LLM Integration Architecture Teachers Timetable Management System

Principal Architect View • GenAI-safe • Domain-governed • Production-ready

This document defines the complete Large Language Model (LLM) integration strategy for the Teachers Timetable Management System. It covers architectural placement, workflows, prompt strategy, validation, testing and evaluation, observability, and governance.

1. Why LLM is Required

- Interpret unstructured OCR text from PDFs and images
- Normalize timetable semantics such as days, subjects, and time ranges
- Assist humans and systems without becoming a source of truth

2. Core Architectural Principles

- LLM is assistive, not authoritative
- LLM never writes directly to the database
- All outputs must pass domain validation
- Business rules are enforced only in the domain model
- LLM is treated as untrusted infrastructure

3. Placement in Clean Architecture

The LLM is part of the Infrastructure layer. The Application layer orchestrates prompts and validation, while the Domain layer remains the final authority through aggregates.

4. End-to-End Workflow

- File uploaded and stored in Blob Storage
- OCR extracts raw text
- LLM converts OCR output into structured JSON
- Application layer validates schema
- Domain aggregate enforces invariants
- Valid data persisted; invalid data routed to manual review

5. Prompt Strategy

- Schema-first prompting
- JSON-only deterministic output
- Explicit rejection instructions
- Low temperature for stability
- Prompt versioning

6. Handling Hallucinations & Ambiguity

- Domain invariants reject invalid states
- Confidence thresholds where applicable
- Human-in-the-loop for ambiguous cases
- Idempotent retries

7. Validation Pipeline

- JSON schema validation
- Field-level validation
- Aggregate-level business rule enforcement
- Audit logging of rejections

8. LLM Testing & Evaluation Strategy

LLM behavior is continuously tested and evaluated to ensure correctness, robustness, and safety.

8.1 Offline Evaluation

- Golden OCR datasets with expected outputs
- Precision and recall on extracted fields
- Hallucination rate measurement
- Prompt regression testing

8.2 Automated CI Testing

- Mocked LLM responses for unit tests
- Schema validation tests
- Domain rejection scenarios

8.3 Production Monitoring

- LLM acceptance vs rejection rate
- Manual review percentage
- Drift detection across prompt versions
- Cost per successful timetable

LLMs require continuous evaluation, not one-time testing.

9. Reproducibility & Auditability

- Fixed model versions
- Prompt versioning
- Correlation IDs
- Stored OCR and raw LLM outputs
- Response hashing

10. Security Considerations

- Prompt injection mitigation

- Output sanitization
- No PII leakage
- Secure key management
- Role-based access to AI features

11. Observability for LLM Operations

- Metrics: latency, success rate, rejection rate
- Logs: prompt version, model version
- Traces: upload → OCR → LLM → domain validation

12. Cost & Rate Limit Controls

- Token usage monitoring
- Budgets and alerts
- Rate limiting per tenant
- Batch processing where applicable

13. Closing Summary

The domain decides truth. AI assists interpretation.

This LLM integration architecture ensures that GenAI is used safely and effectively. By combining deterministic prompts, layered validation, continuous testing, and strong observability, the system achieves correctness, auditability, and long-term maintainability.