**Hackathon Project Phases Template** that ensures students can complete it efficiently while covering all six phases. The template is structured to capture essential information without being time-consuming.

Hackathon Project Phases Template

# Project Title:

**CoutureAl: AI study planner**

# Team Name:

Elite Techies

# Team Members:

* Karthik Reddy Muppa
* AKHIL POGAKU
* Ghambeer T
* Brungi Pranav

# Phase-1: Brainstorming & Ideation

## Objective:

The AI Study Planner aims to help students, professionals, and lifelong learners optimize their study schedules using AI. It provides personalized plans based on learning goals, available time, and progress. The goal is to improve time management, retention, and productivity.

## Key Points:

1. **Problem Statement:** Traditional static study tools fail to adapt to changes in a learner's progress or schedule. This leads to missed deadlines, reduced motivation, and ineffective learning.
2. **Proposed Solution:**

The AI Study Planner uses Google Gemini AI to break down goals into daily plans. It lets users add or modify tasks manually, automatically groups them by week/month/year, and marks parent tasks as complete once all sub-tasks are done. It also allows downloading the study plan as a PDF.

1. **Target Users:**

Students (K-12 to college), professionals, exam takers, and lifelong learners. It also aids neurodivergent learners (e.g., ADHD) by providing structured, dynamic study plans.

1. **Expected Outcome:**

Improved productivity, higher focus, dynamic scheduling, and enhanced user motivation through progress tracking and adaptive planning.

# Phase-2: Requirement Analysis

## Objective:

To analyze the core features and technical dependencies needed to deliver a usable and intelligent study planner.

## Key Points:

1. **Technical Requirements:**

* **Programming Language:**Python
* **Frontend:** Streamlit
* **Backend:**Flask
* **Database:** PostgreSQL via SQLAlchemy
* **AI Integration:** Google Gemini API

**Functional Requirements:**

- AI-based breakdown of study goals into day-wise tasks  
 - CRUD operations for tasks  
 - Auto-completion of parent tasks  
 - Manual task addition  
 - PDF schedule export

**Constraints & Challenges:** (Any limitations or risks):

- AI output unpredictability  
 - Ensuring auto-grouping logic works for all cases  
 - Local backend server dependency

# Phase-3: Project Design

## Objective:

Design the interaction between components and plan the UI/UX.

## Key Points:

1. **System Architecture Diagram:**

- User inputs goal via Streamlit UI  
 - Frontend calls Gemini API to generate plan  
 - Tasks are sent to Flask backend for storage  
 - PostgreSQL stores all task data  
 - UI fetches and renders tasks with styling and grouping

1. **User Flow:**

1. Enter goal  
 2. Generate plan via Gemini  
 3. Tasks are displayed and grouped  
 4. Users can complete/delete tasks or download the plan

1. **UI/UX Considerations:**

- Intuitive design with custom CSS  
 - Task grouping with indentation  
 - Button icons for task completion and deletion

# Phase-4: Project Planning (Agile Methodologies)

## Objective:

Plan sprints and task distribution.

## Key Points:

1. **Sprint Planning:**

- Sprint 1: Setup backend, database, and basic UI structure  
 - Sprint 2: Integrate Gemini and dynamic task rendering  
 - Sprint 3: Implement auto-grouping, parent completion, and PDF export  
 - Sprint 4: Final testing, bug fixes, and UI polishing

1. **Task Allocation:**

- Frontend: UI and UX using Streamlit, PDF generation  
 - Backend: Flask API, PostgreSQL setup  
 - AI Integration: Prompt design, Gemini response handling  
 - Team collaboration via GitHub

# Phase-5: Project Development

## Objective:

Build the full-stack system with working AI integration.

## Key Points:

1. **Technology Stack Used:**

 - Frontend: Streamlit  
 - Backend: Flask  
 - AI: Google Gemini API  
 - Database: PostgreSQL  
 - Libraries: dotenv, requests, reportlab, SQLAlchemy, flask-cors

1. **Development Process:**

- Set up Flask with task CRUD API  
 - Design Streamlit UI with buttons and styles  
 - Add Gemini integration for goal processing  
 - Implement parent-child grouping logic  
 - Test and fix logic for auto-complete and indentation

3.**Challenges & Fixes:**

- Gemini sometimes gives unexpected output → handled via line-by-line filtering  
 - Streamlit reruns → minimized using stateful reruns  
 - Parent completion logic → improved with group mapping

# Phase-6: Functional & Performance Testing

## Objective:

Verify functionality and ensure smooth performance.

## Key Points:

**Test Cases Executed:**

- Task addition, deletion, and update  
 - Plan generation and rendering  
 - Auto-completion of parent tasks  
 - Manual task handling  
 - PDF export

**Bug Fixes & Improvements:**

- Improved error handling for API failures  
 - Enhanced UI styling and hover effects  
 - Optimized rerun behavior

1. **Final Validation:**

The system meets initial goals: study plan generation, task management, dynamic grouping, and export.

1. **Deployment**: Local deployment on `localhost`; can be hosted via Streamlit Cloud or Render for production.

# Final Submission

1. **Project Report**
2. **GitHub/Code Repository Link**