Error Handling & Fallback Architecture (Detailed)
Teachers Timetable Management System

1. First Principle – Errors Are Expected

In the Teachers Timetable Management System, errors are not treated as exceptional conditions. They are expected states of the workflow.

This is a direct consequence of the problem domain. The system works with unstructured inputs (PDFs and images), relies on OCR which is inherently imperfect, integrates Large Language Models which are probabilistic by nature, and enforces strict academic and scheduling rules through the domain model.

For these reasons, error handling is intentionally designed into the architecture from the start, rather than being patched in later as defensive code.

2. Error Handling Philosophy

Across the entire pipeline, the system follows three non-negotiable architectural rules.

- Fail early where validation is cheap and deterministic
- Fail safely when uncertainty or ambiguity exists
- Never persist invalid or incomplete domain state

These principles apply consistently, regardless of whether the failure originates from user input, OCR processing, AI interpretation, or business rule enforcement.

3. End-to-End Error Handling Pipeline

The timetable extraction workflow is intentionally split into multiple stages, each with clear responsibility for error detection and handling.

Upload → Pre-validation → OCR → LLM → Schema Validation → Domain Validation → Persistence → Manual Review

Each stage owns only its class of errors and either passes forward clean data or produces a well-defined failure outcome.

4. Handling Bad Uploads

What is considered a bad upload

- Unsupported or disallowed file formats
- Corrupt or partially uploaded PDF or image files

- Extremely low-resolution or unreadable images
- Empty files with no content
- Files exceeding configured size limits

Where it is handled

Bad uploads are handled strictly at the system boundary, in the Upload or API layer.

Strategy

- Synchronous validation during upload
- Immediate rejection with descriptive error messages
- No forwarding of invalid files to OCR or AI components

Outcome

- Clear and actionable feedback to the user
- No unnecessary downstream processing cost
- No contamination of OCR, AI, or domain pipelines

Example: “Upload rejected: File format not supported or unreadable.”

Architectural insight: Bad uploads are user errors, not AI problems, and must be handled early.

5. Handling Unreadable or Poor OCR Output

Typical OCR failure scenarios

- Skewed or rotated images
- Handwritten timetables that are unreadable
- Poor lighting or scan quality
- Missing or inconsistent table structure

Where it is handled

OCR failures are detected in the OCR stage and interpreted in the application layer.

Strategy

- OCR confidence scoring for extracted text
- Configurable confidence thresholds
- Marking low-confidence outputs as OCR_FAILED

- Skipping LLM processing entirely for unreadable content
- Routing the item to manual review

Outcome

- Prevention of hallucinated AI output
- Human intervention only when genuinely required

Key rule: If OCR cannot reliably read the content, AI must not attempt to guess.

6. Handling Ambiguous Data

Ambiguous data is the most critical and dangerous class of errors in this system. Ambiguity arises when information is partially visible, incomplete, or contextually unclear.

Examples of ambiguity

- A subject name without an associated time slot
- Period references such as “Mon Period 2” without start or end times
- Teacher names that are partially visible
- A day mentioned without a subject

Where it is handled

Ambiguity is handled jointly at the LLM boundary and the domain boundary.

LLM strategy

- The LLM is explicitly instructed not to infer or guess
- Unclear or missing fields are returned as null values

```
{
  "day": "Wednesday",
  "subject": "Mathematics",
  "startTime": null,
  "endTime": null
}
```

Domain strategy

The domain model evaluates ambiguous data using business rules.

- Optional data missing: accepted with flags
- Mandatory data missing: rejected
- Conflicting or overlapping slots: rejected

Rejected entries are routed to the manual review queue.

Architectural insight: Ambiguity itself is acceptable; guessing is not.

7. Handling Missing Fields

Missing data is classified based on its importance to timetable correctness.

Optional fields

- Teacher name
- Room number
- Accepted but marked as incomplete

Mandatory fields

- Day
- Time range
- Subject
- Rejected by the domain model

Where it is enforced

All mandatory field enforcement occurs inside the domain aggregate. This ensures no invalid timetable can be persisted and prevents bypass from UI or AI layers.

8. Manual Review as a First-Class Fallback

When manual review is triggered

- Unreadable OCR output
- Rejected LLM output
- Domain invariant violations
- Low-confidence scenarios

What manual review does

- Allows a human to correct ambiguous or missing information
- Re-applies the same domain validation rules
- Persists data only after successful validation

Manual review is a designed safety mechanism, not a system failure.

9. Retry vs Reject Strategy

Retry conditions

- Transient OCR or infrastructure failures
- Temporary AI service unavailability
- Network or timeout issues

Reject conditions

- Ambiguous data
- Domain rule violations
- Missing mandatory fields

Retrying ambiguous data amplifies risk rather than resolving it.

10. Observability & Feedback Loop

All errors are logged with correlation identifiers and categorized by pipeline stage. This data is used not only for debugging but also for system improvement.

Metrics tracked

- Bad upload rate
- OCR failure rate
- LLM rejection rate
- Manual review percentage

How metrics are used

- Improve user experience
- Tune OCR thresholds
- Refine LLM prompts
- Guide training and documentation

11. Flexibility & Future Evolution

Error handling is implemented as a policy-driven, layered capability. Policies define thresholds, retries, and fallback routing, allowing the system to evolve without changing core domain logic.

- Policies define behavior
- Typed errors define meaning
- Workflows define fallback paths
- The domain defines correctness
- Observability defines evolution

12. Final Architectural Takeaway

Guessing is not error handling.

Retrying ambiguity is dangerous.

Validation gates ensure safety.

The domain model protects correctness.

Manual review is a designed fallback.

The system prefers rejection over corruption.