**Hackathon Project Phases** for the **SWIFT-APPLY** project.

# Hackathon Project Phases

**Project Title: SWIFT-APPLY**

**Accelerating Careers with AI-Powered Applications using Gemini AI API**

**Team Name:**

Code Ninja

**Team Members:**

* Macherla Yogesh
* K Abhinav
* L Venkata Manideep
* N Sai Ashish
* P Ram Charan

## Phase-1: Brainstorming & Ideation

**Objective:**

Develop an innovative platform leveraging AI technology to streamline the job application process and empower users in their career advancement journey.

**Key Points:**

1. **Problem Statement:**

* Recent graduates find it challenging to create tailored resumes and cover letters for entry-level jobs, hindering their chances in a competitive market.

1. **Proposed Solution:**

* Swift-apply generates customized, polished application materials based on users' academic achievements and skills, helping graduates stand out and confidently launch their careers

1. **Target Users:**

* **Recent Graduates:**Individuals who have recently completed their degrees and are entering the job market for the first time.
* **Students in Transition:**Undergraduates nearing graduation who are preparing to apply for entry-level positions.
* **Career Changers:**Individuals looking to switch fields and needing to create new resumes and cover letters tailored to different industries.
* **Internship Seekers:**Students or recent graduates seeking internships to gain experience and enhance their resumes.
* **Job Seekers in Competitive Fields:**Individuals applying for positions in highly competitive industries where standout application materials are crucial.
* **Young Professionals:**Those with limited work experience who need guidance in presenting their skills and achievements effectively.

.

1. **Expected Outcome:**

* Users will receive professionally tailored resumes and cover letters that significantly increase their chances of securing interviews and landing entry-level positions in their chosen fields.

## Phase-2: Requirement Analysis

**Objective:**

Define the technical and functional requirements for the Swift-apply App.

**Key Points:**

1. **Technical Requirements:** 
   * Programming Language: **Python.**

○ Backend: **Python.**

○ Frontend: **Streamlit Web Framework.**

○ Database: **SQL lite.**

1. **Functional Requirements:**

* **Resume generation:** The platform allows users to input their academic achievements, work experience, skills, and other relevant information.
* **Cover Letter Generation:** The platform generates a personalized cover letter that highlights the user’s qualifications and enthusiasm for the position.
  + **Interview Preparation:** - The platform must offer a library of common interview questions tailored to various industries and roles.

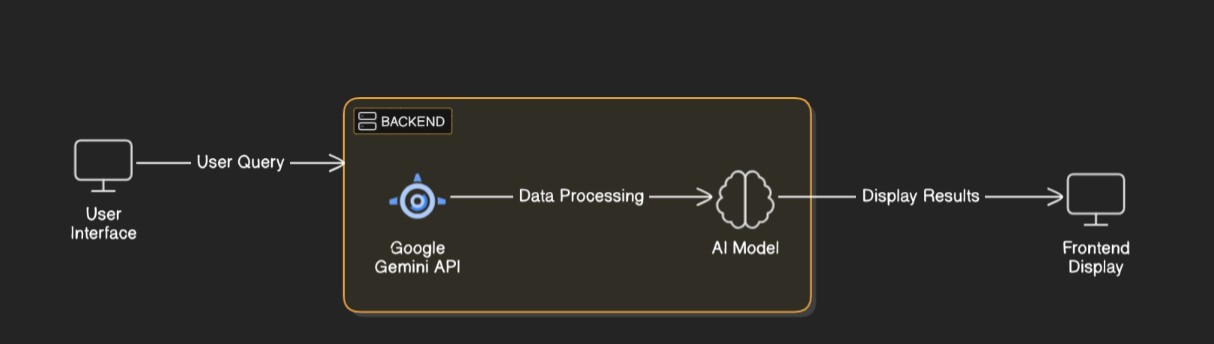
1. **Constraints & Challenges:**

* + Compliance with regulations such as GDPR and CCPA must be maintained.
  + Rapid development may lead to technical debt, which can complicate future updates and maintenance.
  + Differentiating Swift-apply and demonstrating its unique value proposition will be essential.

