# Tech Stack & Database Schema

## 1. Technology Stack

### **Frontend (The Interface)**

* **Framework:** **React.js (Vite)** with **TypeScript**.
  + *Why:* Type safety is critical for handling complex Task/Schedule data structures.
* **State Management:** **Zustand**.
  + *Why:* Lightweight and perfect for managing local calendar state (drag-and-drop updates).
* **UI Library:** **Shadcn/UI** + **Tailwind CSS**.
  + *Why:* Provides accessible, pre-built components (Modals, Popovers) for rapid UI development.
* **Calendar Engine:** **React-Big-Calendar**.
  + *Why:* Handles the complex math of rendering Week/Day views.
* **Visualization:** **Recharts**.
  + *Why:* Needed for the "Burnout Meter" and "Planned vs Actual" analytics.

### **Backend (The Brain)**

* **Framework:** **FastAPI (Python)**.
  + *Why:* Native support for AI libraries, async performance, and auto-generated API docs.
* **Authentication:** **OAuth2 + JWT (JSON Web Tokens)**.
  + *Why:* Stateless, secure session management.
* **Task Scheduling:** **APScheduler**.
  + *Why:* Runs the background "Reflexion Agent" every 3-5 days to summarize logs.
* **AI & ML:** **LangChain** (Logic) + **Scikit-Learn** (Prediction).
  + *Why:* LangChain manages LLM prompts; Scikit-Learn handles simple time-estimation regression.

### **Database (The Memory)**

* **Database:** **PostgreSQL**.
  + *Why:* The only database that handles **Relational Data** (Tasks/Users) and **Unstructured Data** (JSONB for AI Memory) effectively in one place.
* **ORM:** **SQLAlchemy**.

## 3. Database Schema (SQL)

This schema is designed for **PostgreSQL**. It includes JSONB columns for AI flexibility and a dedicated courses table for subject management.

### **A. Identity & Profile**

SQL

-- 1. Users (Auth)  
CREATE TABLE users (  
 id SERIAL PRIMARY KEY,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 username VARCHAR(50) UNIQUE NOT NULL,  
 password\_hash VARCHAR(255) NOT NULL,  
 google\_refresh\_token VARCHAR(255), -- Stores long-term GCal token  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
-- 2. User Profiles (AI Context)  
CREATE TABLE user\_profiles (  
 user\_id INTEGER PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,  
 full\_name VARCHAR(100),  
 major VARCHAR(100),  
 university VARCHAR(100),  
   
 -- The "Digital Twin" Memory  
 current\_archetype VARCHAR(50) DEFAULT 'Unclassified', -- e.g., 'The Night Owl'  
   
 -- Flexible "Cold Start" & Learned Traits  
 -- Stores: Chronotype, Math Confidence, Duration Multipliers  
 onboarding\_data JSONB DEFAULT '{}'   
);

### **B. Academics & Schedule**

SQL

-- 3. Courses (Subjects)  
-- Users define these (e.g., "Calculus", "History") to link tasks to memory.  
CREATE TABLE courses (  
 id SERIAL PRIMARY KEY,  
 user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,  
 name VARCHAR(100) NOT NULL,   
 color\_code VARCHAR(7), -- Hex Color for Calendar UI  
 default\_priority VARCHAR(20) DEFAULT 'Medium',  
 UNIQUE(user\_id, name)  
);  
  
-- 4. Fixed Schedule (Hard Constraints)  
CREATE TABLE fixed\_slots (  
 id SERIAL PRIMARY KEY,  
 user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,  
 day\_of\_week VARCHAR(10) NOT NULL, -- 'Monday', 'Tuesday'  
 start\_time TIME NOT NULL,  
 end\_time TIME NOT NULL,  
 label VARCHAR(100), -- e.g., "Chemistry Lab"  
 is\_google\_event BOOLEAN DEFAULT FALSE,  
 google\_event\_id VARCHAR -- To sync updates back to Google  
);

### **C. Task Engine**

SQL

-- 5. Tasks (The Planner)  
CREATE TABLE tasks (  
 id SERIAL PRIMARY KEY,  
 user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,  
 course\_id INTEGER REFERENCES courses(id) ON DELETE SET NULL, -- Link to Subject  
   
 title VARCHAR(255) NOT NULL,  
 description TEXT,  
 category VARCHAR(50), -- 'Assignment', 'Exam', 'Project'  
 priority VARCHAR(20) DEFAULT 'Medium',  
   
 -- Scheduling Logic  
 deadline TIMESTAMP,  
 estimated\_duration\_mins INTEGER, -- AI Prediction  
 is\_high\_burden BOOLEAN DEFAULT FALSE,  
 status VARCHAR(20) DEFAULT 'Pending',  
   
 -- Recursive Decomposition (Sub-tasks)  
 parent\_task\_id INTEGER REFERENCES tasks(id) ON DELETE SET NULL,  
   
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

### **D. Feedback Loop (AI Memory)**

SQL

-- 6. Task Logs (The Reality)  
CREATE TABLE task\_logs (  
 id SERIAL PRIMARY KEY,  
 task\_id INTEGER REFERENCES tasks(id) ON DELETE CASCADE,  
 user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,  
   
 completion\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 actual\_duration\_mins INTEGER, -- Used to calculate "Planning Fallacy"  
 drain\_intensity INTEGER CHECK (drain\_intensity BETWEEN 1 AND 5), -- Burnout Calc  
   
 mood\_note TEXT,  
 ai\_feedback\_tags JSONB -- e.g., ["procrastinated", "distracted"]  
);  
  
-- 7. AI Rolling Summaries (Reflexion)  
CREATE TABLE ai\_memories (  
 id SERIAL PRIMARY KEY,  
 user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,  
 generated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
   
 -- The text summary for the user to read  
 summary\_text TEXT,   
   
 -- The structured data for the System to read (updates the profile)  
 updated\_traits JSONB   
);

## 4. JSON Data Structures (AI Memory)

To ensure the "Memory" is readable by code (not just LLMs), we enforce the following JSON structures in the onboarding\_data column.

**Storage Location:** user\_profiles.onboarding\_data

JSON

{  
 "global\_settings": {  
 "chronotype": "night\_owl", // Affects scheduling logic (pushes tasks later)  
 "base\_energy\_level": 7 // 1-10 scale  
 },  
 "subject\_modifiers": {  
 "55": { // "55" is the Course ID for "Calculus"  
 "confidence\_score": 3,  
 "duration\_multiplier": 1.5, // System multiplies User Input \* 1.5  
 "drain\_rate": 5 // High drain = Don't schedule back-to-back  
 },  
 "56": { // "56" is Course ID for "History"  
 "confidence\_score": 9,  
 "duration\_multiplier": 0.9, // User usually over-estimates this  
 "drain\_rate": 2  
 }  
 }  
}