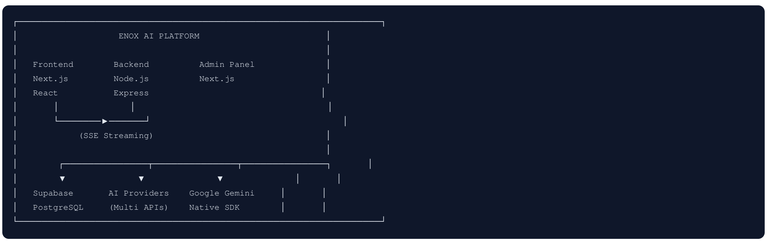
Enox AI — Technical Pipeline Report

System Architecture



1. Frontend → Backend Request Lifecycle

```
User → Auth Cache → API Request → SSE Stream → UI Render
```

Key Steps

- Auth token retrieved from local cache
- Message payload constructed with model and user input
- UI updates instantly and begins streaming

2. Backend Request Processing

Authentication Flow

```
Request → Token → Cache → Verify → User Context
```

Parallel Processing

Component	Purpose
Model	Validation
Agent	Prompt configuration
Rate Limit	Usage control
API Keys	Provider access

SSE Initialization

```
Request → Validate → Open SSE → Stream → Process
```

3. AI Provider Layer

Gemini uses real-time streaming, while other providers use standard APIs.

4. Streaming Protocol (SSE)

```
meta → thinking → content → done
```

Event	Purpose
meta	Initialize chat
thinking	Reasoning phase
content	Response tokens
done	Completion

5. Frontend Rendering

```
Buffer → Animation Loop → UI Render
```

- Smooth typewriter effect
- Minimal re-renders
- Adaptive speed rendering

6. Thinking System

```
AI → Backend → SSE → Frontend → UI
```

Thinking tokens are streamed before final output and displayed separately.

7. Performance

Standard

Stage	Time
Connection	~100ms
Processing	~300ms
First Token	~2s
Complete	~4s

Thinking Mode

Stage	Time
Thinking	5–8s
First Output	~6s
Complete	9–11s

8. Optimizations

#	Optimization	Impact
1	Gemini SDK	Faster streaming
2	SSE-first	Instant response
3	Parallel DB	Lower latency
4	Auth cache	No overhead
5	Reduced history	Faster AI
6	Async DB writes	Non-blocking

Conclusion

The Enox AI system is optimized for real-time performance using streaming architecture, parallel processing, and efficient frontend rendering, delivering a fast and scalable AI experience.