**Hackathon Project Phases Template** for the **STUDBUD** project.

---

# Hackathon Project Phases Template

## Project Title:

**STUDBUD AI LEARNING PLANNER**

## Team Name:

**Codebots**

## Team Members:

- Chakali Vennela
- Buggana Varshini
- Boreda Ashmitha

---

## Phase-1: Brainstorming & Ideation

### Objective:

Develop an AI-powered learning planner that helps students efficiently organize study schedules, set goals, track progress, and receive personalized learning recommendations.

### Key Points:

- **Problem Statement:**

  Students frequently struggle to balance academics, extracurricular activities, and personal life due to poor time management skills. Traditional planning tools do not provide intelligent suggestions based on workload distribution, deadlines, and cognitive fatigue. This study investigates how StudBud AI can leverage machine learning to create optimized study schedules, reducing academic stress and improving efficiency.

**Proposed Solution:**
The planner will analyze students' workload and prioritize tasks intelligently, ensuring optimal time distribution for assignments, revisions, and breaks. It will also integrate the Pomodoro technique and spaced repetition to enhance retention and avoid burnout.

**Target Users:**

- **Students –** High school, college, and university students who need help organizing study schedules, tracking deadlines, and improving productivity.

- **Lifelong Learners –** Individuals taking online courses, preparing for certifications, or self-studying new skills.

- **Expected Outcome:**
The expected outcome is a highly efficient study planner that helps students optimize their learning schedules, track progress, and receive personalized recommendations..

---

# Phase-2: Requirement Analysis

## Objective:

Develop an AI-powered learning planner that helps students efficiently organize study schedules, set goals, track progress, and receive personalized learning recommendations.

## Key Points:

1. **Technical Requirements:**

    - Programming Language: **Python**
    - Backend: **Flask**
    - Frontend: **html Web Framework**
    - Database: **Not required initially (API-based queries)**

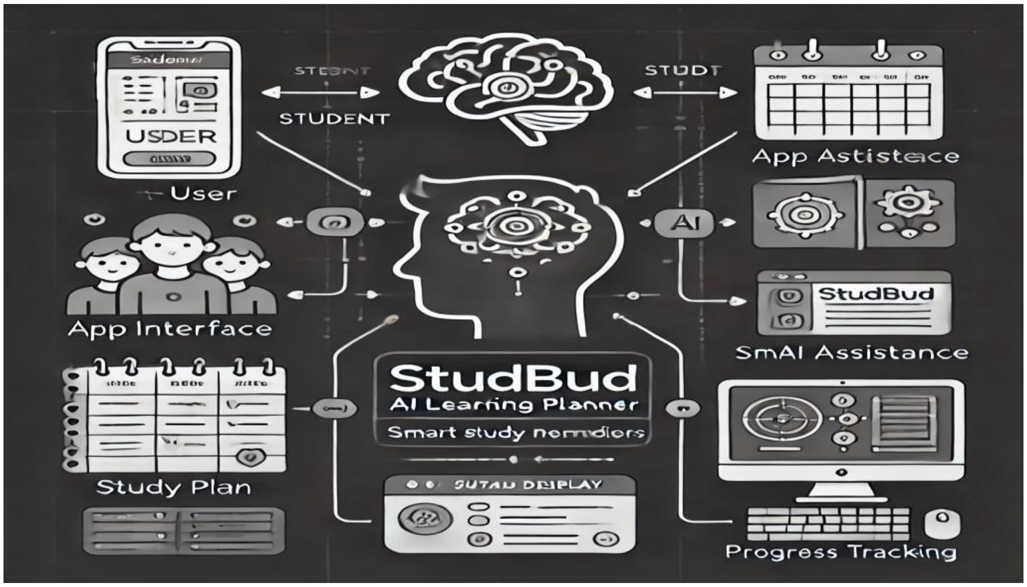
2. **Functional Requirements:**
    - Registration, authentication, and profiles.
    - Personalized schedules with dynamic adjustments.
    - Create, edit, and track study tasks.

3. **Constraints & Challenges:**
    - Ensuring user data protection and compliance with regulations.
    - Maintaining reliable AI recommendations and real-time adjustments.
    - Compatibility issues with third-party learning platforms.
    - Keeping users consistently motivated and reducing drop-off rates.

# Phase-3: Project Design

## Objective:

Develop the architecture and user flow of the application.



## Key Points:

1. **System Architecture:**

   StudBud AI Learning Planner follows a client-server architecture, where the frontend (mobile & web app) handles user interactions. The backend server manages scheduling, task tracking, and study plans.

2. **User Flow:**

   - Users sign up/login, set up their study preferences and goals, and create or import study tasks.
   - They can schedule study sessions, track progress, and get reminders through the planner.
   - The system provides performance insights and allows collaboration with peers or mentors

3. **UI/UX Considerations:**

   - Balancing an intuitive interface with advanced study planning features.
   - Cross-Platform Compatibility which ensuring a seamless experience across web and mobile devices.
   - User Engagement helps designing interactive elements to maintain motivation without overwhelming users.

---

# 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 | varshini | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | vennela | API response format finalized | Basic UI with input fields |
| Sprint 2 | research & Comparison | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | ashmitha | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | varshini | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | vennela | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation & Deployment | 🟢 Low | 1 hour (Day 2) | End of Day 2 | ashmitha | 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:** HTML,JavaScript
    - **Backend:** Flask
    - **Programming Language:** Python
2. **Development Process:**
    - The development follows an agile methodology, starting with requirement analysis
    - UI/UX design, followed by frontend and backend development.
    - Iterative testing and feedback loops ensure functionality, with final deployment and continuous updates for improvements**.**
3. **Challenges & Fixes:**
    - User Retention – Fixed by adding gamification (streaks, rewards) and intuitive UI.
    - Task Overload – Solved with smart scheduling and prioritization features.
    - Platform Integration Issues – Addressed by using standardized APIs for seamless connectivity.

# Phase-6: Functional & Performance Testing

## Objective:

| Test Case ID | Category | Test Scenario | Expected Outcome | Status | Tester |
|---|---|---|---|---|---|
| **TC-001** | Functional Testing | Add a subject and study hours | Subject and hours should be added successfully | ✅ Passed | varshini |
| **TC-002** | Functional Testing | Generate timetable with multiple subjects | Timetable should display all added subjects | ✅ Passed | ashmitha |
| **TC-003** | Performance Testing | API response time under 500ms | API should return results quickly | ⚠ Needs Optimization | vennela |
| **TC-004** | Bug Fixes & Improvements | Fixed incorrect data display issue | Data should be correctly displayed in the table | ✅ Fixed | varshini |
| **TC-005** | Final Validation | Ensure UI is responsive across devices | UI should work on mobile & desktop | ❌ Failed - UI broken on mobile | vennela |

# Final Submission

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