Clerk
Computer Science Senior Capstone
Chris Nolan, Michael Thomas, John Sedoriosa

This project will be developed as an AI-assisted system that takes unstructured user inputs and converts them into structured, actionable items. Rather than functioning as a standard conversational chatbot, there will be a multi-stage pipeline with feedback loops, allowing each stage to be improved independently. User input will be taken in two primary forms: audio notes, email messages, and text input. If audio files are used it will be transcribed into text using a speech-to-text service. Emails will be ingested through permission-based integrations or via structured imports such as .eml or mailbox exports. All inputs are normalized into plain text and stored as source records along with metadata such as timestamps and origin type.

The system will incorporate light adaptive behavior by learning from user interactions over time. When users modify task durations, reschedule events, or complete certain types of work at particular times of day, it will record these patterns and adjust future recommendations accordingly. So, for example, if a user regularly completes a task at a certain time weekly/daily, it will send a reminder in case the user forgot. The learning process will rely on simple statistical aggregation ensuring we can get it done within the project timeline. Temporal information is resolved using a dedicated time and date normalization process that will handle both absolute and relative phrases such as “tomorrow” or “by Friday” and interpret them using the timestamp of the original input and user's configured timezone. When a date or time cannot be resolved with sufficient confidence, the system marks the information as ambiguous rather than guessing. These ambiguities will cause the system to request additional information from the user in a manner.

Each task is assigned a confidence score reflecting the system’s certainty that the item reflects a genuine obligation. The score is calculated by combining multiple signals, including the language used, clarity of detected temporal information, and the presence of an action-object structure. Items with lower confidence are labeled as inferred rather than confirmed, allowing the system to differentiate between firm commitment and tentative intentions while still preserving potentially important information.

Scheduling recommendations are generated using a heuristic approach. The scheduler considers confirmed tasks, calendar availability, and user-defined preferences for example preferred work hours. Tasks are ordered by urgency and priority, then placed into available time slots using a best-fit strategy. Rather than committing changes the system will present suggested schedules for user review.

Languages/Software Needed:

Languages:

- Python
- TypeScript (if we get to mobile)
- SQL

Implicit:

- HTML
- CSS

Software:

- VS Code
- Render
- React
- FastAPI
- Pydantic
- SQLite
- OpenAI API
    - Model: GPT-40 Mini
- Dateparser or python dateutil
- Browser Media Recorder API
- OpenAI Whisper
- Gmail API and Google Calendar API
- GitHub

Roles:

- John - backend
- Chris - frontend
- MT - both/lead coder