📄 Final Assignment Document

AI-Powered Resume Analyzer & Job Tracker

📌 Project Overview

Your final assignment is to build a **full-stack AI-powered web application** that allows users to:

1. Upload or paste their **resume**.
2. Analyze it using **Gemini (or any AI API)** and receive:
    - Resume score
    - Improvement suggestions
    - Grammar fixes
    - ATS readability feedback
3. Save the improved resume data into the database.
4. Create a **Job Tracker Dashboard** where users can:
    - Add job applications
    - Track their status (Applied, Interviewing, Rejected, Offered)
    - Add interview notes
    - Get AI suggestions for improvement

This assignment tests **real-world engineering skills**, including a backend API, database design, authentication, CRUD operations, and AI integration.

🎯 Learning Objectives

By completing this assignment, students will:

✔ Build a production-style **Node.js + MongoDB backend**

✔ Create REST APIs for users, resumes, and job applications

✔ Use **HTML, CSS, JavaScript** or a frontend framework (React optional)

✔ Integrate **AI as a service** using Gemini API

✔ Manage file uploads (text or document content)

✔ Implement CRUD functionality

✔ Use authentication (JWT optional but recommended)

✔ Build a real-world project they can show in their portfolios

---

🧩 Feature Requirements

1️⃣ User Account System

- Register account
- Login
- JWT authentication (optional but strongly recommended)
- Store user details in MongoDB

2️⃣ Resume Analyzer

User can:

- Upload text from resume (plain text)
- Submit resume to backend

- Backend sends data to Gemini AI
- AI responds with structured JSON:

Example response:

```
{
  "score": 82,
  "suggestions": [
    "Add measurable achievements",
    "Rewrite summary more professionally",
    "Improve grammar in job experience section"
  ],
  "grammarFixes": "... corrected content ...",
  "atsScore": 75
}
```

Stored in database

- User ID
- Original resume text
- AI-enhanced text
- AI feedback / score
- Timestamp

3️⃣ Job Tracker Dashboard

A user should be able to:

- Add a job entry with:
    - Company name
    - Position

    - JD link or text
    - Applied date
    - Status (Applied / Interviewing / Rejected / Offered)
    - Notes
- Edit job entries
- Delete job entries
- View them in a dashboard table

Optional AI Feature

User can click:

`"Analyze Job Description"` → AI suggests:

- Required keywords
- Missing skills in user resume

🗃️ Database Design

Users Collection

```
{
  "_id": "ObjectId",
  "name": "string",
  "email": "string",
  "password": "hashed"
}
```

Resumes Collection

```
{
```

```
  "_id": "ObjectId",
  "userId": "ObjectId",
  "originalText": "string",
  "aiImprovedText": "string",
  "aiScore": "number",
  "atsScore": "number",
  "suggestions": "array",
  "createdAt": "date"
}
```

Jobs Collection

```
{
  "_id": "ObjectId",
  "userId": "ObjectId",
  "company": "string",
  "position": "string",
  "jobDescription": "string",
  "status": "Applied | Interviewing | Rejected | Offered",
  "notes": "string",
  "createdAt": "date"
}
```

🖥️ Backend API Requirements

Method	Endpoint	Description
POST	`/api/auth/register`	Create new user
POST	`/api/auth/login`	Login user
POST	`/api/resume/analyze`	Send resume to Gemini and store results
GET	`/api/resume`	Fetch resume history
POST	`/api/jobs`	Add new job
GET	`/api/jobs`	List all jobs

PUT	`/api/jobs/:id`	Update job
DELETE	`/api/jobs/:id`	Delete job

🤖 Gemini API Integration (Node Example)

```
import axios from "axios";

const GEMINI_API_KEY = process.env.GEMINI_KEY;

export const analyzeResume = async (resumeText) => {
  const prompt = `
  Analyze this resume and return:
  - Resume score (0–100)
  - Suggestions to improve
  - ATS score
  - Corrected version

  Resume:
  ${resumeText}
  `;

  const response = await axios.post(

"https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key="
+ GEMINI_API_KEY,
    { contents: [{ parts: [{ text: prompt }] }] }
  );

  return response.data;
};
```

🎨 Frontend Requirements

You may use:

- Plain HTML/CSS/JS
  OR
- React / Next.js (recommended)

Minimum pages:

1. Login / Register Page
2. Resume Upload & Analysis Page
3. Job Tracker Dashboard

Resume Analysis Page

Fields:

- Resume textarea
- Submit button
- Show AI results beautifully

Job Tracker UI

A table:

Company	Position	Status	Actions

Allow editing/deleting rows.

📅 Project Milestones

Week 1 – Backend Setup

- Node.js + Express server

- MongoDB connection
- User authentication
- Postman functional

Week 2 – Resume Analysis

- Create API route for Gemini integration
- Store results in DB
- Return JSON response

Week 3 – Job Tracker

- CRUD routes for job entries
- Connect to DB
- Test APIs

Week 4 – Frontend

- Create UI screens
- Connect APIs
- Display results properly

📌 Bonus Features (Optional)

✔ Upload PDF and convert to text
✔ AI cover letter generator
✔ AI-based job match score
✔ Export resume as PDF
✔ Real-time charts for job progress

📊 Evaluation Criteria

Criteria	Weight
Backend API completeness	30%
MongoDB schema & data handling	20%
AI integration working	20%
Frontend usability and design	20%
Code quality & documentation	10%

📁 Submission Requirements

Students must submit:

1. Source code (GitHub repository)
2. README including:
    - Tech stack
    - Setup instructions
    - API documentation
3. Demo video (3–10 mins) explaining:
    - Features
    - Code walkthrough
    - AI integration demo