PRODUCT OWNER / ANALYST DELIVERABLE

Project Brief

ProductManager Service | Version 1.0 | Greenfield Microservice

Executive Summary

ProductManager is a critical microservice within Acme Corp's platform modernization initiative, serving as the central configuration and contract management hub for the image editing platform. As part of the **Domain Administrate** layer, this service manages "Products"—predefined specifications for image editing workflows that represent contractual agreements with enterprise customers.

> **Core Mission:** Provide a flexible, maintainable system for defining, versioning, and managing complex image editing workflow specifications that guide both automated (AI) and manual (designer) processing.

Problems Solved

- **Fragmented Product Configuration** → Centralized, versioned catalog of customer contracts
- **Extended onboarding (2-4 weeks)** → Reduced to <1 week with self-service tools
- **Lack of audit trails** → Comprehensive versioning with field-level audit logs
- **Rigid billing models** → Flexible transactional + subscription with add-ons

Key Stakeholders

1. **Intake and Onboarding Team** - Primary users for Product configuration
2. **Account Managers** - Read-only visibility into customer Products
3. **Catalog Managers** - AI Engineers and Script Managers for template management
4. **System Administrators** - Platform operations and monitoring

PRODUCT OWNER / ANALYST DELIVERABLE

Business Success Criteria

Measurable outcomes defining project success

Business Objectives

75%	100+	50+
Onboarding Time Reduction	Active Products (Year 1)	Enterprise Customers

Technical KPIs

Metric	Target	Measurement
Service Uptime	99.9%	< 8.76 hours downtime/year
API Success Rate	99.95%	Excluding 4xx client errors
Event Publishing	99.99%	Domain events to Kafka
Read Operations	p95 < 200ms	API response time
Write Operations	p95 < 500ms	API response time
Cache Hit Rate	> 80%	Redis cache efficiency

User Success Metrics

- **Product Creation Time:** < 2 hours (vs. 2-3 days currently)
- **Self-Service Rate:** 90% of operations without engineering escalation
- **Configuration Error Rate:** < 5% rework before activation
- **Template Reuse:** Average template used in 5+ Products

PRODUCT MANAGER DELIVERABLE

Product Requirements Document

ProductManager PRD | Author: PM Agent | Version 1.1

Goals

- Provide a flexible, maintainable system for defining, versioning, and managing complex image editing workflow specifications
- Enable rapid enterprise customer onboarding, reducing cycle time from 2-4 weeks to less than 1 week
- Serve as the platform's single source of truth for customer contracts and product configuration
- Support sophisticated billing models (transactional + subscription with add-ons)
- Maintain comprehensive audit trails and version history for compliance
- Empower PS Script Team and Intake Team to independently manage catalog entities

Background: ProductManager is a critical greenfield microservice within Acme Corp's platform modernization initiative. The current legacy platform suffers from fragmented product configuration, extended onboarding cycles, and inability to support modern billing models.

Document Structure

Section	Description
Goals & Background	Business context and objectives
Requirements	50 Functional + 30 Non-Functional requirements
UI Design Goals	Desktop-optimized responsive web interface
Technical Assumptions	.NET 8, Clean Architecture, CQRS, Kafka
Epic List	8 epics covering full MVP scope

PRODUCT MANAGER DELIVERABLE

Requirements Summary

Functional and Non-Functional Requirements

Functional Requirements (50 Total)

Category	FR Count	Key Capabilities
Product Lifecycle	FR1-FR5	Create, Activate, Deactivate, Delete, Clone
ProductTask Config	FR6-FR9	Add/Remove tasks, TATtC, PATtC calculation
Billing Models	FR10-FR15	Transactional, Subscription, Add-ons
SLA Configuration	FR16-FR18	Priority levels, Turnaround times, Daily limits
Template Management	FR19-FR23	Create, Activate, Deactivate, Update, Clone
Query Operations	FR24-FR29	Search, Filter, Impact Analysis
Versioning & Audit	FR30-FR32	Snapshots, Field-level audit, History
Event Publishing	FR33-FR39	Kafka events for all lifecycle changes
Catalog Entities	FR-CAT-001-010	PhotoshopScripts, BasicActions, Features

