Horizon Career Platform - Complete Documentation

Version: 1.0.0
Date: November 22, 2025
Author: Development Team
Repository: https://github.com/Vaasu08/merged-app

Table of Contents

1. Executive Summary
2. Project Overview
3. Architecture
4. Features & Modules
5. Technology Stack
6. Installation & Setup
7. Database Schema
8. API Documentation
9. AI Integration
10. User Guide
11. Development Guide
12. Deployment
13. Troubleshooting
14. Future Roadmap

---

1. Executive Summary

Horizon is an AI-powered career development platform that revolutionizes how professionals discover career paths, prepare for interviews, optimize resumes, and plan their professional growth. Built with cutting-edge technologies including Google Gemini AI, LangChain, and React, Horizon provides a comprehensive suite of tools for modern job seekers.

Key Highlights

- **AI-Powered Career Matching**: Advanced algorithms analyze skills and recommend optimal career paths
- **Multi-Agent AI System**: 8 specialized AI agents for comprehensive career planning
- **ATS Optimization**: Score resumes against Applicant Tracking Systems with detailed feedback
- **Interview Simulation**: Realistic mock interviews with real-time AI feedback
- **Skill Visualization**: Interactive D3.js-powered skill graphs
- **Data Persistence**: Complete history tracking with Supabase database
- **Real-time Job Search**: Integration with multiple job listing APIs

---

2. Project Overview

Vision

To democratize access to professional career guidance through artificial intelligence, making world-class career coaching available to everyone, anywhere, anytime.

Mission

Provide an integrated platform that:

- Identifies optimal career paths based on individual skills and interests
- Prepares candidates for interviews with realistic simulations
- Optimizes resumes for maximum ATS compatibility
- Tracks professional development progress

- Connects users with relevant job opportunities

Target Users

- **Job Seekers**: Professionals looking for new opportunities
- **Career Changers**: Individuals transitioning between industries
- **Students & Graduates**: Early-career professionals entering the workforce
- **Recruiters**: HR professionals seeking qualified candidates

---

3. Architecture

System Architecture

```
┌───────────────────────────────────────────────────────────┐
│                    Frontend (React)                         │
│  ┌──────────┬──────────┬──────────┬───────────────────┐  │
│  │ Career   │ Resume   │ Interview│ Job Listings      │  │
│  │ Discovery│ Builder  │ Prep     │ & Agent Swarm     │  │
│  └──────────┴──────────┴──────────┴───────────────────┘  │
└───────────────────────────────────────────────────────────┘
                          │
                          ▼
┌───────────────────────────────────────────────────────────┐
│                   API Layer (Express)                       │
│  ┌──────────┬──────────┬──────────┬───────────────────┐  │
│  │ Gemini   │ Resume   │ Interview│ Roadmap           │  │
│  │ AI APIs  │ APIs     │ APIs     │ Generation        │  │
│  └──────────┴──────────┴──────────┴───────────────────┘  │
└───────────────────────────────────────────────────────────┘
                          │
                          ▼
┌───────────────────────────────────────────────────────────┐
│              Database (Supabase/PostgreSQL)                 │
│  ┌──────────┬──────────┬──────────┬───────────────────┐  │
│  │ Users    │ Resumes  │ Sessions │ Applications      │  │
│  │ Profiles │ ATS      │ Roadmaps │ Assessments       │  │
│  └──────────┴──────────┴──────────┴───────────────────┘  │
└───────────────────────────────────────────────────────────┘
                          │
                          ▼
┌───────────────────────────────────────────────────────────┐
│              External Services                              │
│  ┌──────────┬──────────┬──────────┬───────────────────┐  │
│  │ Google   │ LangChain│ Job APIs │ TensorFlow        │  │
│  │ Gemini   │ Agents   │ (Adzuna) │ (Face Detection)  │  │
│  └──────────┴──────────┴──────────┴───────────────────┘  │
└───────────────────────────────────────────────────────────┘
```

Tech Stack Layers

Frontend Layer

- **Framework**: React 18.3.1
- **Build Tool**: Vite 5.4.21
- **Routing**: React Router DOM 6.30.1
- **State Management**: TanStack Query 5.83.0
- **UI Components**: Radix UI, Tailwind CSS

