Clerk Project Scope

Project Objective:

We plan to develop Clerk, an AI-assisted task organization and scheduling system designed to convert unstructured user input into structured, actionable tasks. Clerk *will* accept input in multiple forms, including text, email messages, and audio notes. All inputs will be normalized into plain text and stored with relevant metadata such as timestamps and source type.

Rather than functioning as a standard conversational chatbot, Clerk will use a multi-stage processing pipeline with feedback loops that allow individual stages to be refined independently. The system will analyze user input to extract tasks, deadlines, and contextual information while assigning confidence scores to determine whether an item represents a confirmed obligation or an inferred task. Clerk will also incorporate light adaptive behavior by learning from user interactions over time, such as task completion patterns or preferred work hours, to improve future scheduling recommendations. The goal of this project is to help users manage responsibilities more effectively while minimizing manual task entry and scheduling effort.

Deliverable:

The progress of this project will be tracked through progress reports, phases of big goals, a midterm report, and a final report.

Milestones:

Phase 1 - Foundation (Weeks 1-3):

- Backend: Learn FastAPI, Pydantic, and SQL
- Frontend: Learn basic HTML and CSS
- Sync Github and VS code
- Design a SQL schema that links raw inputs to structured task outputs
- Build a cleaner module to format raw text into JSON

Phase 2 - MVP Core (Weeks 3-8):

- Implement a manual text drop/paste interface for immediate processing
- Develop an AI extraction layer using GPT-4o Mini and regex to map text into structured JSON (tasks, dates, assignees)
- Implement logic to convert relative phrases (e.g., “by Tuesday”) into hard ISO timestamps
- Ensure all processed text inputs successfully save to the SQLite database

Phase 3 - Audio & Logic (Weeks 9-11)

- Integrate OpenAI Whisper for audio note transcription
- Build a browser-based recording interface using the Media Recorder API
- Implement a system to assign confidence values (0-100) to distinguish between “confirmed” obligations and “inferred” tasks

Phase 4 - Scheduling & UI (Weeks 12-14)

- Develop logic to order tasks by urgency and identify available time slots based on user's “preferred work hours”
- Build a modern dashboard to view, edit, and approve suggested schedules
    - Instead of a full grid calendar, build a “Daily Agenda” view. Basically displays tasks in a vertical timeline, broken into time slots
- Implement simple visual notifications in the app to alert the user when a new task has been successfully parsed from a background process
- Fully connect React frontend to the FastAPI backend

Phase 5 - Refinement & Expansion (Weeks 14-17)

- Implement light statistical aggregation to learn user patterns
- Host the application using Render for live testing
- Attempt Google Calendar API integration for two-way syncing if core features are stable
- Cross-platform notifications
- Create final presentation and comprehensive README documentation

In Scope:

- Linear, timeline-based display of tasks for a given day
- An interface for the user to confirm, edit, or reject AI-extracted tasks
- Processing direct text inputs, .txt files, and audio transcriptions.
- Extracting deadlines and converting relative dates to timestamps.
- Assigning confidence scores to tasks based on language clarity.
- Developing a web-based UI for task approval and scheduling visualization.
- Learning user work-hour preferences for better task placement.

Out of Scope:

- Interactive Grid Calendar
- Push notifications
- Complex Integrations: Slack and Gmail thread monitoring are excluded to avoid "unknown variables" (unless tackled as a final reach goal).

- Autonomous Actions: The system will not book appointments or send emails without user approval.
- Sensitive Data: No processing of financial or medical records.
- Mobile App: Development is strictly limited to a responsive web application.

Reach Goals:

- Exporting the “Agenda View” to Google Calendar and enabling a Google Calendar Two-Way Sync.
- Cross-platform notification system (Email or SMS alerts via Twilio/SendGrid).
- Custom branding and logo design.