AI Resume & Portfolio Assistant: Project Documentation

1. Project Journey and Overview

This document provides a comprehensive report on the design, development, and deployment of the **AI Resume & Portfolio Assistant**. The project aims to empower job seekers and professionals by providing AI-powered resume analysis, ATS compatibility scoring, tailored cover letter generation, and seamless export features.

The journey started with researching AI capabilities for natural language understanding of resumes and job descriptions, identifying a suitable language model, and designing a user-friendly web interface. The development went through iterations focusing separately on frontend UI/UX, backend API integration, error handling, and finally on overall system reliability & presentation.

2. Problem Statement

Job applicants often struggle with tailoring resumes to individual job descriptions and crafting engaging cover letters. ATS systems frequently filter out candidates due to missing keywords or poor resume formatting. The AI Resume & Portfolio Assistant bridges this gap by analyzing submitted resumes in context with job descriptions, evaluates strengths and weaknesses, offers concrete recommendations, and generates personalized cover letters—all with downloadable PDF exports.

3. Technical Stack

Frontend

- **React** with **TypeScript** for scalable, modern UI development
- **Framer Motion** for smooth animations and button interactions
- **Lucide Icons** for consistent iconography

- **Axios** for HTTP client requests
- **html2canvas** and **jsPDF** to generate downloadable PDF reports
- **CSS with CSS variables** for light/dark themes and responsive layout
- **Custom components**: FileUpload, JobDescriptionInput, EnhancedResumeAnalysis, CoverLetterGenerator, Profile inputs (GitHub, LinkedIn), ThemeToggle

Backend

- **FastAPI** (Python) for fast, async web API development
- **Pydantic** for request/response models and validation
- **httpx** async HTTP client for calling external AI APIs
- **Environment management** with `.env` file for sensitive keys
- **File parsing libraries**:
  - `PyPDF2` for PDF resume text extraction
  - `python-docx` for DOCX files
- **Middleware**:
  - `CORSMiddleware` to allow React frontend access from different origin
- **Temporary file handling** for uploaded resumes before parsing and analysis

AI Model and API

- **Google Gemini 1.5 Flash** via Google Generative Language API (REST)
  - Used for generating structured resume analysis and cover letters
  - Custom prompt templates ensure consistent JSON output with keys like `overall_rating`, `ats_score`, `skills_match`, and recommendations
  - Temperature and max tokens tuned for balanced generation quality

Database

- No persistent database is used. This is a stateless API serving ad-hoc analysis on resume data uploaded or pasted by the user.

- All state and data management happen in-memory on both frontend (React state hooks) and backend (per-request processing).

4. System Architecture Overview

The system consists of three major components:

1. **Frontend (React):**
    - Collects user inputs (resume, job description, profile links)
    - Sends requests to backend via REST APIs
    - Displays formatted AI analysis and cover letter text
    - Provides PDF export options with client-side rendering
2. **Backend (FastAPI):**
    - Receives requests, parses resume files if applicable
    - Constructs detailed structured prompts for Gemini API
    - Calls Gemini API asynchronously, receives JSON or text outputs
    - Parses AI response, sanitizes and guarantees consistent structure
    - Returns data to frontend for rendering
3. **Google Gemini API:**
    - Performs large language model inference on prompts
    - Returns resume assessments and cover letter generation

No database or additional persistent storage are involved, simplifying deployment.

5. Key Technical Details

Prompt Engineering

- Customized prompts instruct Gemini to respond with strictly formatted JSON containing arrays for skills match/gap, strengths, weaknesses, recommendations, and numeric scores.
- Tuning temperature and max tokens to balance creativity and output length.
- Separate prompts for resume analysis and cover letter generation.

Resume File Processing

- Uploaded PDF or Word files are temporarily saved on backend and parsed using dedicated Python libraries to plain text.
- Text is sanitized and combined with optional profile insights (GitHub, LinkedIn) before prompt construction.

Frontend Export

- Results and cover letter sections are rendered as discrete cards.
- `html2canvas` takes screenshots of these sections and `jsPDF` compiles them into a downloadable, centered PDF document.
- Export buttons are intuitive and available for partial or full reports.

Dark/Light Theme Support

- CSS variables are used to toggle colors and backgrounds dynamically.
- ThemeToggle component persists user preferences.

Resilience and Error Handling

- Backend returns default structured data if AI output parsing fails.
- CORS middleware properly allows cross-origin React frontend access.
- API rate limiting errors from Gemini are caught with recommended fallback or retry logic.
- Frontend validation prevents empty requests.

6. Future Improvements

- Adding a lightweight database for user session persistence and history.
- Real-time GitHub and LinkedIn scraping for enhanced profile insights.
- User authentication and profile management.
- More advanced NLP scoring algorithms with machine learning.
- Multilingual support.
- Containerized deployment (Docker) and CI/CD automation.

7. Conclusion

The AI Resume & Portfolio Assistant demonstrates a full-stack AI application integrating frontend UI, backend APIs, and advanced LLM inference. Using Google Gemini, it delivers actionable resume feedback and personalized cover letters, significantly improving job seekers' application effectiveness. The project balances flexibility and robustness with a clean UX backed by a resilient API architecture.