- **Visualization**: D3.js 7.9.0, Recharts 2.15.4
- **Animation**: Framer Motion 12.23.24

Backend Layer

- **Runtime**: Node.js 18+ (20+ recommended)
- **Framework**: Express 4.21.2
- **AI Integration**: Google Generative AI, LangChain
- **File Processing**: Multer, PDF-Parse, Tesseract.js

Database Layer

- **Primary DB**: Supabase (PostgreSQL)
- **Authentication**: Supabase Auth
- **Storage**: Supabase Storage

AI & ML Layer

- **LLM**: Google Gemini (2.0-flash-exp, 1.5-flash)
- **Multi-Agent**: LangChain + LangGraph
- **Computer Vision**: TensorFlow.js, MediaPipe
- **NLP**: Built-in Gemini capabilities

---

4. Features & Modules

4.1 Career Discovery Module

Description: AI-powered career path analysis and recommendation system.

Components:

- Career Assessment Quiz (30+ questions)
- Skill-based matching algorithm
- Industry trend analysis
- Role proximity visualization
- Career recommendation cards with match scores

Key Files:

- `/src/components/CareerRecommendations.tsx`
- `/src/lib/careerRecommendationEngine.ts`
- `/src/data/careerData.ts`

Features:

- Dynamic assessment questions
- Real-time skill analysis
- Personalized career paths
- Salary insights
- Growth predictions

4.2 Resume Builder & ATS Optimizer

Description: Advanced resume creation and optimization system with ATS scoring.

Components:

- Resume template editor
- AI content enhancement
- ATS scoring engine
- PDF export functionality
- Version control system

Key Files:

- `/src/pages/ResumeBuilder.tsx`
- `/src/lib/aiResumeService.ts`
- `/src/lib/atsScorer.ts`
- `/src/lib/atsScorerAI.ts`

Features:

- **AI-Powered Content Generation**:
  - Professional summaries (60-85 words, keyword-optimized)
  - Achievement-focused bullet points (STAR method)
  - Skill categorization (Technical, Tools, Soft Skills)
- **ATS Scoring (100-point scale)**:
  - Keyword matching (30 points)
  - Formatting analysis (20 points)
  - Content quality (25 points)
  - Experience relevance (25 points)
- **10+ Detailed Suggestions**:
  - Missing keywords identification
  - Format optimization tips
  - Content enhancement recommendations
  - Quantifiable metrics suggestions
- **Database Persistence**:
  - All resumes saved automatically
  - Version history tracking
  - Primary resume designation
  - ATS scores stored with timestamps

4.3 Interview Preparation Module

Description: Comprehensive interview simulation with AI-powered feedback.

Components:

- Role-based interview setup
- Real-time question delivery
- Audio/video recording
- AI feedback generation
- Performance analytics

Key Files:

- `/src/pages/InterviewHome.tsx`
- `/src/pages/InterviewSession.tsx`
- `/src/pages/InterviewFeedback.tsx`
- `/src/contexts/InterviewContext.tsx`

Features:

- **Interview Types**:
  - Technical interviews
  - Behavioral interviews
  - Case study interviews

    - System design interviews

- **Real-time Features**:
    - Live transcription
    - Facial expression analysis (TensorFlow.js)
    - Response timing tracking
    - Confidence scoring
- **Feedback Metrics**:
    - Overall readiness score (0-100)
    - Strengths identification
    - Areas for improvement
    - Question-by-question analysis
    - Follow-up question suggestions

4.4 AI Career Agent Swarm

Description: Multi-agent AI system for comprehensive career planning.

Agents:

1. **Skill Analyzer**: Evaluates technical and soft skills
2. **Career Path Advisor**: Recommends optimal career trajectories
3. **Resume Optimizer**: Enhances resume content
4. **Interview Coach**: Provides interview preparation guidance
5. **Job Market Analyst**: Analyzes industry trends
6. **Learning Path Designer**: Creates personalized learning roadmaps
7. **Salary Negotiator**: Provides compensation insights
8. **Network Builder**: Suggests networking strategies

Key Files:

- `/src/lib/careerAgentSwarm.ts`
- `/src/pages/AgentSwarm.tsx`

Technology:

- LangChain for agent orchestration
- LangGraph for workflow management
- Google Gemini for LLM capabilities
- State management for multi-step workflows

4.5 Job Listings Integration

Description: Real-time job search with multiple API integrations.

