Workflow Service MVP Implementation Checklist (Revised)

This checklist reflects the MVP Phase from the 'Workflow Engine Development Plan (Revised)' including the basic ReactFlow builder (sidebar palette, property editor, connectable edges, JSON export), minimal validation, JSON-first definitions, small node taxonomy, event-first design, multi-tenant RBAC, timers via DB polling, and safe condition evaluation.

0) Prerequisites

☐ Ensure PostgreSQL and Redis containers are running.

☐ Verify API Gateway is routing AuthService and UserService correctly.

☐ Confirm RBAC seed is loaded and JWT auth works end-to-end.

1) Backend Service – src/services/WorkflowService/

☐ Create project files: WorkflowService.csproj, Program.cs, appsettings.json, appsettings.Development.json.

☐ Add Controllers/: DefinitionsController (draft/publish/list/get), InstancesController (start/get/signal/terminate), TasksController (list/claim/complete), AdminController (retry/moveToNode; requires workflow.admin).

☐ Add Domain/: DSL (WorkflowDefinitionJson.cs, NodeTypes.cs), Models (WorkflowDefinition, WorkflowInstance, WorkflowTask, WorkflowEvent, OutboxMessage), Enums (InstanceStatus, TaskStatus).

☐ Add Persistence/: WorkflowDbContext.cs with jsonb mappings for definitions and contexts; configurations for indexes (TenantId, status, due dates); initial EF migrations.

☐ Add Engine/: WorkflowRuntime (Stateless), ConditionEvaluator (JsonLogic/CEL), Executors (Automatic, HumanTask, Timer, Gateway).

☐ Add Background/: TimerWorker (poll DB for due timers; advance instances).

☐ Add Services/: DefinitionService, InstanceService, TaskService, AdminService, EventPublisher (persist WorkflowEvent + OutboxMessage).

☐ Add Security/: Policies.cs with workflow.read, workflow.write, workflow.admin.

☐ Add Health check endpoint.

2) Shared Libraries

☐ Add DTOs: WorkflowDefinitionDto, PublishDefinitionRequestDto, StartInstanceRequestDto, WorkflowInstanceDto, TaskSummaryDto, CompleteTaskRequestDto.

☐ Add Contracts: IWorkflowReadService (optional), IWorkflowEvents (event names/payloads).

☐ Update RBAC seed with workflow.read, workflow.write, workflow.admin.

3) API Gateway – src/services/ApiGateway/ocelot.json

☐ Add routes: /workflow/definitions/* → WorkflowService:5003.

☐ Add routes: /workflow/instances/* → WorkflowService:5003.

☐ Add routes: /workflow/tasks/* → WorkflowService:5003.

4) Docker – docker/services/WorkflowService.Dockerfile

☐ Create Dockerfile for WorkflowService.

☐ Add service to docker-compose.yml (map ports 5003 and 7003; set DB/JWT/Redis env).

☐ Add healthcheck.

5) Frontend – Builder & Basics (React + TypeScript + MUI Premium)

☐ Create src/services/workflowService.ts (REST wrappers for definitions, instances, tasks).

☐ Create src/components/workflow/DefinitionsTable.tsx, InstanceDetails.tsx, MyTasks.tsx (basic screens to exercise endpoints).

☐ Create src/components/workflow/builder/BuilderCanvas.tsx using ReactFlow.

☐ Add a sidebar palette with node types: Start, End, HumanTask, Automatic, Gateway (exclusive), Timer.

☐ Enable drag-and-drop from sidebar to canvas; connect nodes with edges (ReactFlow).

☐ Add a property panel (right-side drawer) to edit node properties (label, assignees/roles for HumanTask, condition expression for Gateway, delay/until for Timer, small config for Automatic).

☐ Implement Save/Export: serialize the ReactFlow graph to the JSON DSL; send to server as draft definition.

☐ Implement Load: load a definition JSON into the canvas (drafts only for MVP).

☐ Minimal validation on Publish: must have exactly one Start, at least one End, all nodes reachable from Start.

☐ Display basic instance/task views; allow claim/complete actions from UI.

6) Events & Outbox (Event-First)

☐ Persist every state change to WorkflowEvent table (append-only).

☐ Write an OutboxMessage for each emitted domain event (even if only logged in MVP).

☐ Add idempotency keys to outbox records to prepare for future RMQ delivery.

7) Guardrails for MVP Scope

☐ Supported node set limited to: Start, End, HumanTask, Automatic, Gateway (exclusive), Timer.

☐ Definitions are versioned and immutable once published; instances bind to a specific version.

☐ Conditions evaluated via JsonLogic/CEL (no raw JavaScript).

☐ Timers handled by DB-polled TimerWorker (no RabbitMQ yet).

☐ All entities include TenantId; controllers enforce workflow.* permissions.

☐ Admin operations (retry, moveToNode) require workflow.admin.

8) Acceptance Criteria – End-to-End

☐ Publish immutable Definition v1 from builder JSON (server validates Start/End/reachability).

☐ Start an instance from Definition v1; instance appears with current node at Start or first actionable node.

☐ Task inbox shows HumanTask; user can claim and complete it; Gateway routes by condition.

☐ Timer node advances when due (verified by TimerWorker).

☐ Events and Outbox entries recorded for definition publish, instance start, task claim/complete, gateway transition, timer fired.

☐ UI can round-trip: Load draft → Edit → Save → Publish → Start → Complete.

9) Smoke Test Flow (Manual)

☐ Create Definition v1 in builder: Start → HumanTask (Approve Request) → Gateway (approve? true:false) → End(Approved)/End(Denied).

☐ Publish v1 (validation passes).

☐ Start instance; verify task appears in 'My Tasks'.

☐ Complete task with approve=true; verify instance completes at Approved end.

☐ Add a Timer before the HumanTask with +5 minutes; verify TimerWorker advances after due time.