**Hackathon Project Phases Template** for the **Smart Resume Generator.**

# Hackathon Project Phases Template

**Project Title:**

**Smart Resume Generator.**

**Team Name:**

AI Artist

**Team Members:**

* B.Tharak.
* G.Nagaraju.
* B.Balaraju.
* M.Jagirdhar.
* M.Akshitha.

## Phase-1: Brainstorming & Ideation

**Objective:**

* To empower users to create professional, impactful resumes quickly and easily, regardless of their design skills or experience.
* To democratize resume building by providing access to sophisticated resume templates and AI-driven content suggestions for everyone.
* To help job seekers significantly increase their chances of landing interviews by optimizing their resumes for both applicant tracking systems (ATS) and human recruiters.
* To streamline the resume creation process, allowing users to focus on highlighting their skills and achievements rather than struggling with formatting and design.
* To provide users with data-driven insights and personalized recommendations to improve their resume content and presentation

**Key Points:**

**Focusing on Features & Technology:**

* To leverage artificial intelligence to analyze job descriptions and suggest relevant skills, keywords, and accomplishments for user resumes.

**Focusing on Competitive Advantage:**

* To become the leading AI-powered resume generator by providing the most accurate, relevant, and actionable resume suggestions

**Focusing on Business Goals:**

* To increase user engagement and retention through continuous improvement and the addition of new features and functionality

**Focusing on Innovation:**

* To continually innovate and improve our resume generator by incorporating the latest advancements in AI, natural language processing, and resume writing best practices.

## Phase-2: Requirement Analysis

**Objective:**

Define the technical and functional requirements for the AutoSage App.

**Key Points:**

**II. Functional Requirements:**

**These describe *what* the system should do. Resume Creation & Editing:**

**II. Functional Requirements:**

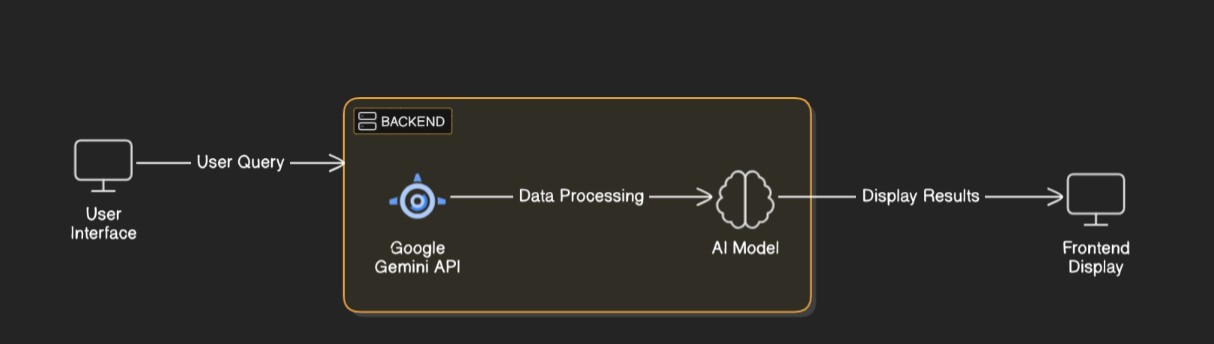
**These describe *what* the system should do.**

* **Resume Creation & Editing:**
  + Management: Enable users to add, remove, and rearrange resume sections (e.g.,
  + Template Selection: Allow users to choose from a variety of customizable resume templates.
  + Section Summary, Experience, Education, Skills).
  + Text Editing: Provide a rich text editor for formatting and modifying text within the resume.
  + Drag-and-Drop Functionality: (Optional, but desirable) Allow users to rearrange elements using drag-and-drop.
  + Version Control: Allow users to save and manage multiple versions of their resume.
  + ATS Optimization: Provide guidance and tools to ensure resumes are ATS-friendly.
* **AI-Powered Assistance:**
  + Keyword Suggestion: Suggest relevant keywords based on job descriptions or industry trends.
  + Skill Matching: Match user skills to job requirements and suggest relevant skills to add.
  + Content Generation: Offer AI-generated content suggestions for resume sections based on user input. Resume Analysis; Analyze existing resumes and provide
* **Integration & Export:**
  + Job Board Integration: Allow users to import job descriptions from popular job boards.
  + ATS Integration: (Ideally) Allow users to directly submit resumes to certain ATS systems.
  + File Export: Export resumes in various formats (e.g., PDF, DOCX, TXT).
  + Sharing Options: Enable users to share their resumes via email or social media

