**Hackathon Project Phases Template** for the **StudBud Website**project.



Hackathon Project Phases Template

# Project Title:

**StudBud - AI Study Planner Using Streamlit and Gemini AI**

# Team Name:

Smarticles

# Team Members:

* Charitha
* Ashrita
* Asha Jyothi



# Phase-1: Brainstorming & Ideation

**Objective:**

Develop an AI-powered study planner using Gemini to help users create customized study schedules, prioritize tasks, and prepare for exams effectively.

**Key Points:**

**Problem Statement:**

* + Many students struggle to organize their study time effectively, leading to missed deadlines and poor exam preparation.
  + Students require personalized study plans that consider their workload, distractions, and available study hours.

**Proposed Solution:**

* + - An AI-powered application using Gemini to generate real-time study timetables based on user inputs such as tasks, study habits, preferences, and personal goals.
    - The app offers insights on time management, distraction handling, and study techniques.

**Target Users:**

* + - Students looking for assistance in managing their study schedules and academic workload.
    - Learners preparing for exams who need structured study plans.
    - Eco-conscious students interested in optimizing their study hours for better efficiency and balance.

**Expected Outcome:**

* A functional AI-powered study planner that provides users with personalized study schedules, prioritization of tasks, and actionable insights based on their preferences.



# Phase-2: Requirement Analysis

## Objective:

Define the technical and functional requirements for the StudBud .

**Key Points:**

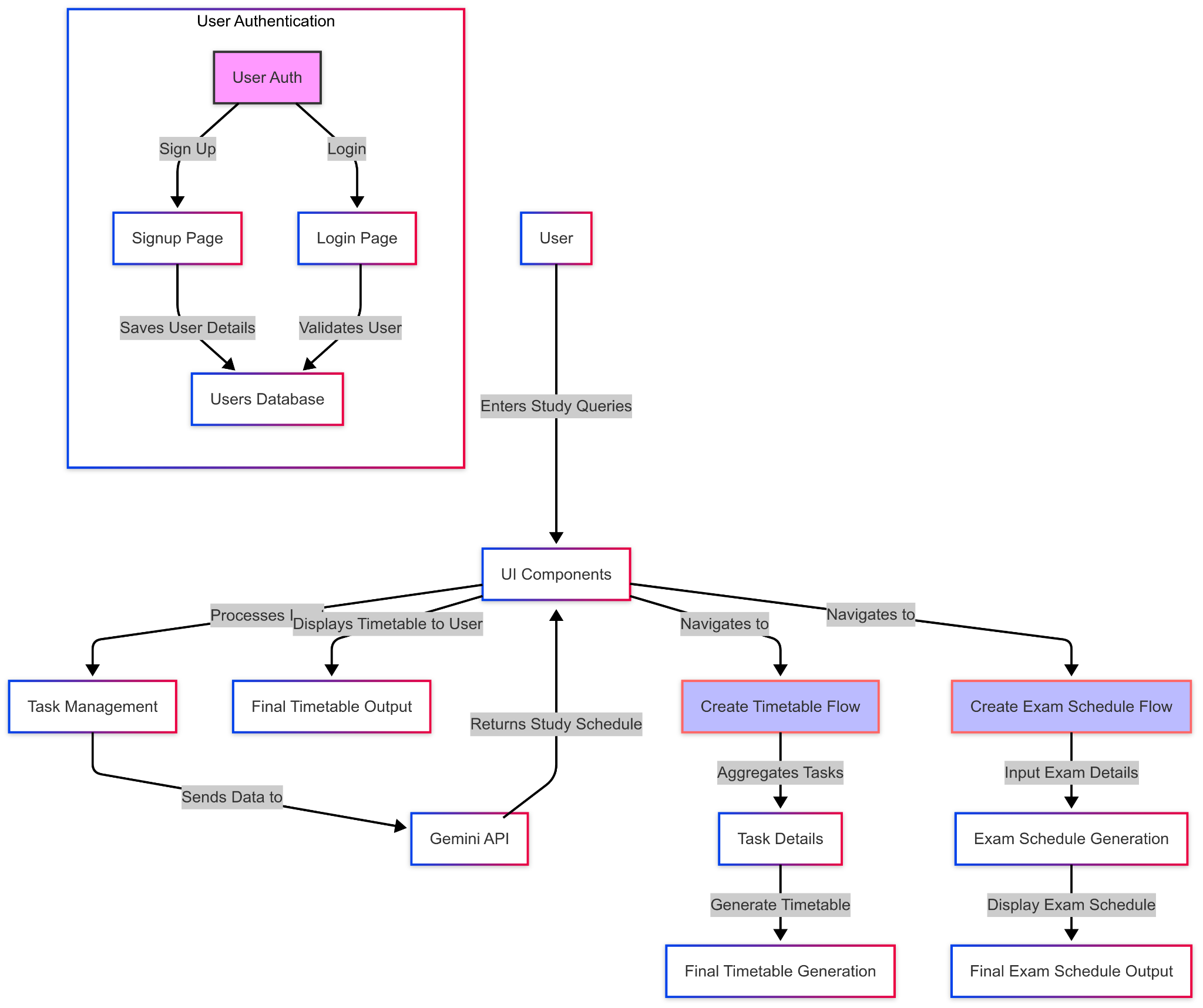
1. **Technical Requirements:**
   * Programming Language: Python
   * Backend: Google Gemini API
   * Frontend: Streamlit Web Framework
   * Database: Not required initially (API-based queries)
2. **Functional Requirements:**
   * Ability to fetch user-defined tasks and study preferences.
   * Display generated timetables and study recommendations in an intuitive UI.
   * Provide real-time updates on tasks and study schedules.
   * Allow users to adjust plans based on study habits, preferences, and distractions.
3. **Constraints & Challenges:**
   * Ensuring real-time updates from Gemini API.
   * Handling API rate limits and optimizing API calls for minimal latency.
   * Providing a seamless user experience with an interactive interface.



