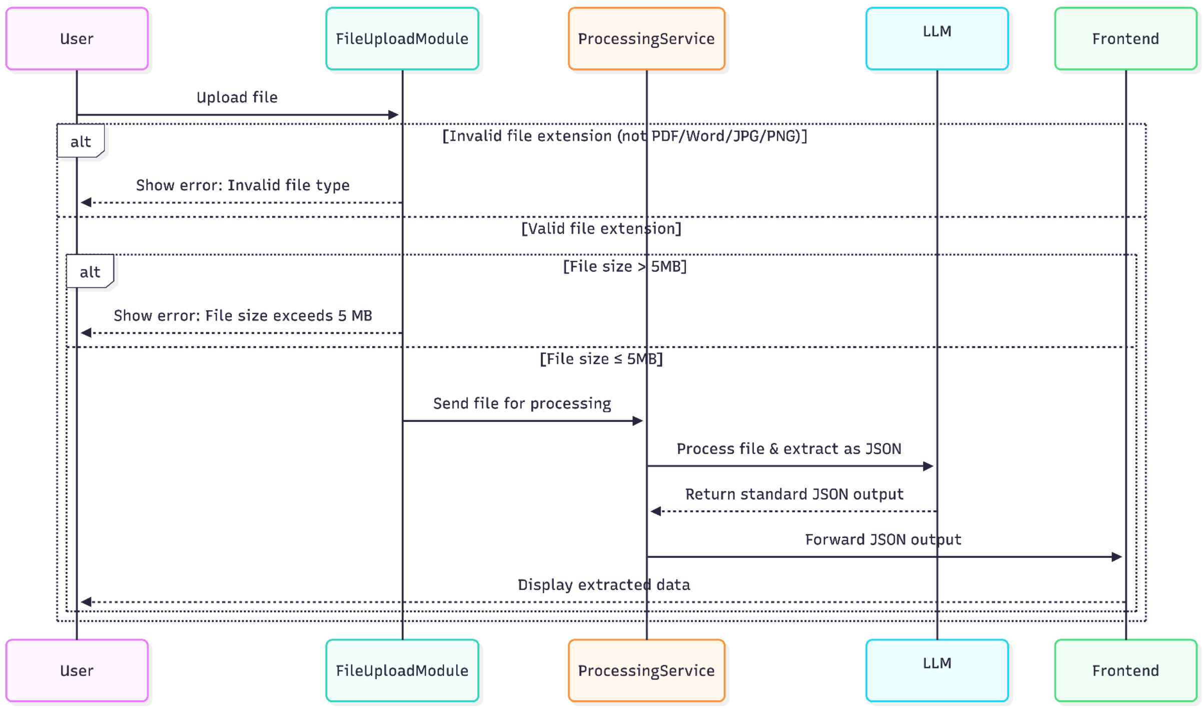
Learning Yogi Assessment

Deliverables

End to End Workflow

File upload → Processing → Extraction → Frontend Display.

Sequence Diagram – Suggested Tool - **MermaidChart.com**



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.

Development / IDE
Google Antigravity / VS Code + Claude

File ingestion and preprocessing.
I have used Nodejs+ Express JS as backend for uploading the File and Preprocessing

OCR or Document Parsing (Tables and Images)
DeepSeek OCR / GPT-4V or GPT-4o / Gemini Pro
Implemented LLM factory Pattern to choose from various LLM's

LLM Integration
Factory Pattern - Can use OpenAI, Anthropic , Deep Seek , Gemini Pro

Backend Orchestration
Node JS + Express JS using TypeScript

Learning Yogi – Assignment – Time Table Parser

Front end Rendering : React

Database Schema
PostgreSQL with JSONB for flexible timetable data

Additional information on your LLM Integration strategy:

I use Multi-stage Approach
Stage 1 : Document Understanding
Use LLM to validate extracted data for logical consistency

- Check time overlaps, duration reasonableness
- Confidence scoring for each extraction

Stage 2 : Validation and Correction
Stage 3 : Fallback Strategy

If confidence < 70%, using vision model , Manual review queue for ambiguous cases

- What part of the pipeline uses it?
Processing Service uses the LLM Integration

- What is the prompt strategy?

```
# Prompt strategy for initial extraction
system_prompt = """You are a timetable extraction expert. Extract structured timetable data from the
provided text.

Output Format (JSON):
{
  "days": ["Monday", "Tuesday", ...],
  "timeblocks": [
    {
      "day": "Monday",
      "subject": "Mathematics",
      "start_time": "09:00",
      "end_time": "10:00",
      "teacher": "optional",
      "notes": "optional"
    }
  ]
}

Rules:
1. Preserve original subject names exactly
2. Infer missing times from context
3. Handle varied formats (12h/24h)
4. Extract all additional notes
"""
```

- How do you ensure accuracy and reproducibility?
- Error handling & fallbacks: - How do you handle bad uploads, ambiguous data, or missing fields?

- **Bad uploads:** File validation (type, size)
- **Ambiguous data:** Confidence scoring + manual review queue
- **Missing fields:** LLM inference with explicit uncertainty markers
- **OCR failures:** Fallback to vision models

- How will you ensure the system is flexible, for possible future updates and needs?

I will implement Plugin architecture for new file formats

- Webhook support for processing completion
- Template library for common timetable formats
- AI training feedback loop (human corrections → fine-tuning)

Prompt Used for Backend Development

You are an expert Node JS Developer with 10+ Years of experience in developing Backend API's for processing PDF / Images and extract text and return output as JSON
Your job is to 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.).
Implement Logic to verify the File TypeAllowed file Types are PDF / Images (JPG / PNG) / DOCX / DOC Files
Flie size is limited to 5 MB only and can be configured in future also
Use LLM Factory Pattern so that different LLM Models can used based on future availability
Use ZOD for Schemas
Use only Postgres Database for schema
Validate the Output with Other LLM like Claude and retrieve the confidence scores and verify that confidence less is greater of equal to 70%,
Accept only responses with more than 70% confidence score.
Document all the end points using OpenSwagger standards
- Returns a JSON response.

Add unit tests for verifying file types and file size to validated the uploaded file. File types to verify are PDF Images (PNG / JPG) Doc and docx file . and File upload size should be not more than 5 MB. Always update the README file whenever we implement any new features

Frontend Strategy

Stack: React 18, TypeScript, Tailwind CSS, React Query

Approach: Component-based architecture with clear separation of concerns

State Management:

1. Server state handled via React Query
2. Minimal global UI state via Context API

Responsiveness: Mobile-first; day view on mobile, week view on larger screens

Reusability: Modular components and custom hooks for logic reuse

Project Structure:

```
src/
├── components/
│   ├── timetable/      # Feature-based UI components
│   └── common/         # Shared UI elements
├── hooks/                      # Reusable logic and data fetching
├── context/            # Global UI state (minimal)
├── utils/              # Helper functions
├── types/              # TypeScript types
└── App.tsx             # Entry Point
```