**Executive Summary**

**Objective**  
This document defines the complete technical architecture, key modules, and workflow for a platform connecting Indian businesses with certified Chartered Accountants (CA) and Company Secretaries (CS), using Vite (React) as frontend and Supabase for backend, APIs, authentication, and storage. All interactions, role-based access, communications, bookings, and KYC compliance processes are covered under a secure, scalable, and serverless SaaS paradigm.

**Scope**

* Secure role-based app for businesses, professionals, and admins.
* Documented workflows: onboarding, KYC, bookings, payments, communication.
* Integration with Razorpay, WhatsApp API, and SendGrid via Supabase Edge Functions.
* Distributed, compliant-by-design using Postgres and Row Level Security (RLS).

**Technology Stack**

**Frontend**

* [Vite](https://vitejs.dev/): Lightning-fast development/build tool.
* [React.js](https://react.dev): SPA implementation.
* State: React Context.
* Styling: CSS Modules, TailwindCSS for rapid prototyping.
* Routing: React Router DOM.

**Backend/API & Database**

* [Supabase](https://supabase.com/):
  + Auth (email/mobile/OTP, social sign-in).
  + Database: PostgreSQL.
  + RESTful & Realtime API (generated from DB).
  + Storage: Documents, images (KYC, service docs).
  + Row Level Security (RLS) for privacy & role management.
  + [Edge Functions](https://supabase.com/docs/guides/functions): Custom serverless logic for webhooks, third-party API interactions.

**Integrations**

* **Razorpay**: Payment processing (handled via Edge Functions).
* **WhatsApp API**: Booking and transaction updates (via Edge).
* **SendGrid**: Transactional email (bookings, onboarding).
* **Twilio/local**: SMS or mobile OTP.
* **Firebase Cloud Messaging (FCM)**: Push notifications.

**User Roles & Access Matrix**

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Business User | CA/CS Professional | Admin |
| Register/Login | Yes | Yes | Yes |
| KYC Upload/Verify | - | Yes | Yes (verify) |
| Post Requirements | Yes | - | - |
| Browse Services | Yes | - | - |
| Bookings/Payments | Yes | - | - |
| Calendar/Schedule | Yes | Yes | - |
| In-App Messaging | Yes | Yes | - |
| Earnings Dashboard | - | Yes | - |
| Content Moderation | - | - | Yes |
| Platform Analytics | - | - | Yes |
| Notifications | Yes | Yes | Yes |

**Access Security**

* All access controlled with RLS policies.
* Route protection in React by role.
* Supabase permissions for backend data.

**Onboarding & Authentication**

**1. Registration Flow**

* SPA registration form (React): Name, email/mobile, password.
* Trigger Supabase Auth API (supabase.auth.signUp).
* OTP (via email/phone) handled by Supabase Auth (Twilio/local SMS, SendGrid).
* On success, redirect to onboarding.

**2. Onboarding**

* **Business User**:
  + Entity type (dropdown: Individual/LLP/Pvt Ltd/Others).
  + Primary compliance needs (checkbox, tags).
  + GSTIN/PAN input (if available).
* **CA/CS Professional**:
  + Essential profile: Name, contact, registration number, specialization.
  + KYC upload: PAN, Aadhaar, ICAI/ICSI document (Supabase storage).
* **Admin**: No onboarding needed (superuser access).

**3. Auth Security**

* All JWT tokens stored HTTP-only/localStorage.
* React Router guards protect private pages.

**Posting Requirements & Service Discovery**

**For Business Users:**

* **Requirement Posting**
  + React Form: Compliance area, urgency, additional notes.
  + Generates entry in requirements table (Supabase).
* **Service Discovery**
  + Fetch CAs/CSs filtered by: specialization, rating, city, fees.
  + List/Map view, paginated, lazy-loaded.
  + Each pro’s card: Name, services, available slots (live query to availability).

**For CA/CS Professionals:**

* **Profile Setup**
  + List offered services, set fees, upload/modify documentation.
  + Manage calendar via drag & drop (calendar React component).

**Realtime Updates:**

* All listings auto-refresh via Supabase REST/realtime listeners.

**Booking & Scheduling Engine**

**1. User Booking Flow**

* **Step 1:** Select service →
* **Step 2:** Pick CA/CS →
* **Step 3:** View calendar → select slot (rendered in React, backed by Supabase availability table) →
* **Step 4:** Accept terms & proceed to payment.

