Architecture Design Document

Teachers Timetable Management System

Authoritative architecture reference for developers, architects, and ADRs.

1. Introduction

This document presents a comprehensive end-to-end architecture for the Teachers Timetable Management System using Domain-Driven Design (DDD), Clean Architecture, and a Modular Monolithic approach.

2. Business Problem Statement

Educational institutions require accurate, conflict-free timetables. Manual methods are error-prone, while OCR and AI introduce uncertainty that must be controlled.

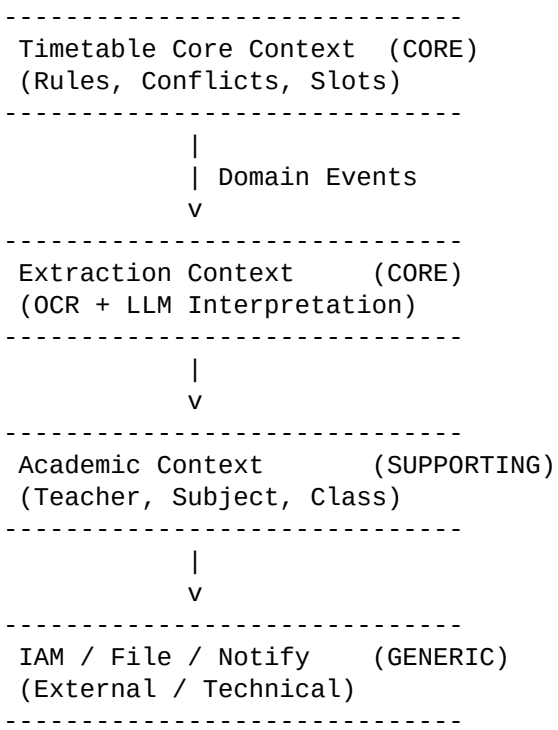
3. Why Domain-Driven Design (DDD)

DDD is adopted to model complex academic rules explicitly and protect them from technical concerns.

4. Core DDD Concepts Applied

- Ubiquitous Language – Timetable, DaySchedule, TimeSlot, TimeRange, TeacherConflict
- Bounded Context – Timetable Management
- Entities – Timetable, TimeSlot, Teacher
- Value Objects – TimeRange, Subject, Period
- Aggregate Root – Timetable
- Domain Services – Conflict detection, availability checks
- Domain Events – TimetableCreated, SlotAdded, ConflictDetected

4.1 Bounded Context Relationships

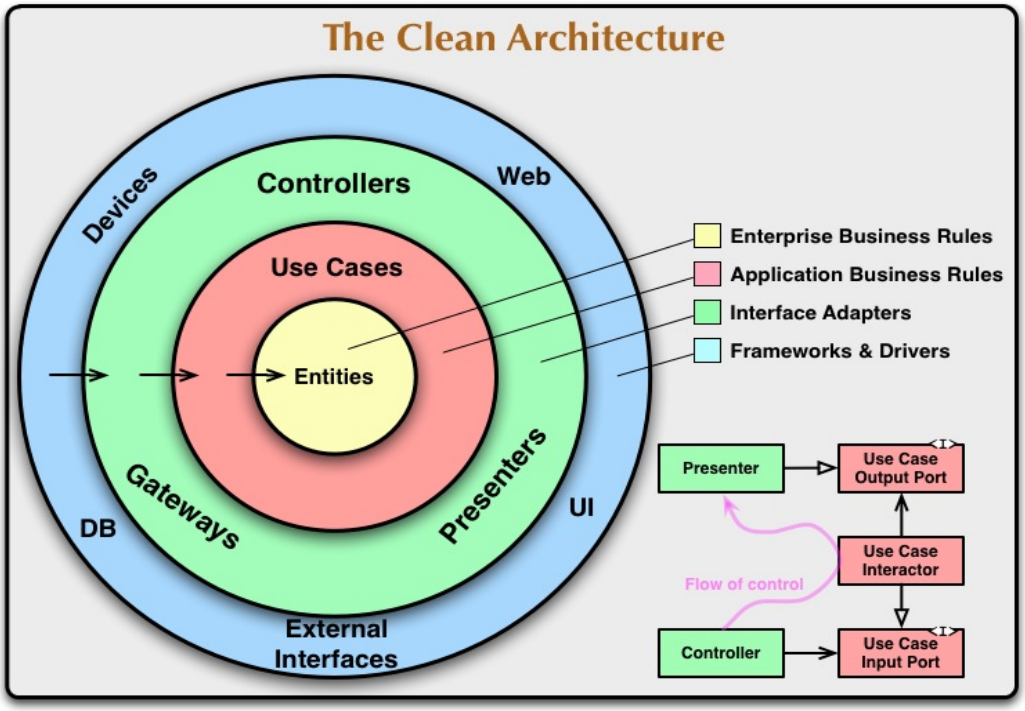


5. Aggregate Design

The Timetable aggregate enforces all scheduling invariants as a single consistency boundary.

6. Clean Architecture Overview

Clean Architecture ensures dependency inversion and separation of concerns.



7. Architecture Layers

Presentation, Application, Domain, Infrastructure layers with inward dependencies.

8. Subdomain Classification

- Core – Timetable Intelligence, Extraction
- Supporting – Academic, Approval, Reporting
- Generic – IAM, Storage, Notifications, Observability

9. Modular Monolithic Architecture

Single deployable system with strongly isolated modules aligned to bounded contexts.

```
TeachersTimetableSystem
- Timetable.Core
- Timetable.Extraction
- Academic.Management
- Approval.Workflow
- Identity.Access
- File.Storage
- Notifications
```

10. End-to-End Workflows

Manual entry and OCR+AI workflows orchestrated via application layer.

11. Error Handling & Consistency

Validation in domain, orchestration errors in application, retries in infrastructure.

12. Security & Governance

Authentication and MFA delegated to external Identity Provider. Domain handles authorization only.

13. Scalability & Performance

Small aggregates, async processing, optimized read models.

14. Testing Strategy

Domain unit tests, application use-case tests, infrastructure integration tests.

15. Architectural Outcome

Strong domain isolation, AI safety, maintainability, and long-term evolvability.