AI Resume & Portfolio Builder: Detailed Project Plan

Problem Analysis

The core issue is that students struggle with creating professional, attractive resumes and portfolios that properly showcase their skills and projects. Generic templates don't highlight individual strengths effectively, leading to missed job and internship opportunities. The solution needs to be:

- **Automated**: Generates tailored content based on student data
- **Professional**: Creates attractive, ATS-friendly formats
- **Accessible**: Uses only free resources and tools
- **User-friendly**: Simple interface for non-technical users

Technical Architecture

Core Technologies

- **Frontend**: Streamlit (web interface)
- **Backend**: Python
- **AI Models**: Free APIs (Google Gemini, Hugging Face)
- **PDF Generation**: ReportLab or FPDF2
- **Data Storage**: JSON format for resume data
- **Deployment**: Streamlit Cloud (free tier)

Detailed Implementation Plan

Phase 1: Foundation Setup (Week 1)

Task 1.1: Environment Setup

```
# Required dependencies
dependencies = [
    "streamlit",
    "google-generativeai",  # For Gemini API
    "requests",            # For API calls
    "reportlab",           # PDF generation
    "json",               # Data handling
    "base64",             # File encoding
    "io",                 # File operations
]
```

Task 1.2: Project Structure

```
ai_resume_builder/
├── app.py                 # Main Streamlit app
├── components/
│   ├── data_input.py     # User input forms
│   ├── ai_generator.py   # AI content generation
│   ├── pdf_creator.py    # PDF generation
│   └── templates.py      # Resume templates
├── data/
│   ├── user_profiles/    # JSON user data
│   └── templates/        # Template files
├── utils/
│   ├── validators.py     # Input validation
│   └── helpers.py        # Utility functions
└── requirements.txt
```

Phase 2: Data Model & Input System (Week 2)

Task 2.1: JSON Schema Design [1] [2] [3]

```
{
  "personal": {
    "name": "string",
    "email": "string",
    "phone": "string",
    "location": "string",
    "linkedin": "string",
    "github": "string"
  },
  "summary": "string",
  "education": [
    {
      "institution": "string",
      "degree": "string",
      "field": "string",
      "gpa": "number",
      "graduation_date": "string",
      "relevant_courses": ["string"]
    }
  ],
  "experience": [
    {
      "company": "string",
      "position": "string",
      "duration": "string",
      "description": "string",
      "achievements": ["string"]
    }
  ],
  "projects": [
    {
      "name": "string",
      "description": "string",
      "technologies": ["string"],
      "github_url": "string",
      "demo_url": "string"
    }
  ],
  "skills": {
    "technical": ["string"],
    "languages": ["string"],
    "tools": ["string"]
  },
  "certifications": [
    {
      "name": "string",
      "issuer": "string",
      "date": "string"
    }
  ]
}
```

Task 2.2: Streamlit Input Forms [4] [5]

```
def create_input_form():
    st.title("AI Resume & Portfolio Builder")

    # Personal Information
    with st.expander("Personal Information", expanded=True):
        name = st.text_input("Full Name*")
        email = st.text_input("Email*")
        phone = st.text_input("Phone Number")
        # ... more fields

    # Education Section
    with st.expander("Education"):
        education_count = st.number_input("Number of Education Entries", 1, 5, 1)
        education_data = []
        for i in range(education_count):
            # Dynamic form creation

    # File Upload for existing resume
    uploaded_resume = st.file_uploader(
        "Upload existing resume (optional)",
        type=['pdf', 'docx', 'txt']
    )
```

Phase 3: AI Integration (Week 3)

Task 3.1: Free AI API Setup[6] [7] [8] [9]

```
# Google Gemini API (Free tier: 60 requests/minute)
import google.generativeai as genai

class AIGenerator:
    def __init__(self):
        self.gemini_key = st.secrets["GEMINI_API_KEY"]
        genai.configure(api_key=self.gemini_key)
        self.model = genai.GenerativeModel('gemini-pro')

    def generate_summary(self, user_data):
        prompt = f"""
        Create a professional resume summary for:
        Name: {user_data['name']}
        Field: {user_data['field']}
        Experience: {user_data['experience']}
        Skills: {user_data['skills']}

        Make it compelling and ATS-friendly, 2-3 sentences.
        """
        response = self.model.generate_content(prompt)
        return response.text
```