Features:

- Job search by keywords and location
- Filtering by salary, type, and experience
- Direct application links
- Job saving functionality
- Application tracking

Key Files:

- `/src/pages/JobListings.tsx`
- `/src/lib/adzunaService.ts`

Data Sources:

- Adzuna API
- Future: LinkedIn, Indeed, Glassdoor

4.6 Skill Graph Visualizer

Description: Interactive D3.js visualization of skill relationships.

Features:

- Force-directed graph layout
- Skill node clustering
- Role proximity analysis
- Interactive zoom and pan
- Skill dependency mapping

Key Files:

- `/src/pages/SkillGraph.tsx`
- `/src/components/SkillGraphVisualizer.tsx`

4.7 Learning Roadmap Generator

Description: Personalized learning path creation with phase tracking.

Features:

- AI-generated roadmaps
- Phase-based learning structure
- Resource recommendations
- Progress tracking
- Timeline estimation

Key Files:

- `/src/pages/RoadmapView.tsx`
- `/src/contexts/RoadmapContext.tsx`
- `/src/components/roadmap/`

4.8 AI Chatbot

Description: 24/7 career guidance chatbot.

Features:

- Natural language understanding
- Context-aware responses
- Career advice
- Resume tips
- Interview preparation help

Key Files:

- `/src/components/Chatbot.tsx`
- `/src/lib/chatbotService.ts`

---

5. Technology Stack

Frontend Dependencies

```
{
  "@google/generative-ai": "^0.24.1",
  "@langchain/core": "^1.0.6",
  "@langchain/google-genai": "^1.0.3",
  "@langchain/groq": "^1.0.1",
  "@langchain/langgraph": "^1.0.2",
  "@radix-ui/react-*": "Latest versions",
  "@supabase/supabase-js": "^2.57.4",
  "@tensorflow/tfjs": "^4.22.0",
  "d3": "^7.9.0",
  "react": "^18.3.1",
  "react-router-dom": "^6.30.1",
  "framer-motion": "^12.23.24",
  "jspdf": "^3.0.3",
  "pdfjs-dist": "^5.4.394"
}
```

Backend Dependencies

```
{
  "@supabase/supabase-js": "^2.57.4",
  "cors": "^2.8.5",
  "dotenv": "^16.6.1",
  "express": "^4.21.2",
  "multer": "^2.0.2",
  "openai": "^6.7.0",
  "pdf-parse": "^2.4.5",
  "tesseract.js": "^5.1.1"
}
```

---

6. Installation & Setup

Prerequisites

- **Node.js**: v20+ (v18.19.1 minimum, but 20+ recommended for all features)
- **npm**: v9.2.0+
- **Git**: Latest version
- **Supabase Account**: For authentication and database
- **Google Gemini API Key**: For AI features

Step-by-Step Installation

1. Clone Repository

```
git clone https://github.com/Vaasu08/merged-app.git
cd merged-app
```

2. Install Dependencies

```
# Frontend dependencies
npm install

# Backend dependencies
cd server
```

```
npm install
cd ..
```

3. Environment Configuration

Create `.env` in root directory:

```
# Google Gemini AI
VITE_GEMINI_API_KEY=your_gemini_api_key_here

# Supabase Configuration
VITE_SUPABASE_URL=your_supabase_project_url
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key

# Optional: Job Search APIs
VITE_ADZUNA_APP_ID=your_adzuna_app_id
VITE_ADZUNA_API_KEY=your_adzuna_api_key
```

Create `server/.env`:

```
# Server Configuration
PORT=4000
CORS_ORIGIN=http://localhost:8080

# Google Gemini AI
GEMINI_API_KEY=your_gemini_api_key_here

# Supabase Configuration
SUPABASE_URL=your_supabase_project_url
SUPABASE_SERVICE_ROLE_KEY=your_supabase_service_role_key
```

4. Database Setup

Run the SQL scripts in your Supabase SQL editor:

```
# Execute in order:
1. database-setup.sql
2. database-business-data.sql
```

Tables Created:

