# 

# 

# Jeeny Smart Task Manager: Architecture & Design Documentation

### Author: Muhammad Affan

### Date: 6th July 2025

Table of Contents

[ 1](#_Toc202726712)

[Jeeny Smart Task Manager: Architecture & Design Documentation 1](#_Toc202726713)

[Author: Muhammad Affan 1](#_Toc202726714)

[Date: 6th July 2025 1](#_Toc202726715)

[1. Tech Stack Analysis 3](#_Toc202726716)

[Current Implementation Stack 3](#_Toc202726717)

[2. Architecture Rationale 3](#_Toc202726718)

[a) Next.js 14+ with App Router 3](#_Toc202726719)

[b) Supabase (PostgreSQL + BaaS) 3](#_Toc202726720)

[c) Typescript 3](#_Toc202726721)

[d) Tailwind CSS 3](#_Toc202726722)

[3. High-Level Architecture Diagram 4](#_Toc202726723)

[4. Database Schema & Design 4](#_Toc202726724)

[5. Detailed Schema 5](#_Toc202726725)

[6. Key Schema Features 6](#_Toc202726726)

[a) Security 6](#_Toc202726727)

[b) Performance 6](#_Toc202726728)

[c) Flexibility 6](#_Toc202726729)

[7. Why Current Stack is Optimal for This Project 7](#_Toc202726730)

## Tech Stack Analysis

### Current Implementation Stack

* Frontend Framework: Next.js 14+ with App Router
* Runtime: Node.js
* Database: Supabase (PostgreSQL)
* Authentication: Supabase Auth
* Styling: Tailwind CSS
* Real-time Features: Supabase Realtime
* Deployment: Vercel
* Language: Typescript

## Architecture Rationale

### Next.js 14+ with App Router

* Full-Stack Integration: Combines frontend and backend in unified codebase
* Server-Side Rendering: Enhanced SEO and faster initial page loads
* Built-in API Routes: Eliminates need for separate backend server
* File-based Routing: Intuitive and maintainable route organization
* Typescript Integration: Native type safety support
* Performance Optimizations: Automatic code splitting, image optimization

### Supabase (PostgreSQL + BaaS)

* Managed PostgreSQL: Production-ready database without DevOps overhead
* Integrated Authentication: OAuth, JWT, and row-level security built-in
* Real-time Subscriptions: WebSocket-based live updates for collaboration
* Auto-generated APIs: RESTful and GraphQL endpoints from database schema
* Row-Level Security: Database-enforced user data isolation
* Cost-Effective: Generous free tier for development and scaling