Task 3.2: Alternative Free APIs[10] [11] [12]

```
# Hugging Face Inference API (Free tier: 300 requests/hour)
def generate_with_huggingface(prompt, model="microsoft/DialoGPT-medium"):
    headers = {"Authorization": f"Bearer {hf_token}"}
    api_url = f"https://api-inference.huggingface.co/models/{model}"

    response = requests.post(api_url, headers=headers, json={"inputs": prompt})
    return response.json()
```

Phase 4: Content Generation Engine (Week 4)

Task 4.1: AI Content Generators

```
class ContentGenerator:
    def __init__(self, ai_model):
        self.ai = ai_model

    def enhance_experience_descriptions(self, experiences):
        enhanced = []
        for exp in experiences:
            prompt = f"""
            Enhance this work experience description using action verbs and quantifiable
            Position: {exp['position']}
            Company: {exp['company']}
            Description: {exp['description']}

            Return 3-4 bullet points that are ATS-friendly and impactful.
            """
            enhanced_desc = self.ai.generate_content(prompt)
            enhanced.append({**exp, "enhanced_description": enhanced_desc})
        return enhanced

    def generate_project_descriptions(self, projects):
        # Similar enhancement for projects
        pass

    def create_cover_letter(self, user_data, job_description):
        # Generate tailored cover letters
        pass
```

Phase 5: PDF Generation System (Week 5)

Task 5.1: Resume Templates with ReportLab [13] [14] [15]

```
from reportlab.lib.pagesizes import letter
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle

class PDFGenerator:
    def __init__(self):
        self.styles = getSampleStyleSheet()
        self.setup_custom_styles()

    def create_resume_pdf(self, user_data, template_style="modern"):
        buffer = io.BytesIO()
        doc = SimpleDocTemplate(buffer, pagesize=letter)
        story = []

        # Header with contact info
        story.append(self.create_header(user_data['personal']))

        # Professional Summary
        story.append(self.create_section("Professional Summary",
                                         user_data['summary']))

        # Experience Section
        story.append(self.create_experience_section(user_data['experience']))

        # Education, Skills, Projects sections...

        doc.build(story)
        buffer.seek(0)
        return buffer

    def create_portfolio_website(self, user_data):
        # Generate HTML/CSS for online portfolio
        pass
```

Task 5.2: Multiple Template Designs

```
class TemplateManager:
    def __init__(self):
        self.templates = {
            "modern": ModernTemplate(),
            "classic": ClassicTemplate(),
            "creative": CreativeTemplate(),
            "ats_friendly": ATSTemplate()
        }

    def generate_resume(self, user_data, template_name):
        template = self.templates[template_name]
        return template.create_pdf(user_data)
```

Phase 6: Portfolio Website Generation (Week 6)

Task 6.1: Dynamic Portfolio Pages[16] [17] [18]

```
def create_portfolio_page(user_data):
    # Create dynamic Streamlit pages
    st.set_page_config(
        page_title=f"{user_data['name']} - Portfolio",
        page_icon="□□",
        layout="wide"
    )

    # Header with photo and contact
    col1, col2 = st.columns([1, 2])
    with col1:
        if user_data.get('photo'):
            st.image(user_data['photo'], width=200)

    with col2:
        st.title(user_data['name'])
        st.subheader(user_data['title'])
        st.write(user_data['summary'])

    # Projects showcase with interactive elements
    create_projects_showcase(user_data['projects'])

    # Skills visualization
    create_skills_chart(user_data['skills'])
```

Phase 7: User Interface & Experience (Week 7)

Task 7.1: Streamlit App Architecture

```
def main():
    st.set_page_config(
        page_title="AI Resume & Portfolio Builder",
        page_icon="▯",
        layout="wide"
    )

    # Sidebar navigation
    with st.sidebar:
        page = st.selectbox("Choose Action", [
            "Create New Resume",
            "Upload Existing Resume",
            "Generate Portfolio",
            "Download Center",
            "Templates Gallery"
        ])

    if page == "Create New Resume":
        show_resume_builder()
    elif page == "Generate Portfolio":
        show_portfolio_builder()
    # ... other pages
```

