📜 Collabryx Project Contract & Architectural Blueprint

Version: 1.0.0
Project: Collabryx (Next.js 14 + Supabase + AI)
Branch Strategy: - main (Production)

- develop (Staging)
- backend/setup (Zain)
- frontend/ui (Ahsan)

1. ⚔️ The Iron Rules (Meta-Contracts)

The Package Manager Pact

- **Tool:** We use **npm** exclusively.
- **Lock File:** package-lock.json is sacred. If you have a merge conflict in this file, delete it and node_modules, then run npm install.
- **No New Toys:** Do not install new libraries (axios, moment, redux, emotion) without a written agreement. Use what is installed (fetch, date-fns, zustand, tailwind).

The Env Variable Treaty

- **Zain's Duty:** If you add a secret to .env.local, you **MUST** immediately add the key (with an empty value) to .env.example and push it.
- **Ahsan's Duty:** Check .env.example after every git pull.

The Shared Validation Strategy

- **Location:** All validation logic lives in lib/validations/.
- **Rule:** Zod schemas are defined **once**.
  - Frontend imports them for react-hook-form.
  - Backend imports them for API route validation.

2. 🗺️ Routing Map (The URL Contract)

Usage: Do not hardcode strings. Import ROUTES from @/lib/constants.

Feature	URL Path	Component Owner
Landing Page	/	Ahsan

Login	/login	Ahsan
Register	/register	Ahsan
Onboarding	/onboarding	Ahsan
Dashboard	/dashboard	Ahsan
AI Assistant	/assistant	Ahsan
Semantic Matches	/matches	Ahsan
User Profile	/profile/[id]	Ahsan
Chat Room	/messages/[id]	Ahsan
Auth Callback	/api/auth/callback	Zain

3. 🗄️ Database Schema (Zain's Domain)

Note to Ahsan: Use these field names exactly in your UI. Do not use camelCase for DB columns; Supabase uses snake_case.

Table: profiles

- id (uuid, PK): References auth.users.id
- email (text): Read-only
- full_name (text): Required
- avatar_url (text): Optional
- bio (text): Optional
- role (text): 'mentor' | 'mentee'
- skills (text[]): Array of strings
- goals (text[]): Array of strings
- embedding (vector(1536)): Generated by OpenAI (Hidden from UI)
- created_at (timestamptz)

Table: conversations

- id (uuid, PK)
- created_at (timestamptz)
- participant_ids (uuid[]): Array of user IDs in the chat

Table: messages

- id (uuid, PK)
- conversation_id (uuid, FK): References conversations.id
- sender_id (uuid, FK): References profiles.id
- content (text): The actual message
- created_at (timestamptz)

4. 🔌 API Response Shapes (The Interface Contract)

Note to Zain: Your APIs must return data in exactly this format so Ahsan's frontend doesn't break.

Endpoint: GET /api/matches

Used in: hooks/use-matches.ts

```
{
  "data": [
    {
      "id": "uuid-123",
      "full_name": "John Doe",
      "role": "mentor",
      "avatar_url": "https://...",
      "skills": ["React", "Node.js"],
      "similarity": 0.89 // Calculated by vector search
    }
  ],
  "error": null
}
```

Endpoint: POST /api/chat (AI Assistant)

Used in: hooks/use-chat.ts

- **Format:** Streaming Text Response (Standard Vercel AI SDK format).
- **Input Body:** { "messages": [{ "role": "user", "content": "..." }] }

5. ☑ Division of Responsibility Checklist

🟤 Zain (Backend & Data)

- [ ] **DB Init:** Create profiles table with RLS policies (Users can only edit their own profile).
- [ ] **Vector Setup:** Enable pgvector extension in Supabase.
- [ ] **Type Gen:** Run supabase gen types and push types/database.types.ts.

- [ ] **Edge Function:** Create generate-embedding function (Triggered on Profile Update).
- [ ] **Edge Function:** Create get-matches RPC function (Cosine similarity search).
- [ ] **API:** Implement route.ts for Auth Callback.

🟢 Ahsan (Frontend & AI UI)

- [ ] **Components:** Build MessageBubble, ChatInput, MatchCard.
- [ ] **Forms:** Build Login/Register forms using react-hook-form + zod.
- [ ] **Hooks:** Create useMatches (Use dummy data [] until Zain finishes).
- [ ] **AI Page:** Build /assistant using useChat from ai/react.
- [ ] **Onboarding:** Build multi-step wizard saving to profiles table.

6. 🚨 Emergency Protocols

"The API Isn't Ready Yet" Protocol:
If Ahsan needs to build the UI but Zain hasn't finished the API:

1. Ahsan **MUST** create a mock data file in lib/mock-data.ts.
2. Ahsan builds the UI using the mock data.
3. When Zain finishes, Ahsan swaps the mock import for the fetch call.

"The Types Are Broken" Protocol:

1. Zain runs the generator locally.
2. Zain commits the file.
3. Ahsan pulls the branch.
4. **NEVER** manually edit types/database.types.ts.