Task 3: White-Label Architecture (System Design)

Context: ArrowFin wants to license this software to a new broker, "Broker X." Broker X needs their own logo, color scheme, and their users' data must be strictly isolated from EdgeClear users.

• How would you handle the database (Single DB with tenant_id vs. Multi-DB)?

• How would you serve different CSS/branding to the frontend based on the domain (e.g.,

insights.brokerx.com)?

Being honest i strongly recommend single database with tenant_id, this approach simplifies the operation and it is easier to backup, migration or maintenance, Adding new tenants (brokers) requires no infrastructure changes, just insert a new tenant record and that is it. Multi-database approaches scale poorly. With 10+ brokers, we would manage 10+ databases, connection pools, and backup schedules. Other case, what would happened if we need to change some field o restructure tables? Would be a mess. Instead a single database with proper indexing on 'tenant_id' performs excellently at scale.

Once we have a table like 'Tenants' with columns like: domain, logo_url, primary_color, secondary_color, company_name etc... We will have a middleware that matches the incoming domain to the tenant record and injects this config into the response or session.

The React app fetches 'api/tenant/config' on load, which returns the branding JSON based on the domain. Use CSS variables to apply colors dynamically:

Example: document.documentElement.style.setProperty('--primary-color', config.primaryColor);

For logos, render . Tailwind can be extended with custom color classes generated from the config. This approach requires no rebuilds, new brokers are onboarded by adding a database row and pointing their domain to the app.