Resume Refiner - CareerScope AI Documentation

Version 1.0

2025-08-04

Prepared by Varun Shah

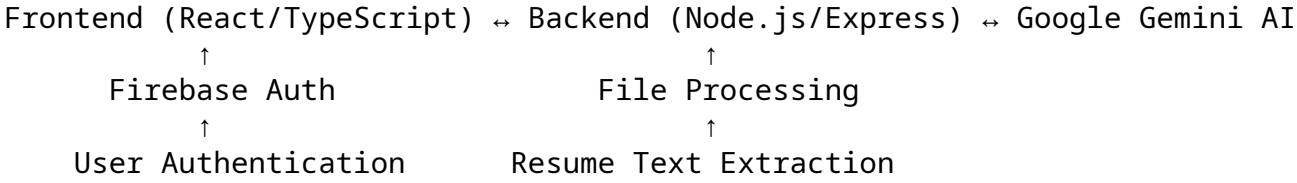
Contents

1 Project Overview

Resume Refiner (CareerScope AI) is a full-stack web application that analyzes resumes using AI to provide career insights, skills assessment, and personalized recommendations. The application helps users understand their career trajectory, identify skill gaps, and discover learning opportunities to future-proof their careers against AI disruption.

2 Architecture

The application follows a client-server architecture:



2.1 Technology Stack

Frontend:

- React with TypeScript
- Vite build tool
- shadcn/ui components with Tailwind CSS
- React Router for navigation
- Firebase for authentication
- Recharts for data visualization

Backend:

- Node.js with Express.js
- Google Gemini AI API
- PDF and DOCX text extraction
- Multer for file uploads
- CORS for cross-origin requests

Deployment:

- Docker containers for both frontend and backend
- Nginx for frontend serving
- Environment-based configuration

3 Features

1. **Resume Analysis**
   - Upload PDF or DOCX resumes
   - AI-powered analysis using Google Gemini
   - Skills assessment with radar chart visualization
   - Career growth predictions
2. **AI Impact Analysis**
   - Risk level assessment (Low/Medium/High)
   - Timeline predictions for AI impact
   - Adaptation recommendations
3. **Learning Paths**
   - Personalized course recommendations
   - Platform-specific links (Coursera, Udemy, LinkedIn Learning)
   - Skill gap addressing
4. **User Management**
   - Firebase authentication (Email/Password, Google)
   - User profile management
   - Protected routes
5. **Responsive UI**
   - Mobile-friendly design
   - Drag-and-drop file upload
   - Real-time feedback and notifications

4 Frontend

4.1 Frontend Components

The frontend is built with React and TypeScript, using shadcn/ui components for a consistent UI experience.

Main Components:

1. **UploadSection** (`/src/components/UploadSection.tsx`)
   - Drag-and-drop file upload interface
   - File validation (PDF/DOCX, 10MB limit)
   - User profile display
2. **AnalysisResults** (`/src/components/AnalysisResults.tsx`)
   - Main dashboard for displaying analysis results
   - Job title and experience level display
   - AI risk assessment badges
3. **SkillsRadarChart** (`/src/components/SkillsRadarChart.tsx`)
   - Radar chart visualization of current vs. recommended skills
   - Interactive chart with legend
4. **EmergingRoles** (`/src/components/EmergingRoles.tsx`)
   - Career growth predictions for emerging roles
   - Match percentage indicators
5. **AIImpactAnalysis** (`/src/components/AIImpactAnalysis.tsx`)
   - AI impact timeline and adaptation potential
   - Summary of AI effects on the user's role
6. **Recommendations** (`/src/components/Recommendations.tsx`)
   - Personalized career recommendations
   - Actionable advice for career growth
7. **LearningPaths** (`/src/components/LearningPaths.tsx`)
   - Recommended courses to address skill gaps
   - Platform and duration information
8. **UserProfile** (`/src/components/UserProfile.tsx`)
   - User profile management
   - Account settings and deletion
9. **Authentication Components**
   - Login (`/src/components/Login.tsx`)
   - Signup (`/src/components/Signup.tsx`)
   - ProtectedRoute (`/src/components/ProtectedRoute.tsx`)

4.2 Routing

The application uses React Router for navigation with protected routes:

```
/               -> Redirects to /login
/login          -> Login page
/signup         -> Signup page
/dashboard      -> Main dashboard (Protected)
/profile        -> User profile (Protected)
/upload         -> Resume upload (Protected)
*               -> 404 Not Found page
```

4.3 Authentication

Firebase Authentication is used for user management:

- Email/password authentication
- Google OAuth integration
- Protected routes requiring authentication
- User profile persistence in Firestore

5 Backend

5.1 API Endpoints

POST /api/analyze-resume

- Upload and analyze a resume file
- Accepts multipart/form-data with 'resume' file field
- Returns detailed career analysis JSON

GET /api/health

- Health check endpoint
- Returns server status

5.2 AI Integration

The backend uses Google Gemini AI for resume analysis:

1. **Text Extraction**
   - PDF parsing using pdf2json
   - DOCX parsing using mammoth
   - Text normalization and cleaning
2. **AI Analysis**
   - Google Generative AI SDK
   - Custom prompt engineering for career analysis
   - Structured JSON response parsing
   - Fallback to mock data on API errors
3. **Caching**
   - MD5-based caching of analysis results
   - Cache directory management
   - Performance optimization
4. **Error Handling**
   - File type and size validation
   - API quota monitoring
   - Graceful fallback to mock data
   - Detailed error logging

6 Deployment

6.1 Docker Configuration

The application is containerized using Docker with separate containers for frontend and backend:

docker-compose.yml:

```
version: '3'
services:
  frontend:
    build: ./frontend
    ports:
      - "80:80"
    depends_on:
      - backend

  backend:
    build: ./backend
    ports:
      - "3000:3000"
    env_file:
      - ./backend/.env
```