**2. Backend Processing**

* **Booking written to** bookings table, status=pending.
* **Slot Locking**: Postgres transactions + RLS ensures double-booking is prevented.

**3. Scheduling Logic**

* CA/CS manage their slots through the portal, updating the availability table.
* Bookings update automatically.
* Task status updated:
  + Pending → Accepted → In Progress → Completed.

**4. Notifications**

* Booking confirmation via SendGrid/Twilio/WhatsApp.
* Reminders 1 hour/day before scheduled call (via Edge Functions).

**Payment System**

**Integration Pattern:**

* **User initiates payment** in app.
* **React** calls Edge Function (/create-order) which:
  + Calls Razorpay API, returns payment order ID.
* **User completes payment** with Razorpay (client SDK or redirect).
* **On payment completion**, webhook fires to Supabase Edge Function:
  + Validates transaction, updates bookings.status to paid.
  + Optionally moves wallet credits.

**Wallet/Referral**

* Supabase table for wallet transactions.
* Referral logic: credited post service completion (Edge/SQL).
* Subscription management (future): recurring billing record in Supabase, invoicing support.

**Admin Dashboard**

* View platform and CA/CS earnings.
* Manual refunds/dispute triggers (Edge Function + admin interface).

**Messaging & Communication**

**In-App Messaging**

* Use Supabase Realtime Channels (or [Socket.IO](http://Socket.IO)) for two-way chat.
* Messages linked to task\_id, stored in messages table (text, attachments).
* UI in React: threaded chat by booking context.

**WhatsApp Integration**

* Edge Function triggers outbound WhatsApp API messages (Twilio/Meta API).
* Booking/appointment notifications, payment status, reminders.
* Admin can send platform-wide alerts, promos using Edge Function.

**Push Notifications**

* Integrate FCM client in React.
* Allow users to subscribe/unsubscribe to categories.

**Admin Panel & Platform Operations**

**UI: React Admin SPA (or integrate with user SPA, role-gated navigation)**

1. **User/Professional Management**
   * Filter/search users, verify or block profiles.
2. **KYC Review**
   * View uploaded documents, approve/reject with comments.
3. **Bookings Management**
   * Search/read bookings, override statuses (for dispute/no-show).
4. **Feedback Moderation**
   * View ratings/comments, redact/flag abusive content.
5. **Transaction/Payment Panel**
   * Track all payments, refunds, pending disbursals.
6. **Bulk Notification/Email/SMS**
   * Compose and send via SendGrid, SMS; monitor analytics.
7. **Audit Log**
   * Track all privileged/admin actions (read-only log for accountability).

**Security, Compliance, Data Privacy**

**Data Security**

* Supabase runs on SSL by default.
* All passwords hashed (bcrypt, managed by Supabase Auth).
* Document uploads scoped per user (Supabase Storage policies).
* RLS enforced for all sensitive tables—users cannot access others' records.

**Compliance**

* **GDPR & Indian IT Act**: Consent banner, logging of sign-up and withdrawals.
* **Data Erasure**: React flow for self-service account deletion; triggers Edge Function cleanup.
* **KYC/PII Storage**: All uploads encrypted at rest; access via token policy.
* **Audit Trail**: All sensitive changes/actions logged.

**Governance**

* Role-based access at both app and DB/API layer.
* Regular KYC expiry checks and notifications.
* Emergency admin override for breaches/disputes.

**Future Enhancements**

* API integration for automated PAN/Aadhaar/ICAI validation.
* Regional language interface support.
* AI-powered compliance chat, automated document parsing (OCR, ML).
* Public APIs for partner integrations.

**Appendix: High-level Architecture Diagram (Describe or attach)**

**Describe/Draw:**

1. **React (Vite) SPA** →
2. **Supabase Auth/Storage/DB/Edge Function API** →
3. **Integrations (Razorpay, WhatsApp, SendGrid) via Edge Functions**.
4. **Push via FCM, Realtime via Supabase Channels/Sockets**.
5. **Roles & Permissions at DB and UI layers**.

**Prepared for:** [Your Product/Engineering Team]  
**Date:** July 2025

**Note:**  
This doc maps directly to your requirements and deployment choices (Vite + Supabase). For DB schema, example Edge Function code (e.g., for Razorpay webhook), or custom UI wireframes, request additional appendices as needed.