Task 7.2: Progress Tracking & Validation

```
def show_progress_tracker(user_data):
    progress_items = [
        ("Personal Info", bool(user_data.get('personal'))),
        ("Education", bool(user_data.get('education'))),
        ("Experience", bool(user_data.get('experience'))),
        ("Skills", bool(user_data.get('skills'))),
        ("Projects", bool(user_data.get('projects')))
    ]

    completed = sum(1 for _, done in progress_items if done)
    progress = completed / len(progress_items)

    st.progress(progress)
    st.write(f"Profile Completion: {progress:.0%}")
```

Phase 8: Advanced Features (Week 8)

Task 8.1: ATS Optimization

```
def optimize_for_ats(resume_content, job_description):
    """Optimize resume content for Applicant Tracking Systems"""
    prompt = f"""
    Analyze this job description and optimize the resume content for ATS:

    Job Description: {job_description}
    Resume Content: {resume_content}

    Provide:
    1. Keyword matching score
    2. Recommended keywords to add
    3. Formatting suggestions
    4. Skills alignment recommendations
    """

    optimization_report = ai_model.generate_content(prompt)
    return optimization_report
```

Task 8.2: Multiple Export Formats

```
def export_resume(user_data, format_type):
    if format_type == "PDF":
        return create_pdf_resume(user_data)
    elif format_type == "JSON":
        return json.dumps(user_data, indent=2)
    elif format_type == "HTML":
        return create_html_portfolio(user_data)
    elif format_type == "LaTeX":
        return create_latex_resume(user_data)
```

Free Resources & APIs

AI Models (All Free Tiers)[7] [8] [9] [6]

1. **Google Gemini Pro**: 60 requests/minute free
2. **Hugging Face Inference API**: 300 requests/hour free
3. **Groq API**: Limited free tier
4. **Cohere API**: Free tier available

Development Tools

1. **Streamlit Cloud**: Free hosting for public apps
2. **GitHub**: Free repository hosting
3. **VS Code**: Free IDE
4. **Python**: Free programming language

Libraries (All Free)[14] [15] [13]

- `streamlit`: Web framework
- `google-generativeai`: Gemini API
- `reportlab`: PDF generation
- `requests`: HTTP requests
- `json`: Data handling
- `base64`: File encoding

Deployment Strategy

Local Development

```
# Setup virtual environment
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Run locally
streamlit run app.py
```

Cloud Deployment (Streamlit Cloud)

1. Push code to GitHub repository
2. Connect Streamlit Cloud to GitHub
3. Configure secrets for API keys
4. Deploy automatically

Implementation Timeline

Week	Focus Area	Key Deliverables
1	Foundation	Project setup, environment configuration
2	Data Model	JSON schema, input forms, validation
3	AI Integration	API setup, content generation functions
4	Content Engine	Experience enhancement, project descriptions
5	PDF Generation	Resume templates, PDF creation system
6	Portfolio Site	Dynamic web portfolio generation
7	User Interface	Complete Streamlit app, navigation
8	Advanced Features	ATS optimization, multiple export formats

Budget Considerations

Total Cost: $0 (using only free tiers)

- Google Gemini API: Free (60 requests/minute)
- Hugging Face API: Free (300 requests/hour)
- Streamlit Cloud: Free hosting
- All Python libraries: Open source and free

Scalability Notes: If usage exceeds free tiers, consider:

- Google Gemini Pro: $0.001 per 1K characters
- Hugging Face Pro: $9/month for higher limits
- Streamlit Cloud Teams: $20/month for private apps

Success Metrics

1. **User Engagement**: Time spent on platform, completion rates
2. **Quality Metrics**: AI-generated content relevance scores
3. **ATS Compatibility**: Resume parsing success rates
4. **User Satisfaction**: Feedback scores, feature usage
5. **Performance**: Response times, error rates

This comprehensive plan provides a complete roadmap for building an AI-powered resume and portfolio builder using only free resources and Python, with Streamlit as the web framework and various free AI APIs for content generation.