Non-Functional Requirements (30 Total)

p95 <200ms
Read Operations

p95 <500ms
Write Operations

100+
Concurrent Users

- **NFR7-11:** Support 2,000 Products, 500 templates, 50 tasks/Product, 200 versions
- **NFR12-14:** 99.9% uptime, 99.95% API success, 99.99% event delivery
- **NFR15-21:** TLS 1.3 encryption, JWT auth, RBAC, indefinite audit retention
- **NFR28-30:** Clean Architecture, CQRS pattern, 80%+ cache hit ratio

ARCHITECT DELIVERABLE

Architecture Document

Event-Driven Microservice with Clean Architecture + CQRS

Technical Summary

ProductManager is a **stateless backend microservice** built on **Clean Architecture with Domain-Driven Design** principles. The service manages two primary aggregate types: **Product aggregates** (client-specific workflow configurations) and **Catalog Entity aggregates** (reusable components like PhotoshopScripts, BasicActions).

> **Architectural Style:** Event-Driven Microservice with Clean Architecture + CQRS. Zero direct HTTP calls between services—all inter-service communication via Kafka events.

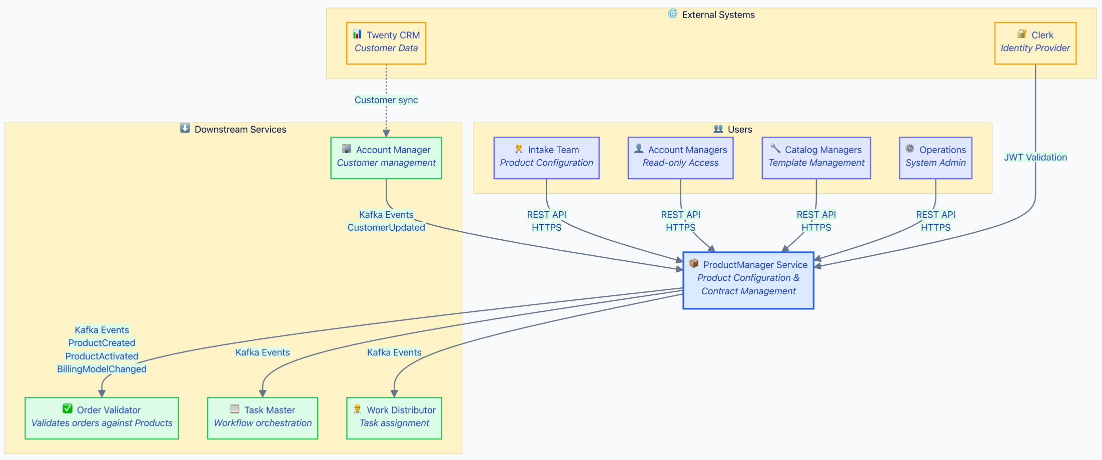
Key Architectural Decisions

Decision	Choice	Rationale
Language/Runtime	`.NET 8 / C# 12`	Platform standard, LTS
Architecture	`Clean Architecture + CQRS`	Testable, AI-assisted dev friendly
Database	`PostgreSQL 15 + EF Core`	ACID, JSON support, migrations
Event Streaming	`Apache Kafka`	Loose coupling, eventual consistency
Caching	`Redis + FusionCache`	Hybrid L1/L2, fail-safe
Workflow	`Temporal`	Durable workflows, saga compensation
Auth	`Clerk JWT + RBAC`	Platform standard

