TaskFlow - Architecture and Design Document

1. Tech Stack Selection

Frontend Technologies

- **React 18.x** - Core frontend framework
- **Material-UI (MUI)** - Component library including:
- **D3.js** - Data visualization library for graph rendering
- **Framer Motion** - Animation library

Backend Services

- **Firebase Authentication** - User authentication service
- **Firestore** - NoSQL database service

Deployment & Hosting

- **Vercel** - Frontend hosting and deployment platform

2. Tech Stack Justification

React was chosen for its component-based architecture and because React's virtual DOM provides optimal performance for frequent UI updates required in task management applications, particularly for drag-and-drop on our KanBan board.

Material-UI (MUI) was chosen for its comprehensive component library and seamless integration with React.

D3.js was selected for the graph visualization feature due to its unparalleled flexibility in creating custom SVG-based visualizations.

Framer Motion was chosen enabling smooth transitions and engaging user interactions throughout the application interface.

Backend and Data Management

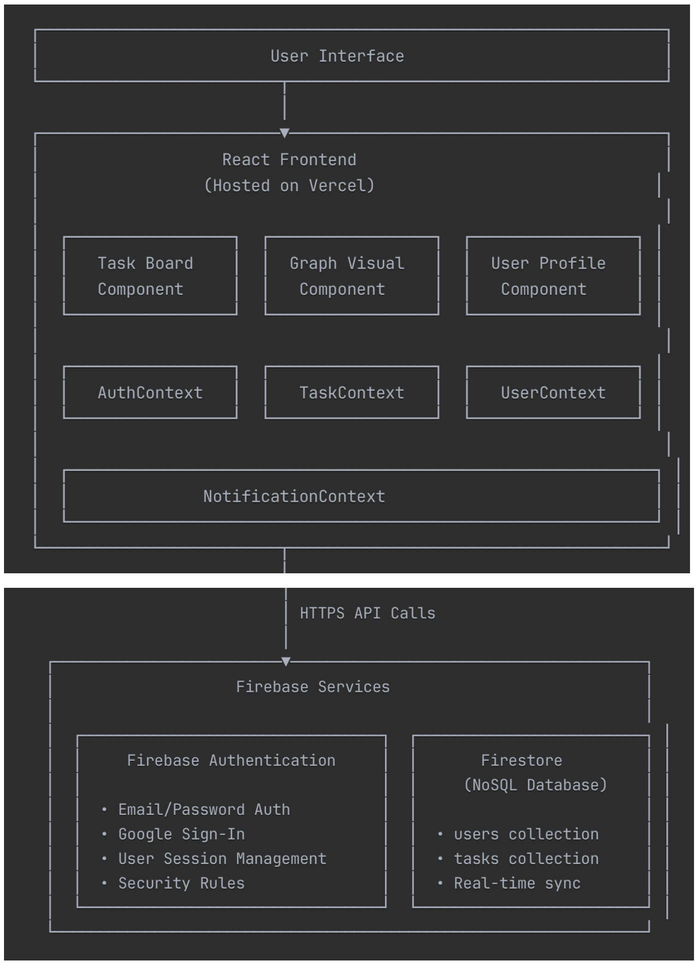
Firebase Authentication was selected for its comprehensive authentication solution that supports multiple sign-in methods (email/password, Google Sign-In) with minimal configuration.

Firestore was chosen for its real-time capabilities, automatic scaling, and serverless architecture. As a NoSQL document database, Firestore provides flexible data modeling suitable for the varying task structures while offering real-time synchronization essential for collaborative features and instant UI updates.

Deployment Platform

Vercel was selected for its optimized React application deployment, automatic CI/CD pipeline, and global CDN distribution. Vercel's integration with Git repositories enables seamless deployment workflows and provides excellent performance optimization for static and dynamic content.

3. High-Level Architecture Diagram



Data Flow Description

1. **User Authentication Flow**: Users authenticate through Firebase Auth, which establishes a secure session and provides user credentials to the React frontend.
2. **Task Management Flow**: The TaskContext manages all task-related operations, communicating with Firestore to perform CRUD operations and maintain real-time synchronization.
3. **User Profile Flow**: The UserContext handles user profile data, storing and retrieving user preferences and avatar configurations from Firestore.
4. **Notification Flow**: The NotificationContext monitors task deadlines and generates client-side notifications based on task status and timing.

4. Database Schema

Firestore Collections Structure

Users Collection (`users`)

Collection Path: `/users`

Document Structure:

```
{
  "uid": "string",              // Firebase Auth unique identifier
  "displayName": "string",      // User's display name
  "email": "string",            // User's email address
  "avatarConfig": {             // Configuration for react-nice-avatar
  }
}
```

Document ID: Firebase Auth `uid`

Security Rules: Users can only read/write their own document

Tasks Collection (`tasks`)

Collection Path: `/tasks`

Document Structure:

```
{
  "userId": "string",           // Foreign key to users collection
  "title": "string",            // Task title (required)
  "description": "string",      // Task description (optional)
  "deadline": "timestamp",      // Due date and time
  "category": "string",        // Task category with emoji (e.g., "💼 Work")
  "tag": "string",             // User-defined tag (e.g., "#creative")
  "status": "string",          // Current status: "To Do" | "In Progress" | "Completed"
  "createdDate": "timestamp"   // Task creation timestamp
}
```

Document ID: Auto-generated by Firestore

Security Rules: Users can only access tasks where `userId` matches their Firebase Auth `uid`