## Phase-3: Project Design

**Objective:**

Develop the architecture and user flow of the application.



**Key Points:**

1. **System Architecture:**

* User enters skills and education qualification via UI.

○ Query is processed using **Google Gemini API**.

○ AI model fetches and processes the data.

○ The frontend displays resume and cover letter.

1. **User Flow:**

* **Step 1:** User enters their skills and education qualifications (e.g., "BTech, Python, Java").
* **Step 2:** The backend calls the Gemini Flash API to retrieve relevant information based on the user's input.
* **Step 3:** The app processes the data received from the API and displays the results in an easy-to-read format for the user.

1. **UI/UX Considerations:**

* + - **Minimalist, user-friendly interface** for seamless navigation.
    - Show only necessary fields at each step to avoid overwhelming the user.
    - Minimalist design with ample space for readability.
    - Provide dark mode/light mode according to the system default.

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

**Objective:**

Break down development tasks for efficient completion.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Task** | **Priority** | **Duration** | **Deadline** | **Assigned To** | **Dependencies** | **Expected**  **Outcome** |
| Sprint 1 | Environment Setup  & API Integration | 🔴 High | 6 hours  (Day 1) | End of Day  1 | Sai Ashish | Gemini API Key,  Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day  1 | Ram Charan | API response format finalized | Basic UI with input fields |
| Sprint 2 | Resume PDF generation | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Yogesh | API response, UI elements ready | Resume PDF |
| Sprint 2 | Error Handling &  Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Manideep | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI  Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Abhinav | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation  & Deployment | 🟢 Low | 1 hour  (Day 2) | End of Day  2 | Entire Team | Working prototype | Demo-ready project |

**Sprint Planning with Priorities**

**Sprint 1 – Setup & Integration (Day 1)**

**(**🔴 **High Priority)** Set up the **environment** & install dependencies.

**(**🔴 **High Priority)** Integrate **Google Gemini API**.

**(**🟡 **Medium Priority)** Build a **basic UI with input fields**.

**Sprint 2 – Core Features & Debugging (Day 2)**

**(**🔴 **High Priority)** Implement **Resume PDF Generation**

**(**🔴 **High Priority)** Debug API issues & handle **errors in queries**.

**Sprint 3 – Sprint 3 – Testing, Enhancements & Submission (Day 2)**

**(**🟡 **Medium Priority)** Test API responses, refine UI, & fix UI bugs.

**(**🟢 **Low Priority)** Final **demo preparation & deployment**.

## Phase-5: Project Development

**Objective:**

Implement core features of the Resume builder.

**Key Points:**

1. **Technology Stack Used:**

* 1. **Frontend:** Streamlit

○ **Backend:** Google Gemini Flash API

○ **Programming Language:** Python

1. **Development Process:**

* 1. Set up the project repository and development environment
* Create interactive forms for users to input their career details, skills, and job preferences.
* Submitting user data for resume and cover letter generation.
* Deploy the application to a live environment.

1. **Challenges & Fixes:**

* **Challenge:** Integration Complexities.

**Fix:** Utilize a well-defined API structure and ensure proper documentation is followed.

* **Challenge:** The response time from the Generative AI model can be slow, impacting user experience.

**Fix:** Implement caching mechanisms to store frequent requests and responses.

## Phase-6: Functional & Performance Testing

**Objective:**

Ensure that the Swift-apply App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional  Testing | Login database | Database integration  test | ✅ Passed | charan |
| TC-002 | Functional  Testing | Pdf format generation with different template | Seasonal tips should be provided. | Need additional templates | yogesh |
| TC-003 | Performance  Testing | API response time under  500ms | API should return results quickly. | ⚠ Needs Optimization | abhinav |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | ✅ Fixed | Developer |
| TC-005 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | ❌ Failed - UI broken on mobile | ashish |
| TC-006 | Deployment  Testing | Host the app using  Streamlit Sharing | App should be accessible online. | 🚀 Deployed | DevOps |