**Hackathon Project Phases Template** for SmartResume Generator: Customized Resumes for Every opportunity.

**Hackathon Project Phases Template Project Title:**

SmartResume Generator: Customized Resumes for Every opportunity.

**Team Name:**

Goal diggers

**Team Members:**

* Adepu Mounika
* Are Sravya
* Basutkar Sony
* Chakali Bhavani

# Phase-1: Brainstorming & Ideation

**Objective:**

Develop an AI-powered **resume generator** that creates **professional, compact, and visually appealing resumes** using **Streamlit** for UI and **Python-docx** for document generation.

### **Key Points:**

1. **Problem Statement:**
   * Job seekers struggle to create professional, well-structured resumes that stand out.
   * Manually formatting resumes is time-consuming, and many lack design expertise.
2. **Proposed Solution:**
   * A **Streamlit-based SmartResume Generator** that auto-generates **customized, single-page resumes.**
   * The app allows users to enter details, upload a profile picture, and download a formatted resume in **.docx** format.
3. **Target Users:**
   * Freshers and professionals needing quick and professional resumes.
   * HR professionals and career coaches looking for a **resume-building tool**.
4. **Expected Outcome:**
   * A functional **AI-powered resume builder** that generates **well-structured resumes** in seconds.

# Phase-2: Requirement Analysis

**Objective:**

Define technical and functional requirements.

