# Project Title:“AI study planner”

**Team Name: 404 coders**

# Team Members:

* Vemula Nishanth Madhav
* Mididoddi Rohit Kumar
* Vemula Vamshi Krishna
* Davi Reddy Mahith Reddy



**Phase-1: Brainstorming & Ideation**

* **1. Problem Statement:**
  + Students struggle to create personalized, effective study schedules that adapt to their learning styles and the vast amount of information they need to process. Traditional study planners are often rigid and don't account for individual needs or dynamically adjust based on performance.
* **2. Proposed Solution:**
  + Develop an AI-powered study planner web application. This application will:
    - Allow users to input their courses, topics, and learning preferences.
    - Utilize Generative AI (Gen-AI) to create personalized study schedules that incorporate spaced repetition, active recall, and other effective learning techniques.
    - Provide progress tracking and adjust schedules based on user performance.
    - Utilize Gen-AI to create practice questions and summaries of study materials.
* **3. Target Users:**
  + Students of all levels (high school, college, professional certifications).
  + Individuals engaged in self-directed learning.
* **4. Expected Outcome:**
  + Improved study efficiency and knowledge retention for users.
  + Reduced stress and anxiety associated with studying.
  + A personalized, adaptable study tool that promotes effective learning habits.

**Phase-2: Requirement Analysis**

* **1. Technical Requirements:**
  + Programming Language: Python
  + Web Framework: Flask
  + Gen-AI Libraries: TensorFlow, PyTorch, Hugging Face Transformers (for NLP tasks like question generation and summarization).
  + Frontend: HTML, CSS, JavaScript.
  + API's: Gemini API.
* **2. Functional Requirements:**
  + Course/Topic Input: Allow users to add courses and specific topics.
  + Learning Style Preferences: Gather information on user's preferred learning methods.
  + Study Schedule Generation: Generate personalized study schedules using Gen-AI.
  + Schedule Adjustment: Dynamically adjust schedules based on progress.
  + Gen-AI Generated practice questions.
  + Gen-AI generated summaries.
  + Calendar Integration: Allow users to sync study schedules with their calendars.
* **3. Constraints & Challenges:**
  + Accuracy of Gen-AI: Ensuring the generated study schedules and content are relevant and effective.
  + Data Privacy: Securely storing and handling user data.
  + Scalability: Handling a large number of users and data.
  + Cost of using Gen-AI API's.
  + Complexity of the Gen-AI implementation.

**Phase-3: Project Design**

* **1. System Architecture Diagram:**
  + User (Web Browser) -> Flask Web Application (Python) -> Gen-AI Model (Gemini API).
  + The Flask application will handle user requests, interact with the Gen-AI model, and manage data in the database.
* **2. User Flow:**
  + User adds courses and topics.
  + User inputs learning preferences.
  + User requests a study schedule.
  + Flask application sends data to the Gen-AI model.
  + Gen-AI model generates a schedule.
  + Flask application stores and displays the schedule.
  + User studies, and updates progress.
  + The system adjusts the schedule based on user progress.
  + User requests practice questions or summaries from the Gen-AI.
* **3. UI/UX Considerations:**
  + Clean and intuitive interface.
  + Responsive design for mobile and desktop.
  + Clear visualization of study schedules and progress.
  + Simple and easy to use inputs for course information.

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

* **1. Sprint Planning:**
  + Sprint 1: Course/topic input.
  + Sprint 2: Basic schedule generation (without Gen-AI).
  + Sprint 3: Integration of Gen-AI for schedule generation.
  + Sprint 4: Schedule adjustment.
  + Sprint 5: Gen-AI question and summary generation.
  + Sprint 6: UI/UX enhancements and calendar integration.
* **2. Task Allocation:**
  + Divide tasks among team members based on their skills (backend, frontend, AI).
  + Use a task management tool (e.g., Trello, Jira).
* **3. Timeline & Milestones:**
  + Set short sprints (e.g., 2 weeks).
  + Define clear milestones for each sprint.
  + Regular stand-up meetings to track progress.

**Phase-5: Project Development**

* **1. Technology Stack Used:**
  + Python, Flask, HTML, CSS, JavaScript, API's from Gemini API.
* **2. Development Process:**
  + Set up the Flask environment.
  + Implement course/topic input.
  + Build the core schedule generation logic.
  + Integrate the Gen-AI model.
  + Develop schedule adjustment.
  + Implement Gen-AI question and summary functions.
  + Design the UI/UX.
  + Implement calendar integration.
* **3. Challenges & Fixes:**
  + Gen-AI model performance: Fine-tune the model with relevant data.
  + Frontend responsiveness: Test on different devices and browsers.
  + API rate limits: implement caching, and monitor API usage.

**Phase-6: Functional & Performance Testing**

* **Objective:** Validate functionality and performance within session-based limitations.
* **Key Points:**
  + **1. Test Cases Executed:**
    - In-Memory Data: Course/topic input, editing, deletion (session-only).
    - Schedule Generation: Varied course loads, preferences (session-only).
    - Schedule Adjustments: Dynamic changes within the current session.
    - Gen-AI: Question and summary accuracy.
    - Performance: Response times, memory usage.
    - Security: Session security, API vulnerabilities.
  + **2. Bug Fixes & Improvements:**
    - Schedule calculation fixes.
    - Gen-AI accuracy improvements (prompt refinement).
    - UI responsiveness fixes.
    - Error handling improvements.
  + **3. Final Validation:**
    - Core functionalities (session-based) are working.
    - Performance is acceptable within session limits.
    - Usability is confirmed.
    - Session security is verified.
    - **Limitations:** Data is lost after the session ends.
  + **4. Deployment:**
    - Flask app deployed on a cloud platform (e.g., Heroku).
    - Demo link: (Insert link).
    - **Deployment Notes:**
      * Session management is critical.
      * Memory usage must be monitored.
      * Best for single-session use.



