ATS Resume Checker - Full Setup Guide

1️⃣ MongoDB Atlas Setup

1. Create a free MongoDB Atlas account.
2. Create a new cluster (Free Tier).
3. Create a database user:
4. Username: `atsuser`
5. Password: `ats12345`
6. Allow access from your IP.
7. Copy the connection string, replace `<password>` with your password:

```
mongodb+srv://atsuser:ats12345@ats-cluster.xxxxx.mongodb.net/?
retryWrites=true&w=majority
```

2️⃣ Backend Setup (Node.js + Express)

Folder structure:

```
backend/
├─ server.js
├─ package.json
└─ node_modules/
```

Install dependencies:

```
npm init -y
npm install express mongoose cors
```

server.js:

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();
app.use(cors());
```

```
app.use(express.json());

// MongoDB connection
mongoose.connect("mongodb+srv://atsuser:ats12345@ats-cluster.xxxxx.mongodb.net/?
retryWrites=true&w=majority")
.then(() => console.log('✅ MongoDB connected'))
.catch(err => console.log('❌ MongoDB connection error:', err));

// User model
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  password: String
});
const User = mongoose.model('User', userSchema);

// Register API
app.post('/register', async (req, res) => {
  const { name, email, password } = req.body;
  const user = new User({ name, email, password });
  await user.save();
  res.json({ success: true, message: 'User registered successfully', user });
});

app.listen(5000, () => console.log('🚫 Server running on http://localhost:
5000'));
```

Test `/register` **API:** - Use PowerShell:

```
Invoke-RestMethod -Uri "http://localhost:5000/register" -Method POST -
ContentType "application/json" -Body
'{"name":"Shrinivas","email":"shri@test.com","password":"123456"}'
```

- Check MongoDB Atlas `users` collection for the inserted user.

3️⃣ Frontend Setup (React + Vite)

Create React App:

```
npm create vite@latest frontend
# Choose React, JavaScript
cd frontend
```

```
npm install
npm run dev
```

App.jsx (simplified login/register form):

```
import { useState } from 'react';

function App() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');

  const handleRegister = async () => {
    const res = await fetch('http://localhost:5000/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ name, email, password })
    });
    const data = await res.json();
    setMessage(data.message);
  };

  return (
    <div>
      <h1>Register</h1>
      <input placeholder="Name" value={name} onChange={e =>
setName(e.target.value)} />
      <input placeholder="Email" value={email} onChange={e =>
setEmail(e.target.value)} />
      <input placeholder="Password" type="password" value={password}
onChange={e => setPassword(e.target.value)} />
      <button onClick={handleRegister}>Register</button>
      <p>{message}</p>
    </div>
  );
}

export default App;
```

Run frontend:

```
npm run dev
```

Open browser: `http://localhost:5173`

4️⃣ Notes on PowerShell & File Uploads

- PowerShell `curl` and `POST` commands behave differently from Linux curl.
- Recommended: Use **Postman** for file upload testing.
- Steps in Postman:
- Method: POST
- URL: `http://localhost:5000/upload-resume`
- Body → form-data → key=`file`, type=File → select PDF

- Send → expect JSON response.

- Backend must have multer:

```
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });
app.post('/upload-resume', upload.single('file'), (req,res)=>{
  if(!req.file) return
res.status(400).json({success:false,message:'No file uploaded'});
  res.json({success:true,filename:req.file.filename});
});
```

5️⃣ Next Steps for ATS App

1. Add `/parse-resume` API using `pdf-parse`.
2. Add React Resume Upload page.
3. Test resume upload & parsing locally.
4. Deploy backend (Render) and frontend (Vercel).
5. Integrate login + ATS scoring + dashboard.

This document captures **everything we've done so far**, including local testing and frontend/backend setup. You can continue from here for ATS features.