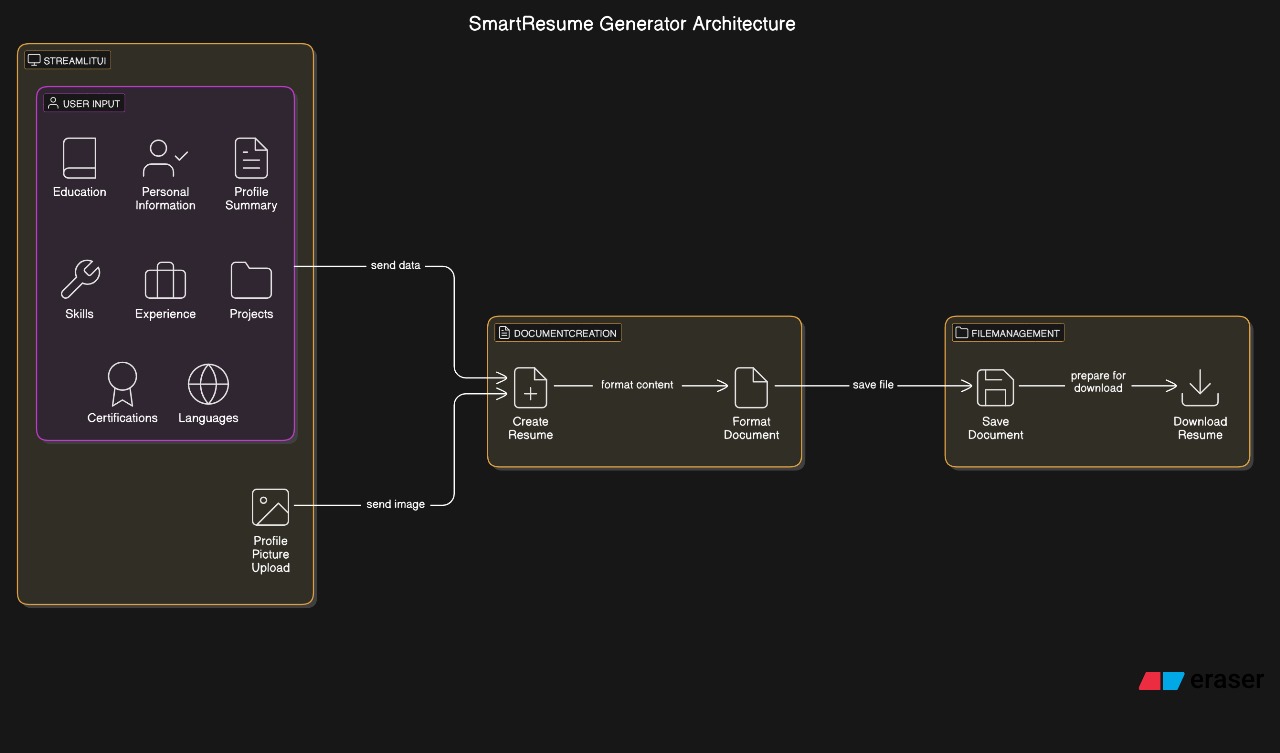
### **Key Points:**

1. **Technical Requirements:**
   * **Programming Language:** Python
   * **Frontend:** Streamlit
   * **Backend:** Python-docx for document generation
   * **Database:** Not required (data is user-input based)
2. **Functional Requirements:**
   * Users enter **personal details, experience, education, skills, and certifications**.
   * A **.docx file is generated with professional formatting**, including a **profile picture**.
   * Resumes are **customized based on user preferences** (e.g., skill-based, experience-focused).
3. **Constraints & Challenges:**
   * Ensuring **resume formatting remains compact and professional**.
   * Handling **varied input lengths** while keeping a **one-page format**.
   * Allowing **profile picture integration** without layout distortion.

# Phase-3: Project Design

**Objective:**

Develop system architecture and user flow for the Smart Resume Generator.



**Key Points:**

1. **System Architecture:**
   * User enters data via Streamlit UI.
   * The app processes data and formats it using Python-docx.
   * The resume is generated and provided as a downloadable .docx file.
2. **User Flow:**
   * Step 1: User enters details via UI.
   * Step 2: The backend processes input and formats it into a professional resume template.
   * Step 3: The user downloads the resume in .docx format.
3. **UI/UX Considerations:**
   * Simple, user-friendly interface with easy data entry fields.
   * Preview option before downloading.
   * Professional templates with structured formatting.

# Phase-4: Project Planning (Agile Methodologies)

**Objective:**

Break down development tasks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Task** | **Priority** | **Duration** | **Deadline** | **Assigned To** | **Outcome** |
| **Sprint 1** | UI Development (Streamlit) | 🔴 High | 5 hours | Day 1 | Developer 1 | Functional UI |
| **Sprint 1** | Resume Formatting (Python-docx) | 🔴 High | 6 hours | Day 1 | Developer 2 | Resume template ready |
| **Sprint 2** | Profile Picture Integration | 🟡 Medium | 3 hours | Day 2 | Developer 3 | Image correctly placed |
| **Sprint 2** | Download Functionality | 🔴 High | 2 hours | Day 2 | Developer 1 | Downloadable .docx file |
| **Sprint 3** | Testing & UI Refinement | 🟢 Low | 3 hours | Day 3 | QA Team | Bug-free UI |
| **Sprint 3** | Final Deployment | 🟢 Low | 2 hours | Day 3 | Entire Team | Live Project |

# Phase-5: Project Development

### **Objective:**

Implement the core features.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Python-docx
   * **Programming Language:** Python
2. **Development Process:**
   * **Step 1:** Build the Streamlit UI.
   * **Step 2:** Integrate Python-docx for resume formatting.
   * **Step 3:** Implement a **download feature.**
   * **Step 4:** Optimize **profile picture integration.**
3. **Challenges & Fixes:**
   * **Challenge:** Handling long input texts.
     + **Fix:** Auto-truncate or adjust font size.
   * **Challenge:** Profile picture distortion.
     + **Fix:** Standardize image dimensions.

# Phase-6: Functional & Performance Testing Objective:

Ensure the app works as expected.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** |
| TC-001 | Functional Testing | Generate a resume with all fields | A properly formatted .docx file should be created | ✅ Passed |
| TC-002 | Functional Testing | Upload profile picture | Profile picture should be placed correctly | ✅ Passed |
| TC-003 | Performance Testing | Resume download speed | Resume should generate in <3 sec | ⚠ Needs Optimization |
| TC-004 | UI/UX Testing | Verify text formatting | Text should be properly aligned and readable | ✅ Passed |
| TC-005 | Bug Fixes | Handle empty fields | Resume should not break on missing fields | ✅ Fixed |

**Final Submission**

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**