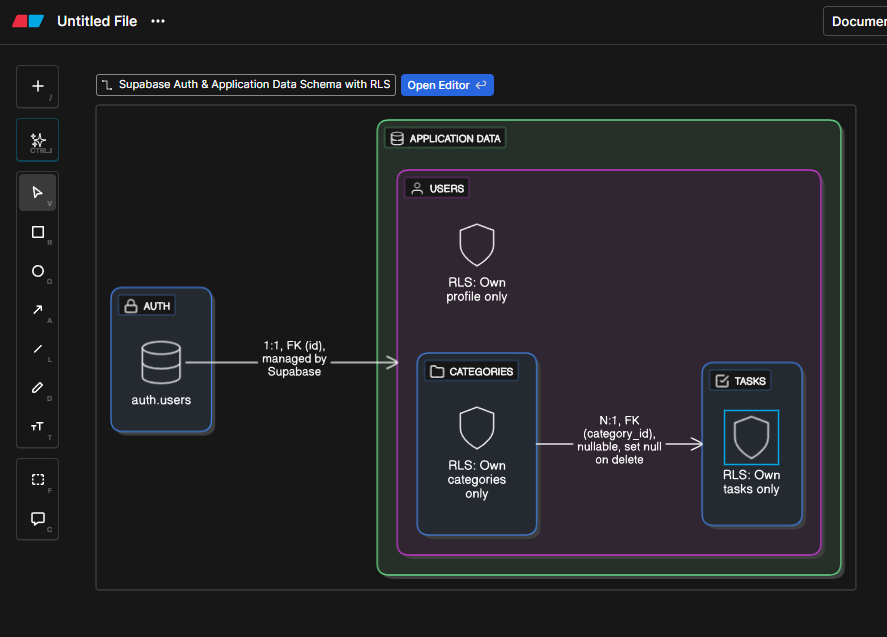
### Typescript

* Compile-time Type Safety: Catches errors before runtime
* Enhanced Developer Experience: Superior IDE support and autocomplete
* Self-documenting Code: Interfaces serve as living documentation
* Team Collaboration: Enforces consistent data structures across team

