update the DB.

**3) Access Control Service + Policies**

**Goal:** Centralize per‑step permissions.

**Prompt:**

Implement IAccessControlService that evaluates permissions for actions: by role, by specific assignee (SelectedByPreviousStep), and flags (AllowCreatorOrPreparer). Add authorization policies (named) and show how a Blazor page or controller checks them.

**4) Workflow Engine v1 (Transitions & Assignment)**

**Goal:** Minimal engine to move steps, compute assignees, and emit events.

**Prompts:**

1. **Engine core**

Implement WorkflowEngine with APIs: StartInstance(requestId), CanAdvance(instanceId), Advance(instanceId, byUserId), SkipOrBranch(instanceId, ruleKey). Support assignment modes: RoleBased, SelectedByPreviousStep, AutoAssign. Emit domain events for StepActivated, StepCompleted, AssignmentChanged.

1. **Unit tests**

Provide xUnit tests covering transitions, assignment modes, and a simple branching rule.

**5) SLA Service + Hangfire Jobs**

**Goal:** Due dates, reminders, single escalation.

**Prompt:**

Implement SlaService that computes DueOn on step activation using SlaRule. Create Hangfire jobs: reminders (T-3, T-1) and a single escalation (T+1). Ensure idempotent job keys and outbox-friendly design.

**6) Dynamic Form Wizard (JSON Schema)**

**Goal:** Render forms per step with conditional fields.

**Prompt:**

Build a Blazor component JsonFormWizard that renders step forms from a JSON schema with conditional visibility rules (simple expressions on prior field values). Show schema examples for Consultant Request and ICT Support, including attachments.

**7) Comments & @Mentions (2‑level threads)**

**Goal:** Threaded comments with internal/requester‑visible modes and mentions.

**Prompt:**

Implement CommentService with APIs: AddComment(thread, body, visibility, parentId?), ListThread(entityType, entityId), AddReaction(commentId, emoji), ResolveMentions. Enforce 2-level reply depth. Provide a Blazor CommentsPane with @mention picker, unread counters, and SignalR live updates.

**8) Attachments Module**

**Goal:** Upload, inline view, soft‑delete purge.

**Prompt:**

Implement AttachmentService with allowed file types, size limit, AV scan hook interface, and inline streaming (range support). Provide UI for drag/drop and an inline viewer. Add a Hangfire purge job for soft-deleted files after 90 days.

**9) Notifications (SignalR + Email)**

**Goal:** Real‑time toasts + email templates.

**Prompt:**

Add a SignalR hub /hubs/notifications and a NotificationService that pushes events for submission, assignment, mentions, and overdue. Provide Razor email templates for the same events and a user notification preferences store.

**10) Task Module (Gating)**

**Goal:** Step can’t complete until tasks done.

**Prompt:**

Implement TaskService with task/subtask dependencies and a gating flag that prevents step completion while open tasks exist. Provide a Blazor Task tab to create/update tasks and attach files.

**11) Dashboards (My Requests / Actions / SLA)**

**Goal:** Basic insights for MVP.

**Prompt:**

Create read models and Blazor pages for **My Requests**, **My Actions**, and **SLA Status** (on-time, due soon, overdue). Add CSV export actions and fast filters (type, status, assignee).

**12) Admin UI (Workflow & Form Designer — basic)**

**Goal:** Configure templates & steps without code.

**Prompt:**

Scaffold Admin pages to CRUD WorkflowTemplate/WorkflowStepTemplate, define assignment mode, AllowedRoles, SLA rules, and upload a JSON schema for step forms. Include a Permission Matrix view (Step × Role/User).

**13) Audit Timeline**

**Goal:** Transparent history.

**Prompt:**

Implement ActivityLog write helpers for create/update/status/assignment/comment events and show a timeline tab on the Request details page. Support filtering by event type.

**14) UAT Hardening & DoD**

**Goal:** MVP done.

**Prompt:**

Provide a UAT checklist and scripts: seed both workflows, run SLA jobs, test notifications, verify RBAC, restore drill, and performance counters. Include p95 latency checks (<200ms read, <500ms write) and a security scan checklist.

**🧠 How to use prompts during the project**

* Use **focused prompts per feature** (like the ones above), not a single giant prompt.
* When you hit an edge case, ask a **micro‑prompt**:

In WorkflowEngine.Advance, show how to enforce Task gating and return a friendly error if tasks exist.

* For fixes/refactors, include **the file names** and I’ll provide patches/diffs.

**🗺️ Suggested weekly flow (solo dev)**

**Week 1:** 0–3 (foundation, domain, migrations, access control)  
**Week 2:** 4–7 (workflow v1, SLA jobs, JSON forms, comments)  
**Week 3:** 8–11 (attachments, notifications, tasks, dashboards)  
**Week 4:** 12–14 (admin basics, audit timeline, UAT & DoD)