Frontend Dockerfile:

- Multi-stage build (Node.js builder + Nginx production)
- Vite build for React application
- Nginx configuration for serving static files

Backend Dockerfile:

- Node.js 18 base image
- NPM dependency installation
- Server port exposure
- Node.js entry point

7 Development Setup

7.1 Prerequisites

- Node.js >= 18.0.0
- NPM >= 8.0.0
- Docker (for containerized deployment)
- Google API key for Gemini AI
- Firebase project with authentication enabled

7.2 Frontend Setup

1. Navigate to the frontend directory:

```
cd frontend
```

2. Install dependencies:

```
npm install
```

3. Create environment variables file:

```
cp .env.example .env
```

4. Add Firebase configuration to `.env`:

```
VITE_FIREBASE_API_KEY=your_firebase_api_key
VITE_FIREBASE_AUTH_DOMAIN=your_firebase_auth_domain
VITE_FIREBASE_PROJECT_ID=your_firebase_project_id
VITE_FIREBASE_STORAGE_BUCKET=your_firebase_storage_bucket
VITE_FIREBASE_MESSAGING_SENDER_ID=your_firebase_messaging_sender_id
VITE_FIREBASE_APP_ID=your_firebase_app_id
VITE_FIREBASE_MEASUREMENT_ID=your_firebase_measurement_id
```

5. Start development server:

```
npm run dev
```

6. Build for production:

```
npm run build
```

7.3 Backend Setup

1. Navigate to the backend directory:

```
cd backend
```

2. Install dependencies:

```
npm install
```

3. Create environment variables file:

```
cp .env.example .env
```

4. Add Google API key to `.env`:

```
GOOGLE_API_KEY=your_google_api_key
PORT=3001
```

5. Start development server:

```
npm run dev
```

6. Start production server:

```
npm start
```

8 Environment Variables

8.1 Frontend (.env)

```
VITE_FIREBASE_API_KEY=your_firebase_api_key
VITE_FIREBASE_AUTH_DOMAIN=your_firebase_auth_domain
VITE_FIREBASE_PROJECT_ID=your_firebase_project_id
VITE_FIREBASE_STORAGE_BUCKET=your_firebase_storage_bucket
VITE_FIREBASE_MESSAGING_SENDER_ID=your_firebase_messaging_sender_id
VITE_FIREBASE_APP_ID=your_firebase_app_id
VITE_FIREBASE_MEASUREMENT_ID=your_firebase_measurement_id
```

8.2 Backend (.env)

```
GOOGLE_API_KEY=your_google_api_key
PORT=3001
```

9 Project Structure

```
resume-refiner/
├── backend/
│   ├── server.js              # Main server file
│   ├── package.json           # Backend dependencies
│   ├── Dockerfile             # Backend Docker configuration
│   ├── .env.example           # Environment variables template
│   ├── README-backend.md      # Backend documentation
│   ├── cache/                 # Analysis result cache
│   └── requirements.txt       # Python dependencies (if any)
├── frontend/
│   ├── index.html             # Main HTML file
│   ├── package.json           # Frontend dependencies
│   ├── Dockerfile             # Frontend Docker configuration
│   ├── nginx.conf             # Nginx configuration
│   ├── vite.config.ts         # Vite configuration
│   ├── tailwind.config.ts     # Tailwind CSS configuration
│   ├── tsconfig.json          # TypeScript configuration
│   ├── src/
│   │   ├── App.tsx            # Main App component
│   │   ├── main.tsx           # Entry point
│   │   ├── firebase.js        # Firebase configuration
│   │   ├── pages/
│   │   │   ├── Index.tsx      # Main dashboard page
│   │   │   └── NotFound.tsx  # 404 page
│   │   ├── components/        # UI components
│   │   ├── hooks/             # Custom React hooks
│   │   ├── lib/               # Utility functions
│   │   └── utils/             # Helper functions
│   └── public/                # Static assets
├── docker-compose.yml         # Docker orchestration
└── README.md                  # Project overview
```

10 Troubleshooting

10.1 Common Issues

1. **CORS Errors**
   - Ensure backend CORS configuration includes your frontend domain
   - Check environment variables for correct API URLs
2. **Firebase Authentication Issues**
   - Verify Firebase project configuration
   - Check environment variables for all required Firebase keys
   - Ensure Firebase Authentication is enabled in Firebase Console
3. **AI Analysis Failures**
   - Verify Google API key is valid and has Gemini permissions
   - Check API quota limits in Google Cloud Console
   - Ensure resume files are not corrupted
4. **File Upload Issues**
   - Verify file type is PDF or DOCX
   - Check file size is under 10MB limit
   - Ensure browser supports drag-and-drop
5. **Docker Deployment Issues**
   - Ensure Docker is running
   - Check port availability (80 for frontend, 3000 for backend)
   - Verify environment variables are correctly set in containers

10.2 Debugging Tips

1. **Frontend Debugging**
   - Check browser console for JavaScript errors
   - Use React DevTools for component inspection
   - Monitor Network tab for API request failures
2. **Backend Debugging**
   - Check server console logs for error messages
   - Verify environment variables are loaded correctly
   - Test API endpoints with tools like Postman
3. **Performance Optimization**
   - Enable caching for repeated analysis requests
   - Optimize image assets for web delivery
   - Minimize bundle size with code splitting

10.3 Support

For issues not covered in this documentation:

1. Check server logs for detailed error messages
2. Verify all environment variables are correctly set
3. Ensure all dependencies are properly installed
4. Contact the development team with detailed error descriptions