- `profiles` - User profile information
- `resumes` - Resume storage and versioning
- `ats_scores` - ATS scoring history
- `interview_sessions` - Interview session data
- `interview_feedback` - Detailed interview feedback
- `roadmaps` - Learning roadmap data
- `roadmap_phases` - Roadmap phase tracking
- `career_assessments` - Career assessment results
- `job_applications` - Job application tracking

5. Start Development Servers

```
# Terminal 1 - Frontend (Port 8080)
npm run dev
```

```
# Terminal 2 - Backend (Port 4000)
npm run server:dev
```

6. Access Application

- **Frontend**: http://localhost:8080
- **Backend API**: http://localhost:4000
- **Network Access**: http://[your-ip]:8080

---

7. Database Schema

Core Tables

profiles

```sql
CREATE TABLE profiles (
  id UUID PRIMARY KEY REFERENCES auth.users(id),
  email TEXT UNIQUE NOT NULL,
  full_name TEXT,
  avatar_url TEXT,
  bio TEXT,
  location TEXT,
  phone TEXT,
  linkedin_url TEXT,
  github_url TEXT,
  portfolio_url TEXT,
  years_of_experience INTEGER DEFAULT 0,
  target_role TEXT,
  target_industry TEXT,
  skills TEXT[], -- Array of skills
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

resumes

```sql
CREATE TABLE resumes (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES profiles(id) ON DELETE CASCADE,
  title TEXT NOT NULL,
  content JSONB NOT NULL, -- Resume data structure
  version INTEGER DEFAULT 1,
  is_primary BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

ats_scores

```sql
CREATE TABLE ats_scores (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES profiles(id) ON DELETE CASCADE,
  resume_id UUID REFERENCES resumes(id) ON DELETE CASCADE,
  job_description TEXT NOT NULL,
  overall_score INTEGER NOT NULL, -- 0-100
  keyword_score INTEGER,
  formatting_score INTEGER,
  content_score INTEGER,
  experience_score INTEGER,
  suggestions JSONB, -- Array of improvement suggestions
  missing_keywords TEXT[],
  created_at TIMESTAMP DEFAULT NOW()
);
```

interview_sessions

```
CREATE TABLE interview_sessions (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES profiles(id) ON DELETE CASCADE,
  role TEXT NOT NULL,
  difficulty TEXT NOT NULL,
  duration INTEGER, -- in seconds
  questions JSONB, -- Array of questions asked
  responses JSONB, -- Array of user responses
  overall_score INTEGER, -- 0-100
  status TEXT DEFAULT 'in_progress',
  created_at TIMESTAMP DEFAULT NOW(),
  completed_at TIMESTAMP
);
```

roadmaps

```
CREATE TABLE roadmaps (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES profiles(id) ON DELETE CASCADE,
  target_role TEXT NOT NULL,
  current_role TEXT,
  timeframe TEXT,
  phases JSONB NOT NULL, -- Array of learning phases
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

Relationships

```
profiles (1) ──── (Many) resumes
profiles (1) ──── (Many) ats_scores
profiles (1) ──── (Many) interview_sessions
profiles (1) ──── (Many) roadmaps
resumes (1) ──── (Many) ats_scores
```

---

8. API Documentation

Backend API Endpoints

Health Check

```
GET /health
Response: { status: 'ok', timestamp: ISO8601 }
```

Gemini AI Endpoints

Generate Text

```
POST /api/gemini/generate
Body: {
  prompt: string,
  model?: string,
  temperature?: number,
  maxTokens?: number
}
Response: {
  text: string,
  model: string,
  duration: number
}
```

Generate Roadmap

```
POST /api/roadmap/generate
Body: {
  targetRole: string,
  currentRole: string,
  skills: string[],
  timeframe: string
}
Response: {
  roadmap: {
    phases: Array<{
      title: string,
      duration: string,
      topics: string[],
      resources: Array<{ title, url }>
    }>
  }
}
```

Resume Enhancement

```
POST /api/resume/enhance
Body: {
  personalInfo: { name, email, phone, location },
  summary: string,
  skills: string[],
  experience: Array<{ company, position, ... }>,
  education: Array<{ institution, degree, ... }>,
  targetRole: string,
  jobDescription: string
}
Response: {
  enhancedSummary: string,
  enhancedExperience: Array<{ bulletPoints: string[] }>,
  optimizedSkills: {
    technical: string[],
    tools: string[],
    soft: string[]
  }
}
```

ATS Scoring