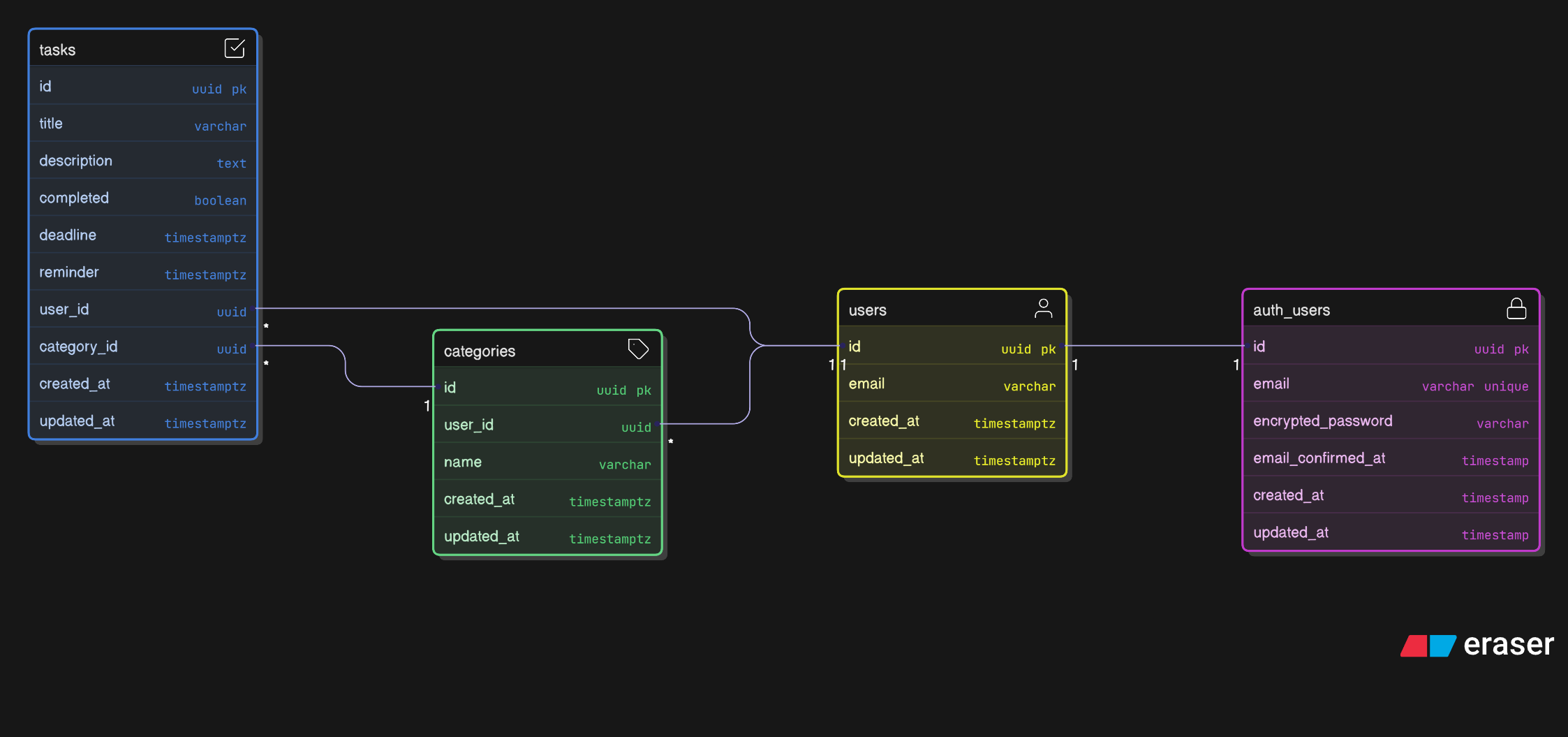
### Tailwind CSS

* Utility-First Approach: Rapid UI development and prototyping
* Responsive Design: Mobile-first methodology built-in
* Theme Support: Easy dark/light mode implementation
* Optimized Bundle: Only used utilities included in production
* Design System: Consistent spacing, typography, and color schemes

## High-Level Architecture Diagram



## Database Schema & Design



## Detailed Schema

* auth.users (Managed by Supabase)

*-- Built-in Supabase auth table*

auth.users (

  id UUID PRIMARY KEY,

  email VARCHAR UNIQUE,

  encrypted\_password VARCHAR,

  email\_confirmed\_at TIMESTAMP,

  created\_at TIMESTAMP,

  updated\_at TIMESTAMP,

*-- ... other auth fields*

)

* users (Application user profile)

CREATE TABLE users (

  id UUID PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE,

  email VARCHAR NOT NULL,

  created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

*-- RLS Policies*

ALTER TABLE users ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Users can view own profile" ON users FOR SELECT USING (auth.uid() = id);

CREATE POLICY "Users can update own profile" ON users FOR UPDATE USING (auth.uid() = id);

* categories

CREATE TABLE categories (

  id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

  name VARCHAR(100) NOT NULL,

  user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

  created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  UNIQUE(name, user\_id) *-- Unique category name per user*

);

*-- RLS Policies*

ALTER TABLE categories ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Users can manage own categories" ON categories

  FOR ALL USING (auth.uid() = user\_id);

* tasks

CREATE TABLE tasks (

  id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

  title VARCHAR(200) NOT NULL,

  description TEXT,

  completed BOOLEAN DEFAULT FALSE,

  deadline TIMESTAMP WITH TIME ZONE,

  reminder TIMESTAMP WITH TIME ZONE,

  user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

  category\_id UUID REFERENCES categories(id) ON DELETE SET NULL,

  created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

*-- Indexes for performance*

CREATE INDEX idx\_tasks\_user\_id ON tasks(user\_id);

CREATE INDEX idx\_tasks\_category\_id ON tasks(category\_id);

CREATE INDEX idx\_tasks\_reminder ON tasks(reminder) WHERE reminder IS NOT NULL;

CREATE INDEX idx\_tasks\_deadline ON tasks(deadline) WHERE deadline IS NOT NULL;

*-- RLS Policies*

ALTER TABLE tasks ENABLE ROW LEVEL SECURITY;

CREATE POLICY "Users can manage own tasks" ON tasks

  FOR ALL USING (auth.uid() = user\_id);

## Key Schema Features

### Security

* Row-Level Security (RLS): Database-enforced user isolation
* UUID Primary Keys: Non-guessable, secure identifiers
* Foreign Key Constraints: Data integrity

### Performance

* Strategic Indexes: Fast queries on user\_id, dates, categories
* Timestamp with Time Zone: Proper timezone handling
* Efficient Relationships: Normalized structure

### Flexibility

* Soft Category Deletion: Tasks retain when category deleted
* Nullable Fields: Optional deadlines and reminders
* Extensible: Easy to add new fields (priority, tags, etc.)

## Why Current Stack is Optimal for This Project

* Rapid Development: Full-stack in single codebase
* Cost Effective: Generous free tiers
* Scalable: Can handle significant growth
* Maintainable: Simple architecture, good DX
* Production Ready: Built-in optimizations and security
* The chosen stack perfectly balances development speed, maintainability, performance, and cost for a task management application that needs real-time features and can scale to thousands of users.

# Thank you all!