■ End-to-End Timetable Upload, Extraction, and Retrieval Architecture

1. User Upload & Edge Security

A teacher uploads a timetable (PDF / image / DOCX) from the Web or Mobile UI built using React.

The request is sent securely over HTTPS and first reaches **Azure Front Door**, which acts as the global edge entry point.

Azure Front Door provides TLS termination, Web Application Firewall (WAF), OWASP protection, and global routing for optimal performance.

No backend resources are exposed directly to the public internet at this stage.

2. API Gateway & Policy Enforcement

Requests are forwarded from Azure Front Door to **Azure API Management (APIM)**, which serves as the centralized API gateway.

Inbound policies enforce JWT/OAuth validation via Azure Entra ID, rate limiting, request validation, and header sanitization.

Outbound policies manage response transformation, header masking, and standardized error handling.

Only requests that pass all policies are routed to backend services.

3. Backend Processing & File Storage

APIM routes validated requests to backend services hosted on **Azure App Service or AKS** inside a private Azure Virtual Network (VNet).

The backend performs authorization checks, generates metadata (teacher ID, timetable ID, timestamps), and uses **Managed Identity** for downstream access.

Uploaded files are stored in **Azure Blob Storage** as immutable raw artifacts, preserving original formats for auditability.

Blob Storage acts as the system of record for original inputs.

4. Event-Driven Asynchronous Processing

Upon successful upload, Azure Blob Storage emits a **Blob Created** event.

This event is published to **Azure Service Bus**, which decouples upload from processing.

Service Bus absorbs traffic bursts, supports retries, and provides dead-letter queues.

The UI request completes immediately without waiting for AI processing.

5. Azure Functions – Orchestration Layer

An **Azure Function App** with a Queue Trigger listens to Service Bus messages.

The function retrieves file references, orchestrates extraction workflows, and manages retries and failures.

This layer ensures scalable, resilient, and fault-tolerant processing.

6. OCR & AI-Based Extraction

Azure Functions invoke **Azure Cognitive Services (OCR)** to extract raw text from documents.

Extracted text is processed by **Azure OpenAI / Gemini** to identify timetable structure and normalize it into JSON.

Original subject names, notes, and annotations are preserved.

AI services are isolated from core domain logic.

7. Normalization & Persistence

AI output is validated and normalized by Azure Functions.

Final structured data is persisted into **Azure SQL Database**.

The database stores canonical timetable models, time blocks per day, confidence scores, and metadata.

Azure SQL Database becomes the single source of truth.

8. Timetable Retrieval & UI Display

When a user requests a timetable, the request flows through Azure Front Door → APIM → Backend Service.

The backend queries Azure SQL Database and returns a unified timetable DTO.

The UI never reads from Blob Storage, OCR, or AI services.

9. Key Architectural Principles Enforced

Separation of concerns across upload, processing, and retrieval.

Event-driven architecture for long-running workloads.

Security-first design using WAF, APIM policies, and Managed Identity.

Scalability through queues and serverless compute.

Auditability via immutable raw storage.

Deterministic reads from SQL, not AI outputs.

■ Summary

User interactions remain fast and secure.

AI and OCR processing is reliable, scalable, and decoupled.

All timetable formats are normalized into a unified UI experience.

The system is production-grade, maintainable, and extensible.