Glasspane — Technology Statement

Technical Architecture & Multi-Provider AI Integration
Your End-to-End Digital Health Dashboard

Multi-Provider AI Architecture

Glasspane is powered by a provider-agnostic AI layer supporting 8 LLM providers: OpenAI, Anthropic, Google Gemini, IBM watsonx, OpenRouter, AWS Bedrock, Ollama, and LM Studio. Each AI task can be independently configured to use any provider with per-task model assignment and automatic fallback chains. The integration follows a Backend-for-Frontend (BFF) pattern: the Next.js server acts as a secure proxy, constructing role-aware prompts enriched with real-time dashboard data before forwarding them to the configured provider. API credentials never reach the browser and every AI response is grounded in the customer's actual infrastructure metrics.

AI Task Endpoints

Glasspane provides 9 dedicated AI task endpoints, each independently configurable:

- AI Summary Generation (/api/ai/summary) — Role-tailored natural-language summaries of dashboard state.
- Streaming Chat (/api/ai/chat) — Multi-turn Q&A with real-time Server-Sent Events streaming and conversation history.
- Anomaly Detection (/api/ai/insights) — Identification of unusual metric patterns with structured anomaly objects and severity classification.
- Risk Briefing (/api/ai/risk-briefing) — AI-generated threat assessments with severity levels and recommended actions.
- SLA Risk Advisor (/api/ai/sla-risk) — SLA breach probability prediction with trend analysis.
- Cost Forecast (/api/ai/cost-forecast) — Spending trajectory analysis with budget impact projections.
- Capacity Planner (/api/ai/capacity-planner) — Resource utilization forecasts and threshold warnings.
- Root Cause Patterns (/api/ai/root-cause-patterns) — Incident pattern analysis identifying systemic issues.
- Change Impact (/api/ai/change-impact) — Deployment risk assessment evaluating stability effects.

Provider Configuration

Providers are assigned per task via environment variables, enabling fine-grained optimization:

- AI_PROVIDER — Global default provider (falls back to mock if not set).

- AI_<TASK>_PROVIDER — Per-task provider override (e.g., AI_CHAT_PROVIDER=anthropic).
- AI_<TASK>_MODEL — Per-task model override (e.g., AI_SUMMARY_MODEL=gemini-2.5-flash).
- AI_<TASK>_FALLBACKS — Ordered fallback chain (e.g., AI_CHAT_FALLBACKS=openai,ollama,mock).

This architecture enables organizations to use the right model for each job — fast and cheap models for summaries, reasoning-depth models for risk analysis, local models for data sovereignty — all without code changes.

Application Architecture

The portal is built on a modern full-stack JavaScript architecture:

- Next.js 16 (React 19) — Full-stack framework providing server-side rendering, API routes (BFF layer), and optimized client hydration.
- TypeScript — End-to-end type safety across client, server, and AI response parsing.
- Tailwind CSS 4 — Utility-first styling with dark mode support and responsive design.
- Tremor — React charting library for interactive time-series visualizations, bar charts, donut charts, and KPI cards.
- Remix Icons — Consistent iconography across all 44+ widgets.
- Vercel — Production deployment with edge network, serverless functions for API routes, and automatic preview deployments.
- html2canvas + jsPDF — Client-side PDF export with pixel-perfect canvas snapshot.
- @dnd-kit/core + @dnd-kit/sortable — Accessible drag-and-drop widget reordering.

Data Flow & AI Integration Pattern

The AI integration uses a three-stage pipeline:

- Context Gathering — The server collects current metrics from the customer's dashboard data (resource utilization, latency, error rates, SLA status, costs) and assembles a structured context object via task-specific gather functions.
- Prompt Construction — Role-aware system prompts are built dynamically using task-specific prompt templates, incorporating the metric context and the user's current view (C-Level, Business, or Technical).
- Provider Inference — The assembled messages are routed through the provider abstraction layer to the configured AI provider. Responses are parsed into typed structures and cached.

Shared Cache & Rate Limiting

Glasspane includes a shared cache layer (lib/ai/shared-store.ts) supporting two backends:

- In-process memory — Default for development and single-instance deployments.

- Upstash Redis REST — For multi-instance production deployments where cache should be shared across serverless functions.

The cache also handles rate limiting with fixed-window quotas per task and customer. Route security supports API key authentication with tenant-aware quotas and request ID tracing.

Route Security

All /api/ai/* routes support optional authentication and rate limiting. When enabled, routes require Authorization: Bearer <token> or x-ai-api-key headers. Each request is validated against the API key's allowed customers and rate limits. Per-route and per-task rate limits are configurable, with enterprise tier support for differentiated quotas.

UX Architecture & Client-Side Enhancements

The portal includes a suite of client-side architectural patterns:

- React Context Providers — Dedicated contexts (RefreshContext, NotificationContext, ComparisonContext, SidebarContext) manage cross-cutting UI state without prop-drilling.
- Real-Time Data Refresh — Auto-refresh toggle with per-widget refresh mechanism and spinner overlays.
- Dashboard Export / PDF — html2canvas canvas snapshot embedded into formatted, downloadable PDF reports.
- Widget Search & Filter — Debounced search with category chip filters for instant metric discovery.
- Notification Center — Slide-out panel with localStorage-persisted read/unread state.
- Widget Favorites / Pinning — useFavorites hook with localStorage persistence and sort-to-top behavior.
- Dark Mode — Standardized design tokens across all widgets for coherent light and dark themes.
- Animated Transitions — CSS keyframe animations and AnimatedNumber component for smooth loading states.
- Comparison Mode — Previous-period baseline with delta indicators for trend analysis.
- Drag-and-Drop Reordering — Accessible, keyboard-friendly widget grid reordering.

Local Model Support

For organizations where data sovereignty is critical, Glasspane supports Ollama and LM Studio as AI providers. The entire AI layer can run on local hardware — infrastructure data never leaves the network. The same task interface, fallback chains, and dashboard experience work identically with local models.

Open Source

Glasspane is open source under AGPL-3.0. The source code is available at https://github.com/MeyerThorsten/Glasspane. The license ensures that any modifications offered as a service must also be shared, preventing transparency-washing where a provider could weaken the tool's visibility features.