```
POST /api/ats/score
Body: {
  resumeText: string,
  jobDescription: string
}
Response: {
  overallScore: number, // 0-100
  breakdown: {
    keywordMatch: number,
    formatting: number,
    contentQuality: number,
    experienceRelevance: number
  },
  suggestions: string[], // 10+ detailed suggestions
  missingKeywords: string[]
}
```

Cache Management

```
GET /api/cache/stats
Response: { size: number, entries: number }

DELETE /api/cache/clear
Response: { message: 'Cache cleared', itemsCleared: number }
```

---

9. AI Integration

Google Gemini Configuration

Models Used

Primary Model: `gemini-2.0-flash-exp`

- Use case: Production-ready tasks
- Speed: Ultra-fast responses
- Cost: Optimized for high volume

Fallback Model: `gemini-1.5-flash`

- Use case: Backup when primary unavailable
- Reliability: Proven stability

Service Architecture

Frontend Service (`/src/lib/geminiService.ts`):

```
export interface GeminiOptions {
  model?: string;
  temperature?: number; // 0.0 - 1.0
  topK?: number;
  topP?: number;
  maxOutputTokens?: number;
  useCache?: boolean;
  systemInstruction?: string;
}

// Main methods
geminiService.generateText(prompt, options)
geminiService.generateJSON(prompt, options)
geminiService.generateStream(prompt, options)
```

Backend Service (`/server/src/geminiClient.js`):

```
// Main methods
generateText(prompt, config)
generateJSON(prompt, schema, config)
getCacheStats()
clearCache()
```

Caching Strategy

- **TTL**: 10 minutes
- **Key Generation**: MD5 hash of (model + prompt + config)
- **Storage**: In-memory Map (upgradeable to Redis)
- **Cleanup**: Automatic on expired entries

Rate Limiting

- **Limit**: Tracked per minute
- **Reset**: Every 60 seconds
- **Retry Logic**: Exponential backoff (3 attempts)

LangChain Multi-Agent System

Agent Configuration

```
// Agent Swarm Setup
const agents = [
  new SkillAnalyzerAgent(),
  new CareerPathAdvisorAgent(),
  new ResumeOptimizerAgent(),
  new InterviewCoachAgent(),
  new JobMarketAnalystAgent(),
  new LearningPathDesignerAgent(),
  new SalaryNegotiatorAgent(),
  new NetworkBuilderAgent()
];

// LangGraph Workflow
const workflow = new StateGraph({
  channels: {
    userInput: { value: null },
    agentResults: { value: [] },
    finalOutput: { value: null }
  }
});
```

Agent Execution Flow

```
User Input → Agent Selection → Parallel Execution → Result Aggregation → Response
```

---

10. User Guide

Getting Started

1. Sign Up / Login

1. Navigate to http://localhost:8080
2. Click "Sign Up" or "Login"
3. Use email/password or OAuth providers
4. Complete profile setup

2. Career Discovery

1. Go to "Career Recommendations"
2. Take the career assessment quiz
3. Review personalized career paths
4. Explore skill graphs
5. Check salary insights

3. Resume Building

1. Navigate to "Resume Builder"
2. Upload existing resume or start from scratch
3. Fill in personal information
4. Add experience, education, skills
5. Click "Enhance with AI" for optimizations
6. Export as PDF

4. ATS Scoring

1. Go to "ATS Assessment"
2. Upload your resume
3. Paste job description
4. Click "Analyze Resume"
5. Review scores and suggestions
6. Apply recommended changes

5. Interview Preparation

1. Visit "Interview Prep"
2. Select role and difficulty
3. Start mock interview session
4. Answer questions (voice/text)
5. Receive detailed feedback
6. Track progress over time

6. Job Search

1. Navigate to "Job Listings"
2. Enter keywords and location
3. Apply filters (salary, type, experience)
4. Click jobs to view details
5. Save interesting positions
6. Track applications

7. Learning Roadmap

1. Go to "Roadmap"

2. Enter target role and current skills
3. Generate personalized roadmap
4. Track phase completion
5. Access recommended resources

---

11. Development Guide

Project Structure

