Hackathon Project Phases for the **StudBud: AI Study Planner** project.

Hackathon Project Phases

Project Title:

StudBud: AI Study Planner Using Gemini Flash

Team Name:

GenStudy

Team Members:

- K.Varun kumar
- G.Srikanth
- A.Arun Teja
- P. Akhil

Phase-1: Brainstorming & Ideation

Objective:

To develop an AI-powered personalized study planner using Gemini 2.0 Flash that creates adaptive study schedules based on students' goals, strengths, and learning preferences. The system optimizes time management and enhances learning efficiency through real-time updates and multi-modal study resources.

Key Points:

1. **Problem Statement:**

   "Studbud: AI Personalized Study Planner" is an AI-powered application that creates customized study plans based on students' specific goals, strengths, weaknesses, and learning preferences. By leveraging Gemini 2.0 Flash, Studbud dynamically adapts study schedules to optimize learning efficiency and maximize academic success.

2. **Proposed Solution:**

    - **AI-Powered Personalization:** Uses Gemini 2.0 Flash to process **student inputs** and **generate tailored study plans**.
    - **Time Management Optimization:** Ensures balanced study time allocation, **prioritizing weak areas** and high-impact topics.
3. **Target Users:**

    - **School & College Students:** Seeking structured study plans for improved academic performance.
    - **Lifelong Learners:** Individuals pursuing new skills or certifications through optimized study schedules.
    - **Tutors & Educators:** Helping students plan effective study routines with AI-driven insights.
4. **Expected Outcome:**

Personalized study schedules lead to better subject mastery.AI-driven recommendations help students focus on weak areas while reinforcing strengths.Structured planning minimizes last-minute cramming and enhances time management.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the StudBud App.

Key Points:

1. **Technical Requirements:**

    - Programming Language: **Python**
    - Backend: **Google Gemini Flash API**
    - Frontend: **Streamlit Web Framework**
    - Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**

    - Creates optimized and structured study plans based on subjects, priorities, and student learning patterns.
    - Modifies study schedules in real-time based on progress, difficulty level, and time constraints.
    - Supports login, registration, and guest mode, allowing users to access study plans with or without an account.

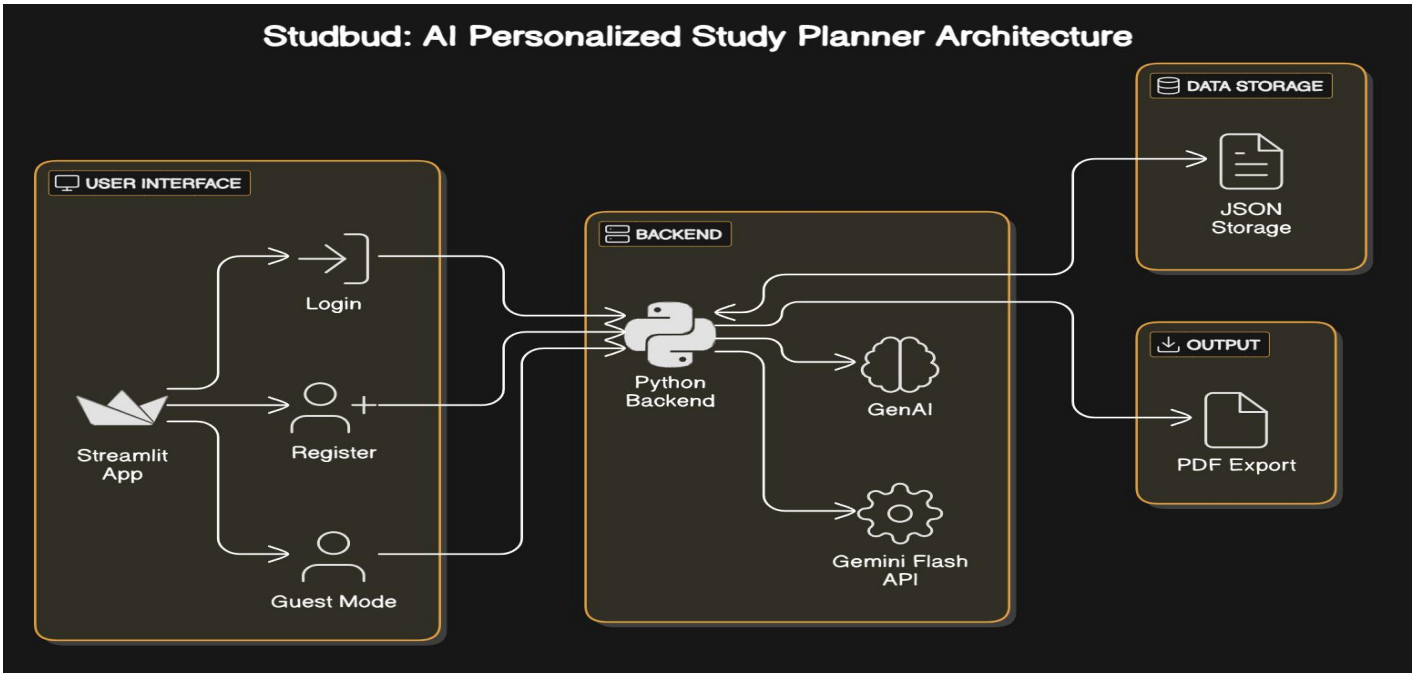
3. **Constraints & Challenges:**

    - Ensuring real-time updates from **Gemini API**.
    - Handling **API rate limits** and optimizing API calls.
    - Providing a **smooth UI experience** with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**

    - Query is processed using **Google Gemini API**.
    - AI model fetches and processes the data.
2. **User Flow:**

    2.1 Registration/Login Flow

    User visits the homepage

    Chooses between:

- Login (Existing users enter credentials)
- Register (New users provide details and create an account)
- Guest Mode (Proceed without login, data not saved)
- Upon login/registration, the user lands on the Dashboard

2.2 Study Plan Generation Flow

- User inputs their academic goals, strengths, weaknesses, and preferences
- The system sends data to Gemini 2.0 API for recommendation
- AI returns a structured study plan with:
- Daily study sessions
- Recommended learning methods
- Suggested breaks and productivity tips
- User reviews and modifies the schedule if needed
- Study plan is saved to JSON (if logged in)

3. **UI/UX Considerations:**

- **Minimalist, user-friendly interface** for seamless navigation.
- **Dark & light mode** for better 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	G.Srikanth	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	Medium	2 hours (Day 1)	End of Day 1	A.Arun Teja	API response format finalized	Basic UI with input fields
Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	K.Varun Kumar	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	Medium	1.5 hours (Day 2)	Mid-Day 2	P. Akhil	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	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 App.

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**.
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	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Tester 3
TC-002	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✅ Fixed	Developer

---

Final Submission

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