C4 Architecture Document Teachers Timetable Management System

What is C4

C4 is a way to describe software architecture at four zoom levels so that different audiences can understand the same system without confusion.

C4 = Context → Containers → Components → Code.

Each level answers a different architectural question.

Level 1 — Context Diagram

Question it answers

Who uses the system, and what external systems does it interact with?

Audience

- Business stakeholders
- CTO / Product leadership
- Security and compliance teams

What it shows

- Teachers Timetable Management System as a black box
- Primary users
- External dependencies
- System security boundary

Actors

- Teacher
- School Admin

System in scope

Teachers Timetable Management System

External systems

- Identity Provider (authentication and authorization)
- OCR Service (text extraction)
- LLM Service (structure interpretation)
- Notification / Email Service (optional)

What this diagram communicates

- Who initiates timetable uploads
- That OCR and LLM are external dependencies
- Where the system boundary exists

Important: No internal details, no Azure services, and no databases appear in the Context diagram.

Level 2 — Container Diagram

Question it answers

What are the major applications or services inside the system, and how do they communicate?

Audience

- Solution architects
- Senior engineers
- Platform and cloud teams

What it shows

- Executables and deployables
- Datastores
- Message queues
- Communication paths

Containers in this system

Frontend

Web UI (React) — allows users to upload timetable documents and view processed results.

Backend

- Timetable API (ASP.NET Core) — owns business workflows and persistence

- Function App — asynchronous orchestration of OCR and LLM processing

Data

- Blob Storage — stores uploaded timetable files
- SQL Database — stores validated timetable data

Messaging

Service Bus — propagates events asynchronously.

External

- OCR Service
- LLM Service

Key flows

- UI → API (upload timetable)
- API → Blob Storage + Service Bus
- Function App → OCR → LLM
- Function App → API (persist validated data)

> Key architectural rule: The Function App orchestrates processing, while the API owns business rules and persistence.

Level 3 — Component Diagram

Question it answers

What are the major building blocks inside a container?

Audience

- Developers
- Code reviewers
- New team members

Example: Timetable API Components

Presentation

TimetableController — handles HTTP requests and responses.

Application

- UploadTimetableUseCase
- ImportTimetableUseCase
- ValidationWorkflow

Domain

- Timetable (Aggregate Root)
- DaySchedule
- TimeSlot
- TimeRange (Value Object)

Infrastructure

- TimetableRepository
- BlobClient
- Observability components

> This diagram makes it explicit that the Domain contains no OCR or LLM logic, all validation occurs in the domain, and infrastructure depends inward.

Level 4 — Code Diagram

Question it answers

How is this implemented in code?

Audience

Developers only.

What it shows

- Classes and interfaces
- Methods
- Aggregates and invariants

Conceptual example

Timetable (Aggregate Root) • AddDaySchedule() • AddTimeSlot() • ValidateNoOverlap() • ValidateMandatoryFields()

TimeRange (Value Object) • StartTime • EndTime • OverlapsWith()

> Level 4 is optional and often represented by actual code and documentation.

How C4 Supports Error Handling & Fallback

- Context level — highlights external dependency risks
- Container level — shows asynchronous fallback paths
- Component level — reveals validation gates
- Code level — enforces invariants and rules

Common Mistakes to Avoid

- Mixing Azure icons into Context diagrams
- Showing classes in Container diagrams
- Placing OCR or LLM inside the Domain
- Skipping Component diagrams for complex systems

Recommended Documentation Structure

- Context Diagram – 1 page
- Container Diagram – 1 page
- Component Diagram (API) – 1–2 pages
- Component Diagram (Function App) – optional
- Code examples – appendix

Final Architect Takeaway

C4 is not about drawing more diagrams. It is about telling the same architectural story at different levels of detail.

For the Teachers Timetable Management System: Context explains who and why, containers explain what runs where, components explain who does what, and code explains how business rules are enforced.