MCP Resume Assistant

A Model Context Protocol (MCP) server that enables AI assistants to interact with resume data and send emails. Built with Next.js frontend and Node.js backend.

What It Does

- **Resume Intelligence**: Parse PDF resumes and answer questions about skills, education, and projects
- **Email Service**: Send professional emails using the Resend API
- **Web Playground**: Interactive interface to test both features

Architecture

```
Frontend (Next.js) ↔ MCP Server (Node.js) ↔ External Services
    ↓                  ↓                    ↓
- Chat Interface     - CV Parser          - Resend API
- Email Form         - Email Service      - PDF Files
- API Routes         - MCP Tools          - Vercel Deploy
```

Tech Stack

Frontend

- Next.js 15.5.2 with TypeScript
- React 19 and Tailwind CSS
- React Hook Form for validation

Backend

- Node.js with MCP SDK
- pdf-parse for PDF extraction
- Resend API for emails
- Zod for validation

Quick Start

1. Clone and install:

```bash
git clone https://github.com/adeeshperera/mcp-resume-assistant.git
cd mcp-resume-assistant
npm install
```

2. Set up environment:

```bash
cp .env.example .env.local
# Add your RESEND_API_KEY
```

3. Add your resume PDF to `/public/` directory
4. Run the app:

```bash
npm run dev  # Frontend at localhost:3000
npm run mcp  # MCP server standalone
```

API Endpoints

Ask About Resume

```bash
POST /api/mcp
{
  "question": "What skills does the person have?"
}
```

Send Email

```bash
POST /api/email
{
  "recipient": "user@example.com",
  "subject": "Hello",
  "body": "Message content"
}
```

Key Features

CV Intelligence

- Extracts text from PDF resumes
- Answers questions about education, skills, projects
- Uses keyword matching for fast responses
- Caches data for better performance

Email System

- Professional email sending via Resend
- Form validation and error handling
- Real-time delivery status
- Mobile-friendly interface

File Structure

```
src/
├── app/              # Next.js frontend
│   ├── components/   # UI components
│   ├── api/          # API routes
│   └── page.tsx      # Main page
└── server/           # MCP backend
    ├── tools/        # MCP tool definitions
    ├── services/     # Core services
    └── index.ts      # Server entry
```

Configuration

Environment Variables

- `RESEND_API_KEY` - Required for email functionality
- `NEXT_PUBLIC_APP_URL` - App URL (optional)

Customization

- Replace PDF in `/public/` with your resume
- Update CV parsing logic in `cv-parser.ts`
- Modify email templates in `email-service.ts`

Testing

```bash
npm test          # Run all tests
npm run test:watch  # Watch mode
npm test -- --coverage  # Coverage report
```

Test files:

- `cv-parser.test.ts` - Resume parsing tests
- `email-service.test.ts` - Email validation tests
- `mcp-tools.test.ts` - MCP functionality tests

Deployment

Deploy to Vercel:

```bash
npm run build
vercel --prod
```

Set environment variables in Vercel dashboard.

How It Works

1. **Frontend** provides chat interface and email form
2. **API Routes** handle HTTP requests and validate input
3. **MCP Server** processes requests using registered tools
4. **CV Parser** extracts and searches resume content
5. **Email Service** sends messages via Resend API
6. **Response** returns formatted data to frontend

Dependencies

Core packages:

- `@modelcontextprotocol/sdk` - MCP implementation
- `next` - React framework
- `pdf-parse` - PDF text extraction
- `resend` - Email service
- `zod` - Input validation

License

Apache 2.0

Built to demonstrate MCP server capabilities with practical resume and email features.