ARCHITECT DELIVERABLE

C4 System Context Diagram

External actors and system boundaries



4
User Types

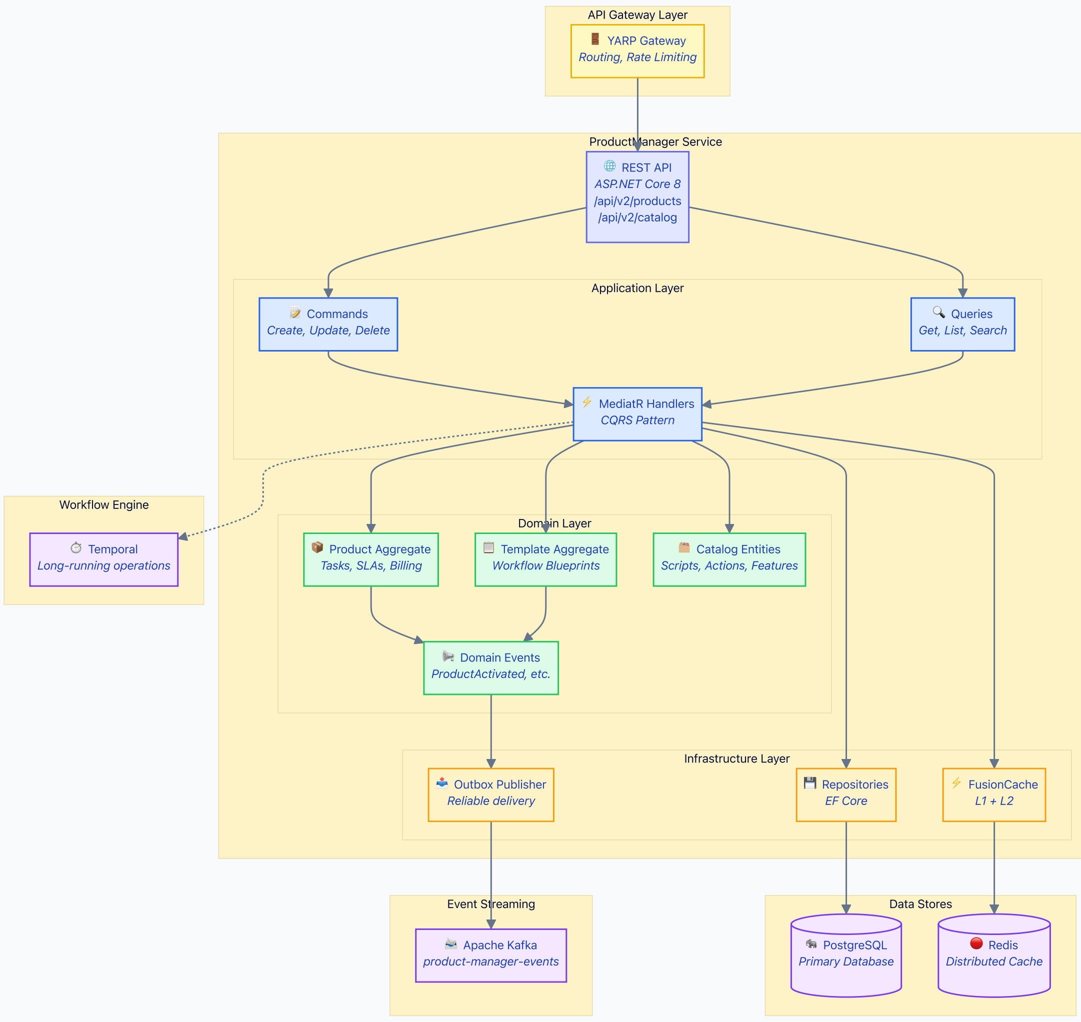
16
Kafka Event Types

4
Downstream Services

ARCHITECT DELIVERABLE

C4 Container Diagram

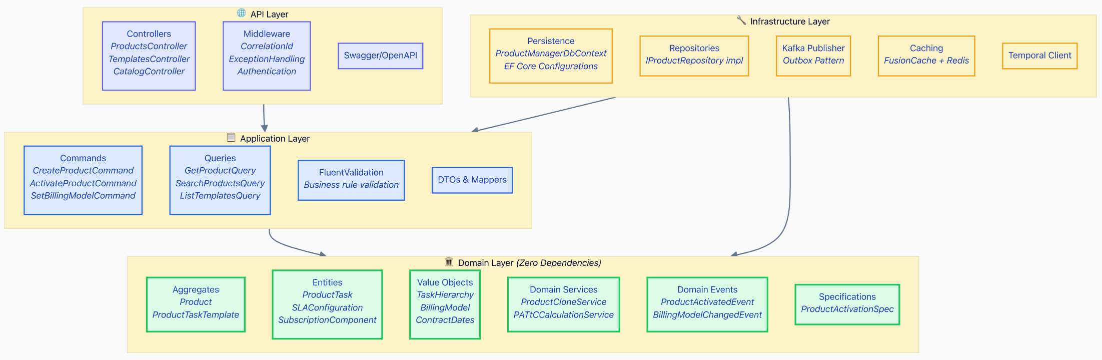
Internal service architecture and data flows



