How would you keep costs under $5,000/month initially?

I'd keep costs in check by actively controlling how traffic is distributed between providers, favoring the more cost-efficient option and using the more expensive model only where it adds clear value. I'd also watch token usage closely so prompts don't slowly become more expensive over time.

What monitoring would help identify cost optimization opportunities?

I'd focus on tracking tokens per request, total tokens per provider, and estimated cost over time. Seeing these trends makes it easy to spot inefficient prompts, sudden usage spikes, or providers that are no longer cost-effective.

Caching strategy considerations?

For this use case, I'd cache responses for identical or very similar user profiles. Even short term caching would reduce repeated LLM calls without noticeably affecting the user experience.

How would you structure A/B tests between providers?

I'd route each request to a variant defined in configuration, where the variant explicitly specifies the provider and prompt version. That keeps experiments easy to reason about and simple to adjust without touching application code.

What metrics would determine the winner?

I'd look at a combination of response quality, latency, error rate, and cost per request. The "winner" would be the option that delivers consistently good results while staying fast and affordable.

How long should tests run?

I wouldn't fixate on a specific time window. Instead, I'd let tests run until each variant has enough traffic to produce stable, trustworthy metrics. Usually, a few thousand requests or a week or two in practice.

Identify three realistic failure modes and how you would handle them

The most likely issues are provider outages or timeouts, prompt or configuration mistakes, and unexpected increases in cost.

If a provider fails, I'd rely on retries and fallback to another provider defined in configuration.

If a prompt or config change causes bad results, I'd roll back to a known, good version immediately.

If costs spike, I'd rebalance traffic and adjust prompts based on telemetry.

How would you detect these failures quickly?

I'd rely on structured telemetry and alerts for latency, error rates, token usage, and configuration version changes so problems show up quickly and are easy to trace.

How would you measure output quality for this use case?

I'd regularly sample outputs for human review to check whether recommendations are relevant and useful, and compare quality across providers and prompt versions.

What automated checks would you implement?

I'd add simple automated checks to catch obvious issues, like unresolved placeholders, empty or malformed responses, or outputs that are far shorter or longer than expected.