Custom CMS Blueprints

C# (.NET) + Python (FastAPI)

Yeh document **do mukammal custom CMS blueprints** deta hai: 1) **C# / .NET based CMS** (Enterprise, long-life) 2) **Python FastAPI based CMS** (AI, automation, data-heavy SaaS)

Dono CMS **subscription-based SaaS** ke liye design kiye gaye hain (pages + product control).

PART A — C# CUSTOM CMS (Enterprise Grade)

1. Kis liye best hai

- Enterprise clients
- Long-term government / corporate systems
- Heavy permissions, audit, compliance

---

2. Tech Stack

- Backend: ASP.NET Core Web API
- ORM: Entity Framework Core
- Auth: JWT + Refresh Token + SAML/OAuth (optional)
- Admin UI: Blazor Server **ya** React
- DB: PostgreSQL / SQL Server
- Cache: Redis

---

3. High-Level Architecture

- API Layer (Controllers)
- Application Layer (Services, Use-Cases)
- Domain Layer (Entities, Rules)
- Infrastructure Layer (DB, External services)

Clean Architecture follow hogi.

---

4. Folder Structure

- src/
- Core/
    - Entities/

    - Interfaces/
    - Enums/
- Application/
    - Services/
    - DTOs/
    - Validators/
- Infrastructure/
    - Db/
    - Repositories/
    - Migrations/
- API/
    - Controllers/
    - Middlewares/
    - Filters/
- AdminUI/
    - Pages/
    - Components/

---

5. Core CMS Modules

- Page Manager (Home, Pricing, Features, Docs, Legal)
- Block Builder (Hero, FeatureList, PricingTable)
- Media Library
- SEO Manager
- Feature Flags
- Subscription Hooks (read-only from billing)

---

6. Data Entities (Example)

- Page
- PageBlock
- Media
- SeoMeta
- FeatureToggle
- AuditLog

---

7. Security

- Role-based Access Control (RBAC)
- Policy-based authorization
- Audit logs immutable
- Input validation + sanitization

---

8. Use Case Example

Admin → Page Edit → Publish → Cache Invalidate → Live Site Update

---

PART B — PYTHON FASTAPI CUSTOM CMS (AI / SaaS Friendly)

1. Kis liye best hai

- AI platforms
- Automation tools
- Rapid iteration SaaS

---

2. Tech Stack

- Backend: FastAPI
- ORM: SQLAlchemy
- Validation: Pydantic
- Auth: JWT + OAuth
- Admin UI: React / Next.js
- Tasks: Celery / RQ
- DB: PostgreSQL

---

3. High-Level Architecture

- API Routers
- Service Layer
- Domain Models
- Async Background Workers

---

4. Folder Structure

- app/
- api/
    - routes/
- core/
    - config.py
    - security.py
- models/
- schemas/
- services/
- workers/
- main.py

---

5. Core CMS Modules

- Page API
- Block Engine (JSON based)
- Docs / Knowledgebase
- Media Service
- SEO + Metadata
- Feature Flags
- Usage-aware content

---

6. Data Models (Example)

- Page
- Block
- DocArticle
- MediaAsset
- SeoMeta
- FeatureToggle

---

7. Security

- Async rate limiting
- Token scopes
- Audit trails
- Input sanitation

---

8. Use Case Example

Editor → Update Pricing Page → Save Draft → Approve → Publish

---

PART C — Dono CMS ka Common Logic

Shared Concepts

- Multi-tenant support
- Subscription-aware rendering
- Feature gating by plan
- Versioned content

PART D — Comparison (Seedhi Baat)

- Stability + compliance → **C# CMS**
- Speed + AI + automation → **Python CMS**
- Best strategy → **Hybrid**
- CMS Core: C#
- AI / Automation APIs: Python FastAPI

PART E — Next Practical Steps

- DB ER diagram generate karna
- Admin UI wireframes (Figma)
- Starter repos (C# + Python)
- API contracts (OpenAPI)

Yeh document **production-grade custom CMS bananay ke liye kaafi hai**.