```
merged-app/
├── src/
│   ├── components/       # React components
│   │   ├── ui/          # Radix UI components
│   │   ├── roadmap/     # Roadmap-specific components
│   │   └── *.tsx        # Feature components
│   ├── pages/           # Route pages
│   ├── lib/             # Utility libraries
│   │   ├── geminiService.ts
│   │   ├── careerAgentSwarm.ts
│   │   ├── atsScorer.ts
│   │   └── ...
│   ├── contexts/        # React contexts
│   ├── hooks/           # Custom hooks
│   ├── types/           # TypeScript types
│   └── data/            # Static data
├── server/
│   └── src/
│       ├── app.js       # Express server
│       └── geminiClient.js
├── public/              # Static assets
├── database-setup.sql   # Database schema
└── package.json
```

Code Style

- **TypeScript**: Strict mode enabled
- **ESLint**: Configured with React plugins
- **Formatting**: Consistent indentation (2 spaces)
- **Naming**: camelCase for variables, PascalCase for components

Adding New Features

1. Create Component

```
// src/components/NewFeature.tsx
import { useState } from 'react';

export function NewFeature() {
  const [state, setState] = useState();

  return (
    <div>
      {/* Component JSX */}
    </div>
  );
}
```

2. Add Route

```
// src/App.tsx
import NewFeature from './pages/NewFeature';

// In Routes:
<Route path="/new-feature" element={<NewFeature />} />
```

3. Create API Endpoint (if needed)

```
// server/src/app.js
app.post('/api/new-feature', async (req, res) => {
  try {
    // Handle request
    res.json({ success: true });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

4. Add Database Table (if needed)

```
-- In database-setup.sql
CREATE TABLE new_feature_data (
  id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
  user_id UUID REFERENCES profiles(id),
  data JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);
```

Testing

```
# Run linter
npm run lint

# Build for production
npm run build

# Preview production build
npm run preview
```

---

12. Deployment

Frontend Deployment (Vercel/Netlify)

Vercel

```
# Install Vercel CLI
npm i -g vercel

# Deploy
vercel --prod
```

Environment Variables:

- `VITE_GEMINI_API_KEY`
- `VITE_SUPABASE_URL`
- `VITE_SUPABASE_ANON_KEY`

Netlify

```
# Install Netlify CLI
npm i -g netlify-cli

# Build and deploy
npm run build
netlify deploy --prod --dir=dist
```

Backend Deployment (Railway/Heroku/Render)

Railway

```
# Install Railway CLI
npm i -g @railway/cli

# Login and deploy
railway login
railway init
railway up
```

Environment Variables (Server):

- `PORT`
- `GEMINI_API_KEY`
- `SUPABASE_URL`
- `SUPABASE_SERVICE_ROLE_KEY`
- `CORS_ORIGIN`

Database (Supabase)

Already managed - just ensure:

- Row Level Security (RLS) enabled
- Proper authentication policies
- Regular backups configured

---

13. Troubleshooting

Common Issues

Issue: "pdfjs-dist not found"

Solution:

```
npm install pdfjs-dist --save
```

Issue: "Node engine mismatch"

Solution: Upgrade to Node.js 20+

```
nvm install 20
nvm use 20
```

Issue: "Gemini API rate limit"

Solution: Implement request queuing or upgrade API tier

Issue: "CORS errors"

Solution: Update `server/.env`:

```
CORS_ORIGIN=http://localhost:8080,https://your-domain.com
```

Issue: "Supabase connection failed"

Solution: Verify credentials in `.env` and check Supabase dashboard

Issue: "Build fails with TypeScript errors"

Solution:

```
npm run lint
# Fix reported errors
```

Performance Optimization

1. **Enable Gemini caching** (already implemented)
2. **Lazy load components**:

```
const AgentSwarm = lazy(() => import('./pages/AgentSwarm'));
```

3. **Optimize images**: Use WebP format
4. **Code splitting**: Vite handles automatically
5. **Database indexing**: Add indexes on frequently queried columns

---

14. Future Roadmap

Phase 1: Enhanced AI (Q1 2026)

- [ ] GPT-4 integration as alternative LLM
- [ ] Voice-based interview simulation
- [ ] Real-time interview translation
- [ ] Custom agent creation UI

Phase 2: Enterprise Features (Q2 2026)