## Phase-3: Project Design

**Objective:**

Develop the architecture and user flow of the application.



**Key Points:**

1. **AI-Powered Assistance Module:**

**Keyword Suggestion Engine:** Suggests relevant keywords based on job descriptions and industry trends.

**Skill Matching Engine:** Matches user skills to job requirements and suggests skills to add.

**Content Generation Engine:** Generates content suggestions for resume sections based on user input, leveraging large language models.

**Resume Analyzer:** Analyzes existing resumes and provides feedback on areas for improvement.

1. **User Flow:**

* 1. Step 1: User enters a query (e.g., "Best motorcycles under ₹1 lakh").

○ Step 2: The backend **calls the Gemini Flash API** to retrieve vehicle data.

○ Step 3: The app processes the data and **displays results** in an easy-to-read format.

1. **UI/UX Considerations:**

* 1. **Minimalist, user-friendly interface** for seamless navigation.

○ **Filters for price, mileage, and features**.

○ **Dark & light mode** for better user experience.

## Phase-4: Project Planning (Agile Methodologies)

**Objective:**

**Project Overview:**

* **Project Name:** Smart Resume Generator
* **Project Goal:** Develop a user-friendly, AI-powered resume generator that empowers job seekers to create professional, ATS-optimized resumes efficiently.
* **Project Manager:** [Name]
* **Start Date:** [Date]
* **End Date:** [Date] (Estimated - subject to change)
* **Budget:** [Amount] (Estimated - subject to change)

**II. Project Phases:**

1. **Initiation & Planning (2-4 Weeks):**
   * **Tasks:**
     + Define project scope, goals, and objectives in detail.
     + Conduct thorough requirements gathering and analysis.
     + Develop a detailed project plan, including tasks, timelines, resources, and budget.
     + Identify key stakeholders and establish communication channels.
     + Perform a risk assessment and develop a risk management plan.
     + Select the appropriate development methodology (Agile/Scrum).
     + Define success metrics.
   * **Deliverables:**
     + Project Charter
     + Requirements Document
     + Project Plan
     + Risk Management Plan
     + Communication Plan
   * **Resources:** Project Manager, Business Analyst, Key Stakeholders
2. **Design (2-4 Weeks):**
   * **Tasks:**
     + Develop the system architecture, including the frontend, backend, and database.
     + Design the user interface (UI) and user experience (UX).
     + Define the API specifications for integrating with external services.
     + Choose the technology stack.
     + Design the AI models for keyword suggestion, skill matching, and content generation.
     + Create wireframes and prototypes of the user interface.
     + Conduct design reviews with stakeholders.
   * **Deliverables:**
     + System Architecture Diagram
     + UI/UX Design Specifications
     + API Specifications
     + Database Schema
     + AI Model Design Documents
     + Wireframes/Prototypes
   * **Resources:** UI/UX Designer, Software Architects, Data Scientists, Database Administrator
3. **Development (8-12 Weeks):**
   * **Tasks:**
     + Develop the frontend user interface.
     + Develop the backend API and business logic.
     + Implement the database schema.
     + Implement the AI models.
     + Integrate with external services (job boards, ATS systems).
     + Write unit tests and integration tests.
     + Conduct code reviews.
   * **Deliverables:**
     + Frontend Code
     + Backend Code
     + Database Implementation
     + AI Model Implementations
     + API Integrations
     + Unit Tests
     + Integration Tests
   * **Resources:** Frontend Developers, Backend Developers, AI/ML Engineers, Database Administrator, Quality Assurance Engineers
4. **Testing (2-4 Weeks):**
   * **Tasks:**
     + Conduct functional testing to ensure all features work as expected.
     + Conduct performance testing to ensure the system can handle the expected load.
     + Conduct security testing to identify and address vulnerabilities.
     + Conduct usability testing to ensure the user interface is easy to use.
     + Conduct accessibility testing to ensure the system is accessible to users with disabilities.
     + Fix bugs and retest.
   * **Deliverables:**
     + Test Cases
     + Test Results
     + Bug Reports
     + Fixed Code
   * **Resources:** Quality Assurance Engineers, Usability Testers, Security Testers
5. **Deployment (1-2 Weeks):**
   * **Tasks:**
     + Prepare the production environment.
     + Deploy the application to the production environment.
     + Configure the system for optimal performance.
     + Conduct final testing in the production environment.
     + Train users on how to use the system.
     + Prepare documentation.
   * **Deliverables:**
     + Deployed Application
     + Configuration Documentation
     + User Training Materials
     + Deployment Plan
   * **Resources:** DevOps Engineers, System Administrators, Technical Writers
6. **Maintenance & Support (Ongoing):**
   * **Tasks:**
     + Monitor the system for performance and security issues.
     + Fix bugs and address user support requests.
     + Deploy updates and new features.
     + Maintain documentation.
     + Collect user feedback and identify areas for improvement.
   * **Deliverables:**
     + Bug Fixes
     + Updates and New Features
     + Updated Documentation
   * **Resources:** Support Team, Developers, DevOps Engineers

