Berikut tahapan detail harian (8 hari) untuk Backend Engineer menggunakan Cursor AI, sesuai arsitektur microservices Agentic AI SAT/UTBK:

✅ Day 1 – Setup Repo & Struktur Dasar

🔹 Tujuan:

- Setup monorepo/polyrepo untuk microservices.
- Buat Dockerfile dasar untuk setiap service.
- Setup API Gateway sederhana dengan routing.

🔹 Prompt Cursor AI:

"Generate a FastAPI project with Dockerfile for Auth, User, Data, Question, and Assessment services. Create an API Gateway in FastAPI that routes requests to these services. Use a folder structure suitable for microservices."

🔹 Output:

Repo siap, API Gateway dapat menerima request.

✅ Day 2 – Auth & User Service

🔹 Tujuan:

- Implementasi Auth Service (OAuth2, JWT, refresh token).
- Implementasi User Service (CRUD profil, RBAC).
- Buat PostgreSQL schema untuk tabel user dan roles.

🔹 Prompt Cursor AI:

"Create an Auth Service in FastAPI with OAuth2 and JWT support (including refresh tokens). Add a User Service with PostgreSQL integration using SQLAlchemy. Implement CRUD for user profiles and role-based access control (RBAC)."

🔹 Output:

Auth & User Service berjalan dengan PostgreSQL.

✅ Day 3 – Data Service & Vector DB Service

🔹 Tujuan:

- Implementasi Data Service untuk soal, user, aktivitas.
- Setup Vector DB Service (Qdrant/Chroma/Milvus) dan koneksi dasar.

🔹 Prompt Cursor AI:

"Create a Data Service in FastAPI that connects to PostgreSQL with SQLAlchemy for questions, users, and activities tables. Create a Vector DB Service using Qdrant client to store and retrieve embeddings. Add API endpoints for CRUD operations."

🔹 Output:

Data Service & Vector DB siap digunakan.

✅ Day 4 – Embedding & Agent Orchestrator Service

🔹 Tujuan:

- Implementasi Embedding Service (menggunakan OpenAI/SBERT).
- Integrasi ke Vector DB untuk penyimpanan embedding.
- Implementasi awal Agent Orchestrator (Planner, Generator, Evaluator).

🔹 Prompt Cursor AI:

"Create an Embedding Service in FastAPI using OpenAI API or Sentence-BERT to generate embeddings for text. Store embeddings in Qdrant. Implement an Agent Orchestrator Service with basic endpoints for planning, generating, and evaluating content."

🔹 Output:

Embedding Service & Orchestrator pipeline dasar siap.

✅ Day 5 – Question & Assessment Service

🔹 Tujuan:

- Implementasi Question Service (CRUD bank soal, caching).
- Implementasi Assessment Service (sesi ujian, adaptive scoring).

🔹 Prompt Cursor AI:

"Create a Question Service in FastAPI with CRUD endpoints for question bank stored in PostgreSQL. Add caching using Redis. Create an Assessment Service that manages exam sessions and adaptive scoring logic."

🔹 Output:

Question & Assessment Service berjalan.

✅ Day 6 – Analytics & Notification Service

🔹 Tujuan:

- Implementasi Analytics Service (logging aktivitas, analitik performa).
- Implementasi Notification Service (email/real-time notifikasi).

🔹 Prompt Cursor AI:

"Create an Analytics Service in FastAPI to log user activities into PostgreSQL and expose API for aggregated analytics. Create a Notification Service that sends emails (SMTP) and real-time notifications (WebSocket)."

🔹 Output:

Analytics & Notification Service siap.

✅ Day 7 – Integrasi Message Broker & CI/CD

🔹 Tujuan:

- Integrasi Message Broker (Kafka/RabbitMQ) untuk event-driven komunikasi.
- Setup service discovery (Consul/K8s DNS).
- Buat CI/CD pipeline di GitHub Actions.

🔹 Prompt Cursor AI:

"Add event-driven communication using RabbitMQ between services. Implement a basic CI/CD pipeline with GitHub Actions for linting, testing, building Docker images, and pushing to a registry. Add service discovery with Consul for all services."

🔹 Output:

Event bus, CI/CD, dan service discovery aktif.

✅ Day 8 – Testing, Monitoring & Final Deployment

🔹 Tujuan:

- Lakukan end-to-end testing.
- Setup monitoring (Prometheus, Grafana, ELK).
- Buat dokumentasi API (Swagger/OpenAPI).

🔹 Prompt Cursor AI:

"Add OpenAPI/Swagger docs for all services. Setup Prometheus metrics endpoints for FastAPI services. Add Grafana dashboards for monitoring. Perform end-to-end tests for all microservices with pytest."

🔹 Output:

Sistem siap deploy di Kubernetes dengan dokumentasi lengkap.