- [ ] Team collaboration tools
- [ ] Admin dashboard
- [ ] Bulk resume processing

- [ ] White-label solution
- [ ] API for third-party integrations

Phase 3: Advanced Analytics (Q3 2026)

- [ ] Career trajectory prediction
- [ ] Salary trend analysis
- [ ] Market demand forecasting
- [ ] Skill gap heat maps

Phase 4: Mobile Apps (Q4 2026)

- [ ] React Native iOS app
- [ ] React Native Android app
- [ ] Offline mode support
- [ ] Push notifications

Phase 5: Community Features (2027)

- [ ] Peer resume reviews
- [ ] Mentor matching
- [ ] Discussion forums
- [ ] Success stories sharing
- [ ] Referral system

Appendix A: Environment Variables Reference

Frontend (.env)

```
VITE_GEMINI_API_KEY=          # Required for AI features
VITE_SUPABASE_URL=            # Required for auth/database
VITE_SUPABASE_ANON_KEY=       # Required for auth/database
VITE_ADZUNA_APP_ID=           # Optional for job search
VITE_ADZUNA_API_KEY=          # Optional for job search
```

Backend (server/.env)

```
PORT=4000                     # Server port
GEMINI_API_KEY=               # Required for AI
SUPABASE_URL=                 # Required for database
SUPABASE_SERVICE_ROLE_KEY=    # Required for database admin
CORS_ORIGIN=                  # Comma-separated allowed origins
```

Appendix B: API Keys Setup

Google Gemini API Key

1. Visit: https://makersuite.google.com/app/apikey
2. Click "Create API Key"
3. Copy and paste into `.env` files

Supabase Setup

1. Create project: https://supabase.com/dashboard

2. Go to Settings → API
3. Copy:
   - Project URL → `SUPABASE_URL`
   - Anon Public Key → `SUPABASE_ANON_KEY`
   - Service Role Key → `SUPABASE_SERVICE_ROLE_KEY`

Adzuna API (Optional)

1. Register: https://developer.adzuna.com/
2. Create application
3. Copy App ID and API Key

Appendix C: Database Queries

Common Queries

Get User's Primary Resume

```sql
SELECT * FROM resumes
WHERE user_id = 'user-uuid'
AND is_primary = true
ORDER BY updated_at DESC
LIMIT 1;
```

Get Recent ATS Scores

```sql
SELECT
  a.overall_score,
  a.created_at,
  r.title as resume_title
FROM ats_scores a
JOIN resumes r ON a.resume_id = r.id
WHERE a.user_id = 'user-uuid'
ORDER BY a.created_at DESC
LIMIT 10;
```

Get Interview Performance Trend

```sql
SELECT
  DATE_TRUNC('week', created_at) as week,
  AVG(overall_score) as avg_score,
  COUNT(*) as session_count
FROM interview_sessions
WHERE user_id = 'user-uuid'
AND status = 'completed'
GROUP BY week
ORDER BY week DESC;
```

Appendix D: Contributing Guidelines

Pull Request Process

1. Fork the repository

2. Create feature branch: `git checkout -b feature/amazing-feature`
3. Commit changes: `git commit -m 'Add amazing feature'`
4. Push to branch: `git push origin feature/amazing-feature`
5. Open Pull Request

Commit Message Format

```
type(scope): subject

body

footer
```

Types: feat, fix, docs, style, refactor, test, chore

Code Review Checklist

- [ ] Code follows style guidelines
- [ ] Tests pass
- [ ] Documentation updated
- [ ] No console.log statements
- [ ] Error handling implemented
- [ ] Type safety maintained

Appendix E: Support & Contact

Getting Help

- **Documentation**: This file
- **Issues**: https://github.com/Vaasu08/merged-app/issues
- **Discussions**: https://github.com/Vaasu08/merged-app/discussions

Reporting Bugs

Include:

1. Steps to reproduce
2. Expected behavior
3. Actual behavior
4. Screenshots (if applicable)
5. Environment (OS, Node version, browser)

Feature Requests

Submit via GitHub Issues with:

- Clear description
- Use case
- Expected benefits
- Implementation suggestions (optional)

Document Version: 1.0.0
Last Updated: November 22, 2025
Maintained By: Horizon Development Team

End of Documentation