Learning Yogi Assessment

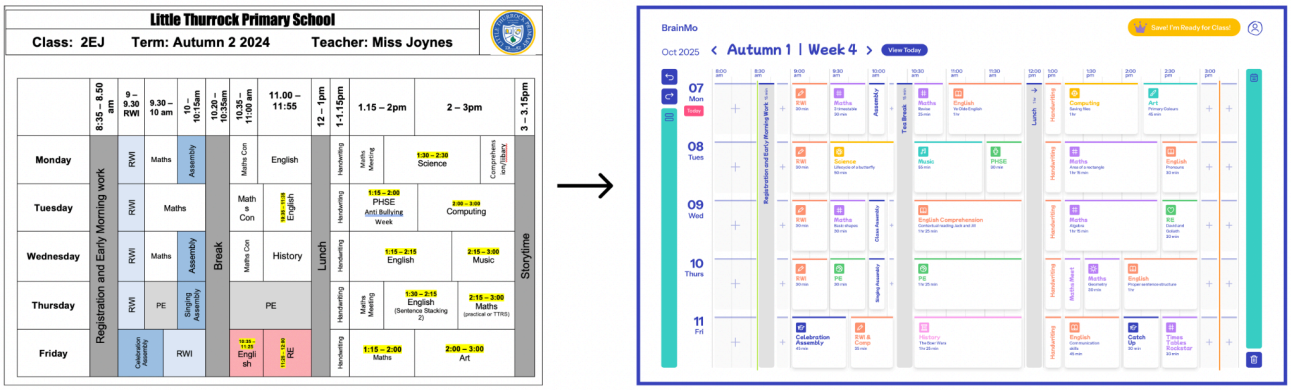
Technical Architect Role

Context

We are developing an online platform for teachers to manage and visualize their weekly class timetables. Teachers often have their timetables in various formats—scanned images, PDFs, Word documents—each styled differently. Our goal is to allow a teacher to upload their current timetable document, and have our system:

1. Extract all relevant Timeblock Events (e.g., “Registration” "Maths", "Play", "Snack Time").
2. Detect Start and End Times or Durations accurately.
3. Preserve the original names and any additional notes teachers may have included.
4. Display the extracted timetable in our own Frontend UI.

Note: Example timetables will be provided to you, but these represent only a small subset of the real-world variety. Your extraction process should be designed to work regardless of how the uploaded timetable is formatted — whether it's typed, scanned, colour-coded, or handwritten. Robustness and adaptability matter.



Simple example of provided teacher timetable to pretty product frontend UI.

For additional information see Designer Handover: [Loom Video Link]

Objective

For this assessment, you’ll design and partially implement this workflow, balancing architectural planning with hands-on API and system design. We want to see how you think, structure, and execute.

You are also strongly encouraged to use AI-based tools (e.g., ChatGPT, Claude, Copilot, Cursor, Windsurf, etc.) and other productivity plugins to accelerate planning and development as you will need to complete this task within 48 hours.

Deliverables

1. Architectural Design Plan

Provide a concise PDF document that outlines:

- End-to-end workflow for the system:
    - File upload → Processing → Extraction → Frontend Display.
    - Presented as an annotated diagram (sequence diagrams, flowcharts, etc.)
- Recommended tools, frameworks, and programming languages for:
    - File ingestion and preprocessing.
    - OCR or document parsing (especially for tables or images).
    - LLM integration.
    - Backend orchestration.
    - Frontend rendering.
- Suggested database schema for storing timeblocks.
- Additional information on your LLM Integration strategy:
    - What part of the pipeline uses it?
    - What is the prompt strategy?
    - How do you ensure accuracy and reproducibility?
- Error handling & fallbacks:
    - How do you handle bad uploads, ambiguous data, or missing fields?
- How will you ensure the system is flexible, for possible future updates and needs?

2. Backend Development Task

Build a small backend prototype in Node.js that:

- Exposes a POST endpoint to receive an uploaded file (image, PDF, or DOCX).
- Processes that file and attempts to extract timetable blocks.
    - You can hardcode assumptions or provide a sample file.
    - Use any combination of OCR libraries (e.g., Tesseract), document readers (e.g., PyMuPDF, pdfplumber), or LLM calls (OpenAI, Claude, etc.).
- Returns a JSON response.

3. Frontend Strategy (Light Touch)

You don’t need to build the full frontend, but include:

- A brief overview of what stack or framework you'd recommend (React, Vue, etc.).
- Any libraries or UI patterns you’d use for:
    - Timetable display
    - Responsive UI for variable timetable formats

Submission Requirements

1. **Source Code Repository:**
    - Please complete the technical implementation in a shared Git repository.
    - Use clear, consistent commit messages that reflect meaningful units of progress.
    - We'll review your Git history to understand your workflow and decisions — so commit early and often, especially after completing:
        - Architectural diagrams / planning stages
        - Backend endpoints
        - Any AI-integration logic
        - Parsing strategies / LLM prompt logic

2. **README Documentation:**
    - Include detailed setup instructions.
    - Document all API endpoints (URLs, methods, request/response formats).
    - Explain any known issues or limitations.
    - Describe how AI-based tools were used to enhance productivity.

3. **Handover video**
    - You are also required to create your own handover video using a tool such as Loom to talk through the task as you've completed it, the decisions you made and why, along with explaining where and how you used ai editors (which is strongly encouraged).