⁂

1. https://angelo-lima.fr/en/json-resume-standardized-cv-format-developers-automation/
2. https://github.com/jsonresume
3. https://jsonresume.org/schema
4. https://discuss.streamlit.io/t/new-component-a-customizable-file-uploader-component-for-streamlit/96271
5. https://docs.streamlit.io/develop/api-reference/widgets/st.file_uploader
6. https://www.reddit.com/r/ChatGPTCoding/comments/1avftzm/free_openai_api_alternatives/
7. https://ai.google.dev/gemini-api/docs/pricing
8. https://apidog.com/blog/google-gemini-api-key-for-free/
9. https://dev.to/garciadiazjaime/gemini-api-the-free-tier-that-makes-developers-happy-28nk
10. https://dev.to/ajmal_hasan/how-to-use-hugging-face-ai-models-as-an-api-5eja
11. https://huggingface.co/learn/cookbook/en/enterprise_hub_serverless_inference_api
12. https://www.reddit.com/r/LocalLLaMA/comments/1fi90kw/free_hugging_face_inference_api_now_clearly_lists/
13. https://www.nutrient.io/blog/top-10-ways-to-generate-pdfs-in-python/

14. https://templated.io/blog/generate-pdfs-in-python-with-libraries/
15. https://apitemplate.io/blog/a-guide-to-generate-pdfs-in-python/
16. https://www.youtube.com/watch?v=rviQtjkxQQY
17. https://blog.streamlit.io/land-your-dream-job-build-your-portfolio-with-streamlit/
18. https://www.linkedin.com/posts/abhisheak-saraswat_building-a-professional-web-portfolio-with-activity-7256579348827185153-kZ7m
19. https://www.linkedin.com/pulse/discover-top-5-free-ai-resume-making-tools-2024-expert-nikhil-kumar-ihsvc
20. https://www.geeksforgeeks.org/python/resume-generator-app-using-python/
21. https://www.resume-now.com
22. https://pub.towardsai.net/create-a-smart-resume-builder-with-python-and-gpt-4-step-by-step-tutorial-39c5a092a018
23. https://www.myperfectresume.com
24. https://pypi.org/project/pyresume/
25. https://www.careerflow.ai/resume-builder
26. https://www.youtube.com/watch?v=iLE6_3Sp0CA
27. https://www.youtube.com/watch?v=WvFp7OlTxLE
28. https://www.rezi.ai
29. https://github.com/orgs/community/discussions/155868
30. https://resumebuild.com/resume/examples/python-developer/
31. https://www.visualcv.com/ai-resume-builder/
32. https://discuss.streamlit.io/t/create-your-portfolio-in-streamlit-using-airtable-as-cms-and-materializecss-for-a-modern-look/78685
33. https://github.com/SudeepAcharjee/Resume-Builder
34. https://www.resumebuild.ai
35. https://python.plainenglish.io/improve-your-machine-learning-or-data-science-portfolio-with-app-development-and-deployment-made-4599323a5901
36. https://github.com/mudler/LocalAI
37. https://community.openai.com/t/free-solutions-for-testing-and-developing-with-the-api/310654
38. https://www.reddit.com/r/GoogleGeminiAI/comments/1ky3sgg/gemini_api_billing_does_enabling_paid_tier/
39. https://semaphore.io/blog/localai
40. https://www.bluebash.co/blog/ultimate-guide-to-using-hugging-face-inference-api/
41. https://wotnot.io/blog/openai-alternatives
42. https://huggingface.co/docs/inference-providers/en/index
43. https://ai.google.dev/gemini-api/docs/rate-limits
44. https://www.youtube.com/watch?v=G7aXgKlhbGk
45. https://www.postman.com/postman-student-programs/hugging-face-inference-api-free/overview
46. https://cloud.google.com/free
47. https://www.edenai.co/post/best-openai-api-alternatives-in-2024

48. https://discuss.huggingface.co/t/free-models-using-api/55532
49. https://www.scribd.com/document/595154970/Create-Your-Own-Resume-Data-in-JSON-Format
50. https://realpython.com/creating-modifying-pdf/
51. https://pypi.org/project/streamlit-chunk-file-uploader/
52. https://stackoverflow.com/questions/79359964/use-streamlit-file-uploader-component-to-add-a-file-into-snowflake-stage
53. https://docs.rxresu.me/product-guides/exporting-your-resume-as-json
54. https://discuss.streamlit.io/t/new-component-chunk-file-uploader-break-through-the-upload-size-limit/61117
55. https://www.reportlab.com/docs/reportlab-userguide.pdf
56. https://www.youtube.com/watch?v=awsjo_1tqlM
57. https://pypi.org/project/reportlab/
58. https://www.visualcv.com/resume-skills/json/
59. https://docs.reportlab.com
60. https://discuss.streamlit.io/t/file-uploading-and-reading-using-st-file-uploader/31897