# Phase-3: Project Design

## Objective:

Develop the architecture and user flow of the application.



## Key Points:

1. **System Architecture:**
   * User inputs study-related queries via the UI.
   * Query is processed using the Google Gemini API.
   * AI model fetches and processes task and schedule data.
   * The frontend displays generated timetables and insights.
2. **User Flow:**
   * **Step 1:** User enters study tasks and preferences (e.g., "Create timetable for 3 subjects over 1 week").
   * **Step 2:** The backend calls the Gemini API to retrieve study schedule suggestions based on the provided tasks.
   * **Step 3:** The app processes the data and displays results in a clear and organized format.
3. **UI/UX Considerations:**
   * Minimalist, user-friendly interface for intuitive navigation.
   * Filters for adjusting study hours, tasks, and preferences.
   * Dark & light mode for an enhanced user experience.

.



# 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 | Member 1 | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day 1 | Member 2 | API response format finalized | Basic UI with input fields |
| Sprint 2 | Study Tasks Input & Schedule Generation | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Member 1& 2 | API response, UI elements ready | Task input and timetable generation |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Member 1&3 | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Member 2& 3 | 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 **search & comparison functionalities**. **(**🔴 **High Priority)** Debug API issues & handle **errors in queries**.

## 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 StudBud.

## Key Points:

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Develop **vehicle comparison and maintenance tips logic**.
   * Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.

**Fix:** Implement **caching** to store frequently queried results.

* + **Challenge:** Limited API calls per minute.

**Fix:** Optimize queries to fetch **only necessary data**.



# Phase-6: Functional & Performance Testing

## Objective:

Ensure that the StudBud App works as expected.

| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| --- | --- | --- | --- | --- | --- |
| TC-001 | Functional Testing | Input tasks for creating a study timetable | Timetable should display relevant study hours. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Input "Tips for studying effectively" | Appropriate study tips should be provided | ✅ Passed | Tester 2 |

| TC-003 | Performance Testing | Measure API response time | API should return results quickly. | ⚠ Needs Optimization | Tester 3 |
| --- | --- | --- | --- | --- | --- |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | ✅ Fixed | Develop er |
| TC-005 | Final Validation | Ensure UI is responsive. | UI should work on desktop. | ❌ Failed - UI broken on desktop | Tester 2 |
| TC-006 | Deployment Testing | Host the app using Streamlit Sharing | App should be accessible online. | 🚀 Deployed | DevOps |



# Final Submission

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