Frontend Requirements - Stevie Awards Recommendation System

Hey! Here's what you need to build for the Stevie Awards recommendation system frontend.

Tech Stack
- Next.js 14+ (App Router recommended)
- TypeScript
- Tailwind CSS
- Supabase Auth (for authentication)

Screens You Need to Build

1. Landing Page (`/`)
What it looks like:
- Hero section with title: "Find Your Perfect Stevie Award"
- Brief description of what the system does
- "Get Started" button
- Clean, professional design with Stevie Awards branding

Logic:
- If user is NOT logged in → Show "Login" button → Redirect to `/auth`
- If user IS logged in → Show "Get Started" button → Check profile completion:
  - Call `GET /api/users/profile`
  - If `has_completed_onboarding` is false → Redirect to `/onboarding`
  - If true → Redirect to `/chat`

---

2. Login/Signup Page (`/auth`)
What it looks like:
- Simple login form (email + password)
- "Sign up" option for new users
- "Forgot password" link
- Use Supabase Auth UI components

What you need to do:
- Implement Supabase Auth (email/password login)
- After successful login, redirect to landing page (it will handle the profile check)

---

3. Onboarding Page (`/onboarding`)
What it looks like:
- Simple form with these fields:
  - **Full Name** (required, text input)
  - **Email** (pre-filled from Supabase, read-only)

  - **Country** (required, dropdown with all countries)
  - **Organization Name** (required, text input)
  - **Job Title** (optional, text input) - e.g., "CEO", "Marketing Manager"
  - **Phone Number** (optional, text input)
  - **Company Website** (optional, URL input)
- "Complete Profile" button at the bottom
- Clean, simple form design

What you need to do:
- Show this page only for first-time users (when `has_completed_onboarding` is false)
- Validate required fields before submission
- Call `POST /api/users/profile` with form data
- After successful submission, redirect to `/chat`

Protected Route:
- User must be logged in to access this page

---

4. Chat Interface (`/chat`)
What it looks like:
- Chat-style interface (like ChatGPT or WhatsApp)
- AI messages on the left with an avatar/icon
- User messages on the right
- Input box at the bottom for user to type responses
- Progress bar at the top: "Question 2 of 5"
- "Start Over" button in the header

How it works:
1. When page loads, call `POST /api/conversation/start` to get the first question
2. Display AI's question in the chat
3. User types their answer and hits Enter (or clicks Send)
4. Call `POST /api/conversation/respond` with their answer
5. Show typing indicator ("...") while waiting for response
6. Display AI's next question (or recommendations if done)
7. Repeat until `conversation_state` is "complete"
8. When complete, show recommendations in the same chat or redirect to results page

UI Details:
- Show typing indicator while waiting for AI response
- Disable input while waiting for response
- Auto-scroll to bottom when new messages appear
- Show progress: "Question 2 of 5" (use `progress.current` and `progress.total` from API)
- Make it mobile-responsive
- Add smooth animations for messages appearing

Protected Route:
- User must be logged in AND have completed onboarding

---

5. Recommendations Results (can be same page as chat or separate)
What it looks like:
- List of recommended award categories as cards
- Each card shows:
  - **Category name** (big and bold)
  - **Program name** (e.g., "American Business Awards")
  - **Description**
  - **Why it matches** (match reasons as bullet points)
  - **"Free" badge** if `is_free` is true
  - **Similarity score** (optional, show as percentage: "85% match")
- Group by program if multiple programs are recommended
- "Start Over" button to begin a new conversation

UI Details:
- Make cards visually appealing with good spacing
- Show top 5-10 recommendations
- Mobile-responsive grid layout (1 column on mobile, 2-3 on desktop)
- Add hover effects on cards

API Endpoints You'll Use

1. Get User Profile
Endpoint: `GET /api/users/profile`

When to call: After login to check if user has completed onboarding

Headers:
```
Authorization: Bearer <jwt-token-from-supabase>
```

You get back:
```json
{
  "success": true,
  "user": {
    "id": "uuid",
    "email": "john@techcorp.ai",
    "full_name": "John Doe",
    "country": "India",
    "organization_name": "TechCorp AI",
    "job_title": "CEO",
    "has_completed_onboarding": true
```

```
  }
}
```

OR (if new user):
```json
{
  "success": true,
  "user": {
    "id": "uuid",
    "email": "john@techcorp.ai",
    "has_completed_onboarding": false
  }
}
```

---

2. Complete User Profile
Endpoint: `POST /api/users/profile`

When to call: When user submits onboarding form

Headers:
```
Authorization: Bearer <jwt-token-from-supabase>
```

Send:
```json
{
  "full_name": "John Doe",
  "country": "India",
  "organization_name": "TechCorp AI",
  "job_title": "CEO",
  "phone_number": "+91-9876543210",
  "company_website": "https://techcorp.ai"
}
```

You get back:
```json
{
  "success": true,
  "user": {
    "id": "uuid",
    "email": "john@techcorp.ai",
    "full_name": "John Doe",
```

```
    "country": "India",
    "organization_name": "TechCorp AI",
    "job_title": "CEO",
    "phone_number": "+91-9876543210",
    "company_website": "https://techcorp.ai",
    "created_at": "2024-01-15T10:30:00Z"
  },
  "is_new_user": true
}
```