ARCHITECT DELIVERABLE

Clean Architecture Layers

Dependency flow and layer responsibilities



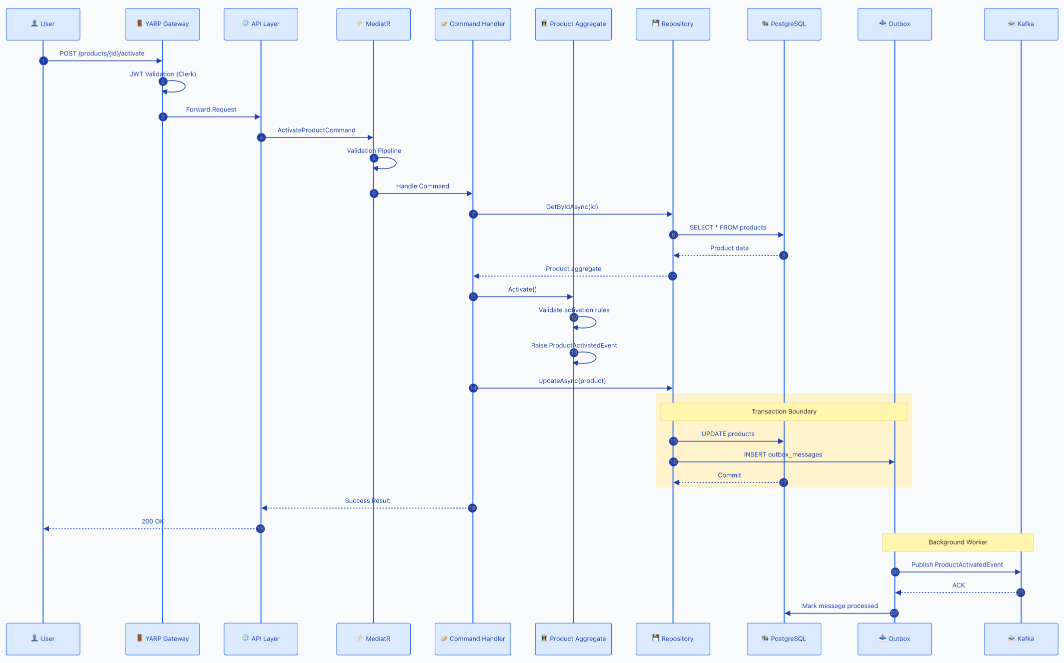
Layer Dependencies

Layer	Dependencies	Key Components
`Domain`	None (pure C#)	Aggregates, Entities, Value Objects, Domain Services
`Application`	Domain only	Commands, Queries, Handlers, Validators
`Infrastructure`	App + Domain	EF Core, Repositories, Kafka, Redis
`API`	All layers	Controllers, Middleware, DI Config

ARCHITECT DELIVERABLE

Data Flow Architecture

Request processing and event publishing flows



Key Patterns Illustrated

- **CQRS:** Commands separated from queries via MediatR pipeline
- **Outbox Pattern:** Events written to DB in same transaction, then published
- **Domain Events:** Raised within aggregate, persisted reliably
- **At-Least-Once Delivery:** Background worker ensures Kafka delivery

