

■ Build a complete backend in Node.js + Express + TypeScript for “MindDecoder” — an AI-powered academic wellness app

■ Requirements:

- Environment: Replit (Node.js + TypeScript)
- Folder structure:
 - /server → backend code (index.ts, routes, ml.ts)
 - /server/ml → your ML model files (model.pkl or predictor.py)
 - /data → optional local database (SQLite/Postgres)
 - /.env → stores API keys and DB URL

■■ Backend Endpoints:

1■■ POST /api/mocktest

→ Integrate OpenAI GPT-4 API to generate mock test questions.

- Input (JSON): { "subject": "Math", "difficulty": "Medium" }
- Output (JSON):

```
{
  "questions": [
    { "q": "What is the derivative of sin(x)?", "a": "cos(x)" },
    ...
  ]
}
```

2■■ POST /api/predictor

→ Upload and analyze a PDF (like a question paper or notes) using your ML model.

- Accept a file upload via multer.
- Use pdf-parse to extract text from the PDF.
- Pass extracted text to your ML model (in /server/ml/predictor.py or /server/ml/model.js).
- Return model predictions as JSON.
- Example response:

```
{
  "predicted_topics": [
    { "topic": "Recursion", "probability": 0.92 },
    { "topic": "Dynamic Programming", "probability": 0.87 }
  ],
  "recommendation": "Focus on Recursion and DP next."
}
```

3■■ (Optional) /api/wellness

→ Collect focus and mood metrics to store for future analytics.

■ ML Integration:

- If your model is Python-based:
 - Use Node's `child_process.spawn` to run the Python script.
 - Pass the extracted PDF text as input.
 - Parse and send the Python model output back to the frontend.
- If your model is JS/TS-based:
 - Import directly (e.g., `import { predict } from './ml/model';`) and use it inside the /api/predictor route.

■ Dependencies to Install:

```
npm install express cors dotenv openai multer pdf-parse child_process cross-env
npm install -D typescript tsx @types/express @types/node @types/multer
```

■ Example .env file:

```
OPENAI_API_KEY=your_openai_api_key_here
DATABASE_URL=file:./data/prepnova.db
PORT=5000
```

■ Expected Flow:

- Frontend sends POST /api/mocktest → GPT API returns generated questions.
- Frontend uploads a PDF to POST /api/predictor → server extracts text → ML model predicts topics → returns insight

■ Start Command:

npm run dev → cross-env NODE_ENV=development tsx server/index.ts

■ Output:

Backend runs two major intelligent systems —

- GPT-powered content generation (Mock Tests)
 - ML-powered topic prediction from PDFs
- working seamlessly with your existing frontend.