## Phase-5: Project Development

**Objective:**

Implement core features of the AutoSage App.

**Key Points:**

**Development Environment Setup:**

1. **Hardware & Software:**
   * Developers need appropriate laptops/desktops with sufficient processing power, memory, and storage.
   * Operating Systems: (Windows, macOS, Linux - chosen based on team preference)
   * IDE: Integrated Development Environment (e.g., VS Code, IntelliJ IDEA, PyCharm). Configure IDE with necessary plugins for linting, code formatting, and debugging.
   * Version Control: Git is essential. Setup Git with GitHub, GitLab, or Bitbucket for collaborative coding and version tracking.
   * Containerization (Docker): Install Docker and Docker Compose for creating consistent development environments.
2. Environment Consistency: Use Docker and Docker Compose to create consistent development environments across all team members. This ensures that the code works the same way on everyone's machine.
3. Secrets Management: Use environment variables or a secrets management tool to store sensitive information (API keys, database passwords) securely and avoid hardcoding them in the code.
4. Code Style & Linting: Establish and enforce a consistent coding style using linters (e.g., ESLint, Pylint) and code formatters (e.g., Prettier, Black). This helps improve code readability and maintainability.

**II. Coding Standards & Best Practices:**

1. Clean Code Principles: Follow clean code principles to write code that is easy to read, understand, and maintain. Focus on meaningful names, small functions, and clear separation of concerns.
2. SOLID Principles: Apply the SOLID principles of object-oriented design (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion) to create modular and maintainable code.
3. DRY (Don't Repeat Yourself): Avoid code duplication by creating reusable functions and components.
4. YAGNI (You Ain't Gonna Need It): Avoid adding unnecessary features or complexity to the code.
5. Comments & Documentation: Write clear and concise comments to explain the purpose of the code. Generate API documentation using tools like Swagger or JSDoc**.**

