**Application Interfaces Chart:**

**Application Interface Diagram** illustrates how core components interact via APIs and data flow channels.

At the user-facing layer, the **Frontend Web App**—built using React.js or equivalent frameworks—acts as the primary interface for students, tutors, and admins. All user interactions flow through the **Backend REST API**, implemented using Node.js/Express or Python/FastAPI, which handles business logic, authentication, and data orchestration.

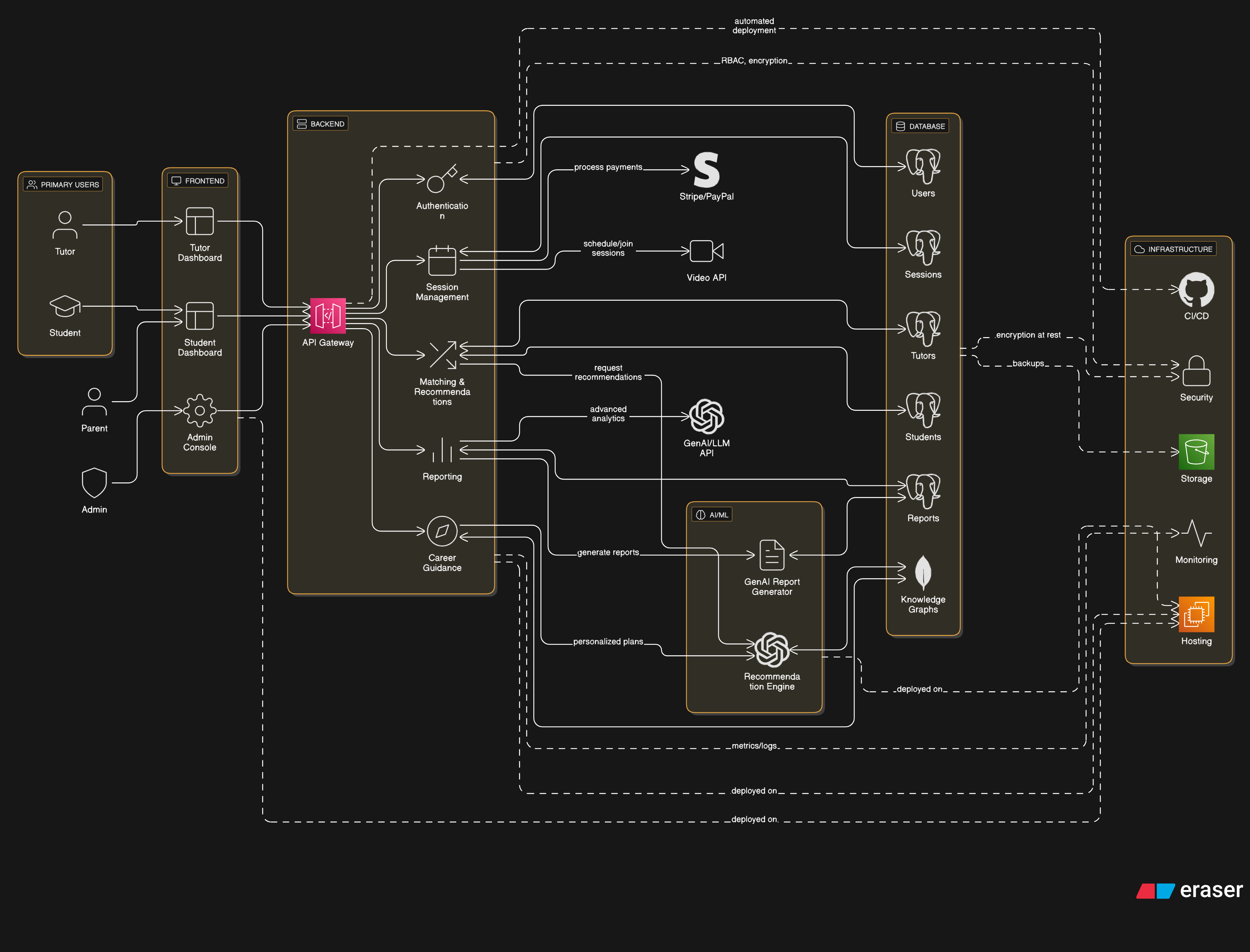
The **Database Layer**, using PostgreSQL or MongoDB, securely stores all application data including user profiles, session logs, knowledge graphs, and reports.

The **AI/ML Microservices**, built in Python and integrated with GenAI/LLM APIs such as OpenAI, process learning goals and behavior data to deliver personalized recommendations and automated planning.

The system is enhanced by integration with external APIs:

* **Video Conferencing APIs** (Zoom/WebRTC) to enable real-time sessions
* **Payment Gateways** (Stripe, PayPal) for secure transactions
* **LLM APIs** (OpenAI, Azure OpenAI) to enrich user experience with intelligent content and tutor matching

This layered, API-driven interface model allows seamless coordination across features like registration, tutoring sessions, assessments, and analytics while maintaining robust data security and service separation.



**Sample Database Schema**

Our Product is backed by a structured and scalable database schema designed to support user management, session tracking, and personalized learning features. The data model leverages normalized tables or NoSQL collections, depending on the underlying database (PostgreSQL or MongoDB).

At the core is the Users table/collection, which stores basic user information such as id, name, email, and role (student, tutor, or admin). Additional structured profile data is stored in a flexible profile\_data JSON field to support future extensibility.

The Tutors and Students tables link to the Users table via a foreign key (user\_id), maintaining separate domain-specific attributes. For tutors, fields like expertise, keywords, and sample\_videos capture their teaching credentials and media. For students, interests, goals, and a dynamic knowledge\_graph (stored as JSON) enable personalized learning and AI-driven recommendations.

The Sessions table tracks all scheduled or completed tutoring interactions. It links both students and tutors via student\_id and tutor\_id, and logs details such as session status, time, and notes.Finally, the Reports table serves as a historical record of learning assessments, storing report\_type, structured content in JSON format, and timestamps via created\_at.This schema supports modular development, personalized services, and efficient data retrieval across all platform operations.