Full Stack Developer Assignment - Smart Todo List with AI

To secure the full-time position, you are required to complete this technical assignment. Early submissions are given preference.

In this task, you are required to create a full-stack web application with AI integration for intelligent task management.

Objective:

The goal of this assignment is to build a Smart Todo List application where users can manage their tasks with AI-powered features like task prioritization, deadline suggestions, and context-aware recommendations. The system should use daily context (messages, emails, notes) to provide intelligent task management suggestions.

Backend (Django REST Framework)

GET APIs:

- Retrieve all tasks from the database
- Get task categories/tags
- Fetch daily context entries

POST APIs:

- Create new tasks
- Add daily context (messages, emails, notes)
- Get AI-powered task suggestions and prioritization

AI Integration Module

Create a Python module that handles AI-powered task management:

****Extra marks for intelligent context analysis, smart categorization, and personalized recommendations.**

The system should:

- **Context Processing**: Analyze daily context (WhatsApp messages, emails, notes)

- **Task Prioritization**: Use AI to rank tasks based on urgency and context
- **Deadline Suggestions**: Recommend realistic deadlines based on task complexity
- **Smart Categorization**: Auto-suggest task categories and tags
- **Task Enhancement**: Improve task descriptions with context-aware details

Input Parameters for AI Features:

- Task details (title, description, category)
- Daily context data
- User preferences (optional)
- Current task load (optional)

AI Pipeline Implementation:

The system should implement AI features that:

- Process daily context to understand user's schedule and priorities
- Generate task priority scores based on context analysis
- Suggest optimal deadlines considering workload and context
- Provide enhanced task descriptions with relevant context
- Recommend task categories and tags automatically

Frontend (NextJS with Tailwind CSS)

User Interface:

The frontend should be developed using NextJS, styled with Tailwind CSS.

Required Pages:

1. **Dashboard/Task List**:
    - Display all tasks with priority indicators
    - Filter by categories, status, priority
    - Quick add task functionality
2. **Task Management Interface**:
    - Create/edit tasks with AI suggestions
    - AI-powered deadline recommendations
    - Context-aware task descriptions

3. **Context Input Page**:
    - Daily context input (messages, emails, notes)
    - Context history view

Database Schema Hints:

Design your database with the following tables structure:

Tasks table: Store task details including title, description, category, priority score, deadline, status, created/updated timestamps.

Context entries table: Store daily context data with content, source type (WhatsApp, email, notes), timestamps, and processed insights.

Categories table: Store task categories and tags with usage frequency.

Tech Stack Requirements:

- **Backend**: Django REST Framework, Python
- **Database**: Supabase (PostgreSQL)
- **Frontend**: ReactJS with Tailwind CSS
- **AI Integration**: OpenAI API, Claude API, Gemini API or **LM Studio** (recommended)
- **Storage**: Supabase storage for any file uploads

AI Integration Options:

Option 1: Use external APIs (OpenAI, Anthropic Claude)
Option 2: Use LM Studio for local LLM hosting (recommended alternative)

LM Studio allows you to run language models locally without requiring external API keys. Download and set up LM Studio with models like Llama or Mistral for task analysis and suggestions. This provides a cost-effective solution and ensures data privacy.

Deadline

7 July, 2025, Monday, 11:55 PM

Submission Format

- Create a GitHub repository and add your code to it.

- In the README, add:
  - Screenshots of the UI you have created
  - Setup instructions for running the application
  - API documentation
  - Sample tasks and AI suggestions from your system
- Include a requirements.txt file with all dependencies
- Add sample context data for testing
- Fill your repo link in this form: https://forms.gle/CUv48PxFwG59RCEt9
- Make sure the code is neat and readable with proper comments and OOPS implementation.

Evaluation Criteria:

- **Functionality** (40%): Working AI features, accurate prioritization, context integration
- **Code Quality** (25%): Clean, readable, well-structured code
- **UI/UX** (20%): User-friendly interface, responsive design
- **Innovation** (15%): Creative AI features, smart context utilization

Bonus Points (Optional):

- Advanced context analysis (sentiment analysis, keyword extraction)
- Task scheduling suggestions based on context
- Integration with calendar or time-blocking features
- Export/import functionality for tasks
- Dark mode toggle

For any doubts, feel free to drop a mail at devgods99@gmail.com
We'll try to respond as soon as possible!

In case you're not able to complete it within the deadline, do submit the code even if a part of it doesn't work. The effort and skills you demonstrate through your code matter.

Note: An early submission will give you an advantage over others.

Helpful Resources:

- LM Studio: https://lmstudio.ai/ (for local LLM hosting)
- Supabase Documentation: https://supabase.com/docs
- Django REST Framework: https://www.django-rest-framework.org/
- OpenAI API: https://platform.openai.com/docs
- React Hooks: https://reactjs.org/docs/hooks-intro.html

Note: While the use of AI tools is acceptable, candidates are expected to demonstrate clear understanding of the code and the ability to adapt or modify it as needed.