PRODUCT MANAGER DELIVERABLE

Epic Structure

8 Epics covering complete MVP scope

Epic	Title	Stories	Focus Area
1	Foundation & Core Product Lifecycle	13	Infrastructure, CRUD, Authentication
2	Workflow Template & Task Configuration	32	Templates, Tasks, Catalog Entities
3	SLA & Billing Model Configuration	11	SLAs, Transactional, Subscription
4	Event-Driven Integration	15	Kafka Events, Versioning, Clone
5	Catalog Entity Management UI	14	Self-service interfaces for teams
6	CI/CD Pipeline & Build Automation	8	GitLab CI, Kubernetes deployments
7	Infrastructure & Observability	7	Monitoring, Alerting, Performance
8	Security Hardening	5	Secrets, Encryption, SAST/DAST

8	97	6
Epics	Total Stories	Sprints Planned

SCRUM MASTER DELIVERABLE

Sprint 1: Foundation

Infrastructure & Core Product CRUD | Weeks 1-2

Sprint Goal

Objective: Establish the foundational infrastructure and implement core Product aggregate with complete CRUD operations, enabling development teams to build on a solid base.

Stories Allocated (10 stories)

Story	Title	Priority	Status
1.1	Project Infrastructure Setup with Health Check	CRITICAL	Done
1.2	Implement Authentication with Clerk	CRITICAL	Done
1.3	Create Product in Draft Status	CRITICAL	Done
1.4	Retrieve Product Details by ID	HIGH	Done
1.5	List and Search Products with Filtering	HIGH	Done
1.6	Update Product Basic Information	HIGH	Done

Sprint Success Criteria

- ✅ All 10 stories completed and QA approved
- ✅ All tests passing (unit + integration)
- ✅ Code coverage ≥ 80% for domain/application layers
- ✅ CI/CD pipeline operational
- ✅ Docker deployment successful

DEVELOPER DELIVERABLE

Story 1.1: Project Infrastructure Setup

Done | QA Score: 95/100

> **As a** DevOps Engineer,
> **I want** a deployable ProductManager service with health check endpoints and monitoring,
> **So that** I can verify the service is running correctly and integrate it into Kubernetes.

Acceptance Criteria (10 ACs)

1. Project created using **ASP.NET Core 8** with **Clean Architecture** structure
2. **PostgreSQL connection** configured via environment variables (12-factor app)
3. **Entity Framework Core 8** with initial migration
4. Health check endpoints: `/health` (liveness) and `/ready` (readiness)
5. Readiness probe validates database connectivity, returns 503 if unreachable
6. **Serilog structured logging** with JSON format and correlation IDs
7. **Prometheus metrics** at `/metrics` endpoint
8. **Multi-stage Dockerfile** for optimized production images
9. **docker-compose** for local development with PostgreSQL
10. **Swagger/OpenAPI** documentation at `/swagger`

Implementation Summary

- **16 tests passing** (7 unit + 9 integration)
- **0 warnings, 0 errors** in build
- **100% reliability** verified over 5 consecutive test runs
- All layers correctly organized with proper separation of concerns

DEVELOPER DELIVERABLE

Story Implementation Details

Technical specifications and file structure

Technology Stack

Component	Version
Language	C# 12.0
Runtime	.NET 8.0 LTS
ORM	Entity Framework Core 8.0.10
Database	PostgreSQL 15.5
Logging	Serilog 3.1.1
Metrics	Prometheus.AspNetCore 8.2.1
Testing	xUnit 2.6.4 + Testcontainers 3.6.0

Project Structure

```
ProductManager/
├── src/
|  ├── ProductManager.API/ (Controllers, Middleware)
|  ├── ProductManager.Application/ (Commands, Queries)
|  ├── ProductManager.Domain/ (Aggregates, Entities)
|  └── ProductManager.Infrastructure/ (Persistence, Events)
├── tests/
|  ├── ProductManager.UnitTests/
|  └── ProductManager.IntegrationTests/
├── Dockerfile
└── docker-compose.yml
```

Key Files Created

- `Program.cs` - Main entry point, DI, middleware configuration
- `CorrelationIdMiddleware.cs` - Request tracing
- `ProductManagerDbContext.cs` - EF Core context
- `DependencyInjection.cs` - Service registration

