Architecture

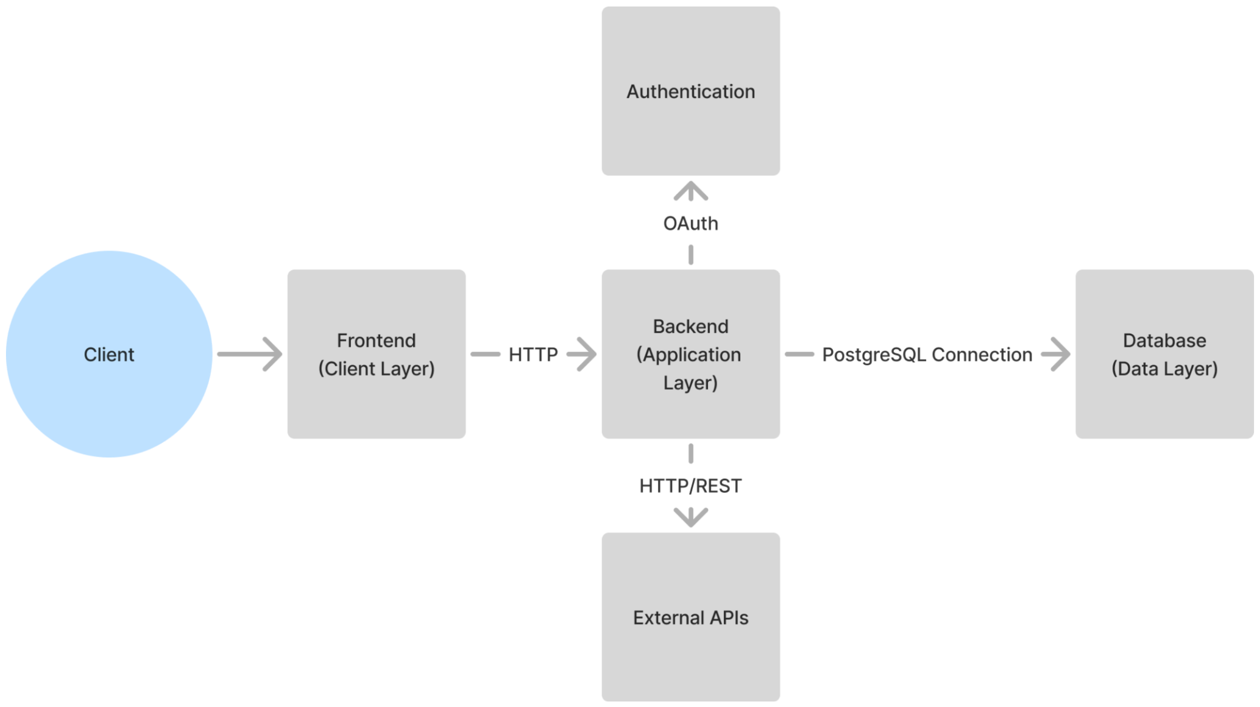
for Evalio: A Note-Taking Platform for mentors

Problem Background

The primary purpose of this project is to build a secure and interactive platform for evaluators and mentors of TalentSprint to make and manage notes, feedback, and student reviews. The platform would also have dedicated pages for all students, mentors, cohorts, and courses. Additionally, there would be a side panel containing a calendar, a to-do list, and an upcoming meetings section. The platform would also incorporate analytical data and insights that are accessible to users.

Architectural Design

Component & Connector Diagram



Component Description

Frontend (Client)

- Provides the user interface (UI) for mentors and instructors to take notes, organize sessions, and access productivity tools.
- Tech Stack: Next.js, Tailwind CSS, TypeScript.
- Functionality:
  - Displays UI components for note-taking, session management, analytics etc.
  - Sends user inputs to the backend via API requests.
  - Manages authentication by redirecting to login pages.
- Connectors: Communicates with the backend using HTTPS.

Backend (Server)

- Handles logic, processes user requests, and manages communication between the front end, database, authentication, and external APIs.
- Tech Stack: Node.js, Express.js (or Next.js API routes), Prisma ORM.
- Functionality::
  - Processes CRUD operations for notes, sessions, and user data.
  - Manages authentication and authorization.
  - Calls external APIs for email summarization, cloud storage, or analytics.
- Connectors:
  - Communicates with the front end via REST APIs over HTTPS.
  - Interacts with the database via SQL queries.
  - Authenticates users through OAuth2/OpenID with an authentication service.
  - Fetches data from external APIs (Zoom/Google/ChatGPT) via HTTPS requests.

Database

- Stores and retrieves application data persistently.
- Technologies: PostgreSQL/MySQL, Prisma ORM.
- Responsibilities:
  - Stores user accounts, notes, sessions, and permissions.
  - Supports transactions, indexing, and optimized queries.
  - Ensures data consistency and backup.
- Connectors:
  - Accessed by the backend via SQL queries

Authentication Service

- Role: Manages user authentication and authorization.
- Technologies: NextAuth.js, Firebase Auth, Auth0, or custom OAuth2 server.
- Responsibilities:
    - Handles login, logout, and session management.
    - Issues authentication tokens (JWT, OAuth2).
    - Ensures role-based access control (RBAC).
- Connectors:
    - Communicates with the backend via OAuth2/OpenID protocols over HTTPS.
    - Can also verify authentication with third-party providers (Google, Microsoft, etc.).

External APIs

- Provides additional functionality like summarization, cloud storage, analytics, or AI-based tools.
- Examples:
    - Google OAuth API (for authentication).
    - AI APIs (e.g: OpenAI, Hugging Face) (for summarization).
- Connectors:
    - Communicates with the backend via HTTPS requests.

Tech Stack

The platform will be built using the following technologies:

- **Backend:** Next.js
- **Frontend:** Next.js (React framework for server-side rendering and static site generation)
- **UI Components:** shadcn/ui (Customizable UI component library for React)
- **Styling:** Tailwind CSS (Utility-first CSS framework)
- **Database:** PostgreSQL (Reliable, Scalable, ACID-compliant)

Architecture Evaluation

- The architecture uses modular components (frontend, backend, authentication, database, external APIs), allowing new features (like analytics or reporting) to be added without affecting the core system.
- Using external APIs means that integrations (e.g., Google Calendar, Zoom) can be added without significantly modifying the backend.
- If the authentication mechanism needs to be replaced, only the authentication module will be affected without modifying the system.
- Tooltips and onboarding features can be implemented using UI libraries without modifying the backend.
- Keyboard shortcuts are handled on the front end without requiring changes to the back end.
- Bulk actions (e.g., deleting, pinning, flagging notes) are handled efficiently through batch processing APIs on the backend. Indexed database queries improve the performance of bulk updates.
- The system is optimized for ~30 mentors, and Next.js efficiently serves frontend requests. API calls are optimized to return responses in <500ms using indexing and caching.
- The backend and database can be containerized using Docker, enabling deployment in scalable environments.
- OAuth2 authentication (e.g., NextAuth.js, Firebase Auth) prevents unauthorized access.

The modular and scalable nature of the architecture ensures that it meets all the non-functional requirements efficiently

---

Prepared by Ananya Sachdev, Ananya Garg, Nishchaya Roy. 14-02-2025

.