---

3. Start Conversation
Endpoint: `POST /api/conversation/start`

When to call: When user lands on chat page

Headers:
```
Authorization: Bearer <jwt-token-from-supabase>
```

Send:
```json
{
  "user_id": "uuid-from-supabase-auth"
}
```

You get back:
```json
{
  "success": true,
  "session_id": "uuid",
  "message": "Hi John! Let's find the perfect Stevie Award for TechCorp AI.",
  "question": "What type of organization is TechCorp AI?",
  "conversation_state": "collecting_org_type"
}
```

---

4. Send User Response
Endpoint: `POST /api/conversation/respond`

When to call: Every time user answers a question

Headers:
```
Authorization: Bearer <jwt-token-from-supabase>
```

Send:
```json
{
  "session_id": "uuid",
  "user_message": "We're a for-profit tech startup"
}
```

You get back (next question):
```json
{
  "success": true,
  "session_id": "uuid",
  "message": "Got it!",
  "question": "How big is your team?",
  "conversation_state": "collecting_org_size",
  "progress": {
    "current": 2,
    "total": 5
  }
}
```

OR you get back (recommendations ready):
```json
{
  "success": true,
  "session_id": "uuid",
  "message": "Perfect! Here are your recommendations:",
  "conversation_state": "complete",
  "recommendations": [
    {
      "category_id": "uuid",
      "category_name": "Company of the Year - Software",
      "description": "Recognizes outstanding software companies...",
      "program_name": "American Business Awards",
      "program_code": "ABA",
      "similarity_score": 0.85,
      "match_reasons": ["Strong innovation focus", "Global reach", "Tech-heavy orientation"],
      "is_free": false
    }
  ],
  "total_matches": 12
```

}
```

---

### 5. Health Check (Optional)
**Endpoint:** `GET /api/health`

**When to call:** On app startup to check if backend is up

---

## Authentication Flow

1. **User visits landing page** → Check if logged in
2. **Not logged in** → Show "Login" button → Redirect to `/auth`
3. **User logs in with Supabase Auth** → Get JWT token (Supabase SDK handles this)
4. **After login** → Redirect to landing page → Call `GET /api/users/profile`
5. **If `has_completed_onboarding` is false** → Redirect to `/onboarding`
6. **If true** → Redirect to `/chat`
7. **Include JWT in all API requests** → Add `Authorization: Bearer <token>` header

**What you need to do:**
- Use Supabase Auth SDK for login/signup
- Get the JWT token from Supabase after login
- Include the token in all API requests to the backend
- Handle token refresh (Supabase SDK does this automatically)
- Redirect to login if token is expired

---

## Important Notes

- The AI generates questions dynamically based on the user's profile - you don't hardcode them
- Since we collect country and org name in onboarding, the AI will ask **only 5-6 questions** instead of 8
- The backend handles all the recommendation logic
- Just display what the AI sends and send back what the user types
- When `conversation_state` is "complete", show recommendations
- Make it mobile-responsive
- Handle errors gracefully (show user-friendly messages)
- Add loading states (spinners, typing indicators)

---

## TypeScript Types (Copy These)
```

```typescript
// User Profile Types
export interface UserProfile {
  id: string;
  email: string;
  full_name: string;
  country: string;
  organization_name: string;
  job_title?: string;
  phone_number?: string;
  company_website?: string;
  has_completed_onboarding: boolean;
  created_at?: string;
}

export interface CreateProfileRequest {
  full_name: string;
  country: string;
  organization_name: string;
  job_title?: string;
  phone_number?: string;
  company_website?: string;
}

// Conversation Types
export interface ConversationStartResponse {
  success: boolean;
  session_id: string;
  message: string;
  question: string;
  conversation_state: string;
}

export interface ConversationRespondResponse {
  success: boolean;
  session_id: string;
  message?: string;
  question?: string;
  conversation_state: string;
  progress?: {
    current: number;
    total: number;
  };
  recommendations?: Recommendation[];
  total_matches?: number;
}

export interface Recommendation {
```

```
  category_id: string;
  category_name: string;
  description: string;
  program_name: string;
  program_code: string;
  similarity_score: number;
  match_reasons: string[];
  is_free: boolean;
}
```

Environment Variables

Create `.env.local`:
```
NEXT_PUBLIC_API_URL=http://localhost:3001
NEXT_PUBLIC_SUPABASE_URL=your-supabase-url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-supabase-anon-key
```

User Flow Summary

```
1. User visits landing page
   ↓
2. Clicks "Login" → Goes to /auth
   ↓
3. Logs in with Supabase
   ↓
4. Redirected to landing page → Checks profile
   ↓
5a. If new user → Goes to /onboarding → Fills form → Saves profile
   ↓
5b. If returning user → Skips onboarding
   ↓
6. Goes to /chat → AI asks 5-6 questions
   ↓
7. User answers questions
   ↓
8. Gets personalized recommendations
   ↓
9. Can click "Start Over" to try again
```

That's it! Let me know if you have questions.

convert this into a proper document