**III. Development Workflow (Example using Agile/Scrum):**

1. Sprint Planning: At the beginning of each sprint, the team plans which user stories and tasks will be worked on during the sprint.
2. Daily Stand-up Meetings: The team meets briefly each day to discuss progress, roadblocks, and plans for the day.
3. Task Assignment: Tasks are assigned to individual developers.
4. **Coding & Testing:**
   * Developers write code to implement the assigned tasks.
   * Write unit tests for each function and component to ensure they work correctly.
   * Run unit tests frequently to catch bugs early.
5. Code Reviews: Before code is merged into the main branch, it is reviewed by other developers to ensure code quality and identify potential issues. Use pull requests in Git for code reviews.
6. Integration Testing: After code is merged, integration tests are run to ensure that different parts of the system work together correctly.
7. Sprint Review: At the end of the sprint, the team demonstrates the completed work to stakeholders and gathers feedback.
8. Sprint Retrospective: The team reflects on the sprint and identifies areas for improvement in the development process

## Phase-6: Functional & Performance Testing

**Objective:**

**I. Functional Testing:**

Functional testing verifies that each component and feature of the application works as expected, according to the requirements specification.

**A. Test Areas & Scenarios:**

1. **User Management:**
   * **Registration:**
     + Verify successful registration with valid credentials.
     + Verify error messages for invalid credentials (e.g., missing fields, invalid email format, weak password).
     + Verify handling of duplicate email addresses.
     + Verify email confirmation process (if applicable).
   * **Login:**
     + Verify successful login with valid credentials.
     + Verify error messages for invalid credentials.
     + Verify "Forgot Password" functionality and password reset process.
     + Verify "Remember Me" functionality.
     + Test account lockout after multiple failed login attempts.
   * **Profile Management:**
     + Verify the ability to update user profile information (name, email, contact details).
     + Verify validation of input fields.
     + Verify saving changes to the database.
     + Test changing password functionality.
   * **Subscription Management (if applicable):**
     + Verify the ability to subscribe to different plans.
     + Verify payment processing integration.
     + Verify plan upgrade/downgrade functionality.
     + Verify cancellation process.
2. **Resume Builder:**
   * **Template Selection:**
     + Verify the ability to browse and select from different resume templates.
     + Verify that templates are displayed correctly.
   * **Section Management:**
     + Verify the ability to add, remove, and rearrange resume sections.
     + Verify the correct rendering of sections in the resume preview.
   * **Text Editor:**
     + Verify the ability to format text (bold, italic, underline, font size, font color).
     + Verify the ability to insert bullet points and numbered lists.
     + Verify the ability to copy and paste text.
     + Verify auto-saving functionality.
     + Verify spell check functionality.
   * **ATS Optimization:**
     + Verify that ATS optimization tips are displayed correctly.
     + Verify that ATS compatibility checks are performed.
     + Verify that issues identified by the ATS compatibility checker are flagged.
   * **Version Control:**
     + Verify the ability to save multiple versions of the resume.
     + Verify the ability to revert to previous versions.
     + Verify the ability to compare different versions.
   * **Resume Preview:**
     + Verify that the resume is displayed correctly in the preview.
     + Verify that the preview reflects all changes made in the editor.
   * **Download/Export:**
     + Verify the ability to download the resume in different formats (PDF, DOCX, TXT).
     + Verify that the downloaded resume is formatted correctly.
3. **AI-Powered Assistance:**
   * **Keyword Suggestion:**
     + Verify that keyword suggestions are relevant to the job description or industry.
     + Verify that the suggestions are displayed correctly.
     + Test the keyword suggestion engine with various job descriptions and industry types.
   * **Skill Matching:**
     + Verify that skill matches are accurate and relevant.
     + Verify that the matching engine identifies missing skills.
     + Test skill matching with different user profiles and job requirements.
   * **Content Generation:**
     + Verify that the generated content is grammatically correct and relevant to the resume section.
     + Verify the quality and originality of the generated content.
     + Test content generation with different resume sections and user inputs.
   * **Resume Analysis:**
     + Verify that the resume analysis provides accurate feedback on areas for improvement.
     + Verify that issues identified by the analyzer are flagged.
     + Test the resume analyzer with various resume samples.

## Final Submission

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