## **Project Title:**

**StudBud : AI study planner**

## **Team Name:**

Team ByteBots

## **Team Members:**

* M. Srinivas
* V. Srivathsa
* M. Vamcy
* A. Sai Vardhan

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

Develop an AI-powered study planner using BERT Architecture (from Transformers), this tool helps students optimize their study schedules to achieve their academic targets efficiently.

### **Key Points:**

1. **Problem Statement:**
   * Students wishing to improve in specific subjects to create a balanced study schedule incorporating various learning methods suited to their preferences.
2. **Proposed Solution:**
   * An AI-powered application using **Gen AI and BERT Architecture** to provide **detailed study plans, Optimize study time and track their projects.**
   * The app offers various learning methods and **user-friendly study plans** based on user preferences.
3. **Target Users:**
   * **Highschool students, University/College students, Competitive exam Aspirants** who need a structured study plan to cover vast syllabi efficiently.
4. **Expected Outcome:**
   * A functional **AI-powered study planner/Information app** to create a powerful and user-friendly application that helps students optimize their study schedules, improve their academic performance, and achieve their learning goals efficiently.

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for the StudBud App.

### **Key Points:**

1. **Technical Requirements:**

* **Programming Languages:**

**Python**: For backend development, AI/ML model integration, and data processing.

**JavaScript/HTML/CSS**: For frontend development (if building a web app).

* **Frameworks and Libraries:**

**TensorFlow/Keras**: For building and training the AI/ML models.

**Hugging Face Transformers**: For integrating BERT or other pre-trained language models.

**Pandas/Numpy**: For data manipulation and pre-processing.

**Scikit-learn**: For additional machine learning tasks (e.g., clustering, classification).

**FastAPI/Flask**: For building the backend API.

* **AI/ML Tools:**

**BERT (or other transformer models)**: For natural language processing (NLP) tasks like understanding student inputs.

**Tokenizers**: For pre-processing text data.

* **Other Tools:**

**Google Colab**: For initial development and backend development.

* **Backend Hosting**:

**Cloud Platforms**: GCP, AWS, or Azure for hosting the backend API.

**Serverless Options**: AWS Lambda or Google Cloud Functions for scalable backend services.

* **Frontend Hosting**:

**Vercel**: For hosting web applications.

1. **Functional Requirements:**

Ability to **fetch user details (Academic goals, strengths, weaknesses, preferred study methods)**.

Display **study session reminders, deadline alerts, progression** in an intuitive UI.

Provide **improvement in weak areas** based on results.

Allow users to **track their pro****gress toward academic goals** based on target grades and performance.

Give **detailed structured study plans based on student preference by BERT Architecture.**

1. **UI Requirements:**

* **Platforms:**

**Web Application**: Accessible via browsers on desktops and mobile devices.

* **Design:**

**Responsive Design**: The UI should adapt to different screen sizes (desktop, tablet, mobile).

**User-Friendly Interface**: Easy navigation, clear instructions, and minimal clutter.

**Dashboard**: For students to view their study plans, progress, and recommendations.

1. **Constraints & Challenges:**

* **Handling Large User Base:**

**The application must scale to support thousands or millions of users.**

* **Resource Intensive AI/ML Models:**

**BERT and other transformer models are computationally expensive and may require significant resources.**

* **Real-Time Recommendations:**

**The application must generate study plans and recommendations in real-time.**

* **Latency:**

**High latency in generating study plans can lead to a poor user experience.**

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the application.

### **Key Points:**

1. **System Architecture:**

**Frontend:** Streamlit

**Backend:** Built using FlaskAI

1. **User Flow:**
   * Step 1: User enters a query (e.g., "Best motorcycles under ₹1 lakh").
   * Step 2: The backend **calls the Gemini Flash API** to retrieve vehicle data.
   * Step 3: The app processes the data and **displays results** in an easy-to-read format.
2. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * **Filters for study plans, user preferences, etc.**
   * **Simple UI** for better user experience.

## 

## **Phase-4: Project Development and Testing**

### **Objective:**

Implement core features of the StudBud App.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Bert Architecture (From Transformers), Tensorflow, Hugging faces API, Numpy, Faker, pandas Libraries, etc.
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **Hugging faces API integration and generate study plans**.
   * Develop **student dataset using Faker, Numpy and pandas library in Python**.
   * Training the model **using scikit-learn with the dataset.**
   * **Evaluating the model on the dataset.**
   * **Build a Backend API to serve the model.**
   * **Build a frontend and host the application on Streamlit.**
3. **Testing Process:**

* Integrate the Backend and Frontend to Streamlit and start the application.
* Test the application with various prompts to generate the study plans.
* Test the latency, performance and runtime with multiple requests and make sure the Hugging Faces API and BERT architecture is running correctly to give relevant information.