QA / ANALYST DELIVERABLE

Architecture Validation Report

Generated: 2025-11-16 | Reviewed By: Architect Agent

Overall Assessment: STRONG 🟢

Key Metrics:
- Requirements Coverage: **92%** (23/25 use cases mapped)
- Domain Model Alignment: **95%** (all entities correctly mapped)
- API Specification: **100%** (all endpoints defined)
- Architecture Integrity: **Excellent**

Domain Model Validation

Category	Count	Status
Entities	13	COMPLETE
Value Objects	7	COMPLETE
Enums	5	COMPLETE
Enterprise Business Rules	10	ALL MAPPED
Service Business Rules	5	ALL MAPPED

Use Case Coverage

25/25	25/25	16/16
Use Cases Mapped	API Endpoints	Event Schemas

QA / ANALYST DELIVERABLE

Architecture Quality Assessment

Pattern validation and compliance check

Clean Architecture: EXCELLENT ✅

The architecture strictly follows Clean Architecture principles with proper layer separation:

- **Domain Layer:** Zero external dependencies (pure C# domain logic)
- **Application Layer:** Depends only on Domain (MediatR, FluentValidation)
- **Infrastructure Layer:** Implements Domain/Application interfaces
- **API Layer:** Thin controller layer dispatching to Application

DDD Implementation: EXCELLENT ✅

Pattern	Implementation	Status
Aggregates	Product (root) + child entities	✓
Value Objects	TaskHierarchy, BillingModel, ContractDates	✓
Domain Services	ProductCloneService, PATtCCalculation	✓
Domain Events	ProductActivated, BillingModelChanged	✓
Specifications	ProductActivationSpecification	✓

Event-Driven Architecture: EXCELLENT ✅

- **Outbox Pattern:** Events staged in database before Kafka publish
- **At-Least-Once Delivery:** Transactional consistency guaranteed
- **Dead Letter Queue:** Failed events handled gracefully
- **Saga Pattern:** Temporal workflows for distributed transactions

QA / ANALYST DELIVERABLE

Requirements Traceability

Complete mapping from requirements to implementation

Traceability Matrix

Requirement Type	Total	Mapped	Coverage
Entities	13	13	100%
Value Objects	7	7	100%
Enums	5	5	100%
Enterprise Business Rules	10	10	100%
Service Business Rules	5	5	100%
Use Cases	25	25	100%
API Endpoints	25	25	100%
Event Schemas	16	16	100%
Domain Services	7	7	100%
TOTAL	**113**	**113**	**100%**

Architecture Patterns Applied

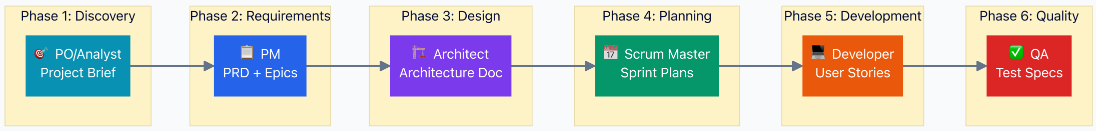
Pattern	Benefit
Clean Architecture	Maintainability, testability
Domain-Driven Design	Business logic clarity
CQRS	Read/write optimization
Event Sourcing	Audit trail, event-driven integration
Repository Pattern	Persistence ignorance
Outbox Pattern	Reliable event delivery

BMAD FRAMEWORK SUMMARY

BMAD Agent Deliverables

Complete workflow from brief to implementation

Agent Workflow



Deliverables by Agent

Phase	Agent	Deliverable	Files
1. Discovery	PO/Analyst	Project Brief	docs/brief.md
2. Requirements	PM	PRD with Epics	docs/prd/*.md (14 files)
3. Architecture	Architect	Architecture Doc	docs/architecture/*.md (25 files)
4. Planning	SM	Sprint Plans	docs/sprints/*.md (6 files)
5. Stories	Dev/SM	User Stories	docs/stories/*.md (97 files)
6. Validation	QA	Test Specs + Report	docs/qa/*.yml (61 files)

6
BMAD Agents

203
Documentation Files

100%
Requirements Traced