Smart Resume Generator: Customized Resumes for Every Opportunity

Project Title:

- Smart Resume Generator

Team Name(69) :

- TechSquads

Team Members:

- P.Akshitha (23WH1A05D3)
- M.Vaishnavi (23WH1A05E2)
- T.Likhitha sree (23WH1A05E5)
- M.Sravani (23WH1A05E8)

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered resume-building tool that tailors resumes dynamically based on job descriptions and user profiles.

Key Points:

1. **Problem Statement:**
    - Job seekers struggle to customize resumes for different opportunities.
    - Recruiters often filter candidates based on ATS-optimized resumes.
2. **Proposed Solution:**
    - An AI-driven resume generator that aligns resumes with job descriptions.
    - Custom formatting, keyword optimization, and design personalization.
3. **Target Users:**
    - Job seekers applying for multiple roles.
    - Recruiters looking for well-structured resumes.
    - Professionals wanting career advancement opportunities.
4. **Expected Outcome:**

- A dynamic AI-powered resume builder with export options.
- Enhanced interview call rates due to ATS-optimized resumes

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the Smart Resume Generator.

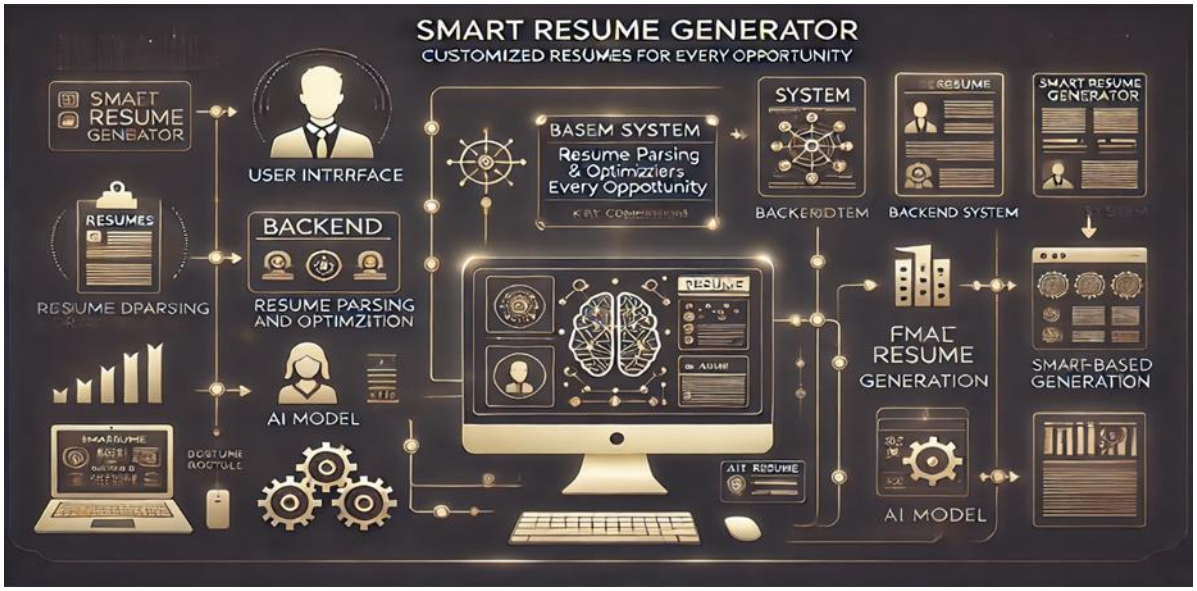
Key Points:

1. **Technical Requirements:**
   - **Programming Language:** Python
   - **Backend:** Flask/FastAPI
   - **Frontend:** Streamlit
   - **Database:** JSON (for user profiles)
   - **AI Integration:** Hugging Face NLP models for text summarization
2. **Functional Requirements:**
   - Extract text from uploaded PDF resumes.
   - Summarize extracted text using AI models.
   - Generate formatted resumes from JSON input.
   - Provide manual resume customization options.
   - Export resumes in PDF format.
3. **Constraints & Challenges:**
   - Ensuring compatibility with different resume formats.
   - Providing an intuitive UI for resume customization.
   - Maintaining text formatting consistency in PDF output.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**
    - User selects input method (Upload JSON, Upload PDF, or Manual Entry).
    - AI model processes text extraction and summarization.
    - User reviews and customizes resume details.
    - Resume is generated and available for download as a PDF.
2. **User Flow:**
    - **Step 1:** User chooses input method (JSON, PDF, Manual Entry).
    - **Step 2:** AI processes input (text extraction, summarization, or user input validation).
    - **Step 3:** User customizes fields if necessary.
    - **Step 4:** System generates a formatted PDF resume.
    - **Step 5:** User downloads the resume.
3. **UI/UX Considerations:**
    - Simple sidebar navigation for input selection.
    - Intuitive form fields for manual entry.
    - Download button for resume PDF.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Outcome
Sprint 1	UI & Backend Setup	High	6 hours	End of Day 1	vaishnavi	Streamlit, Flask	Functional UI
Sprint 1	AI Model Integration	High	4 hours	End of Day 1	Likhitha sree	Hugging Face	AI Text Processing
Sprint 2	Resume Customization UI	High	5 hours	Mid-Day 2	sravani	UI Ready	Editable Resume Fields
Sprint 2	PDF Generation Feature	High	3 hours	Mid-Day 2	Vaishnavi & likhitha	User Input	Downloadable PDF
Sprint 3	Testing & Enhancements	Medium	3 hours	End of Day 2	Akshitha	Completed Features	Bug-Free App
Sprint 3	Final Presentation	Low	1 hour	End of Day 2	Entire Team	Working Prototype	Demo-Ready

Phase-5: Project Development

Objective:

Implement core features of the Smart Resume Generator.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit
    - **Backend:** Flask/FastAPI
    - **Database:** JSON (for resume storage)
    - **AI:** Hugging Face pipeline for text summarization
2. **Development Process:**
    - Implement user input selection (JSON, PDF, Manual Entry).
    - Integrate AI for text summarization.
    - Develop a UI for manual resume customization.
    - Enable real-time PDF generation and downloading.

3. **Challenges & Fixes:**
    - **Challenge:** Handling inconsistent PDF formats.
        - **Fix:** Use pdfplumber for robust text extraction.
    - **Challenge:** Maintaining text structure in PDFs.
        - **Fix:** Utilize FPDF with proper text encoding.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the Smart Resume Generator works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Upload JSON and generate resume	Resume should be generated correctly	✅ Passed	Tester 1
TC-002	Functional Testing	Upload PDF and extract text	Extracted text should be displayed	✅ Passed	Tester 2
TC-003	AI Testing	Summarize extracted text	Summary should be concise	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed broken PDF downloads	Resume should be properly formatted	✅ Fixed	Developer
TC-005	Final Validation	Mobile responsiveness	UI adapts on all devices	❌ Failed - Needs Fixing	Tester 2
TC-006	Deployment Testing	Host on cloud platform	App accessible online	🚀 Deployed	DevOps

---

Final Submission

1. Project Report based on this documentation.
2. Demo Video (3-5 Minutes).
3. GitHub/Code Repository Link.
4. Presentation.
