# Project Proposal: Model Development of an Artificial Intelligence Personalized Learning Assistant (AI Tutor)

## 1. Introduction

The rapid advancement of artificial intelligence (AI) has unlocked immense potential for personalized education. This project aims to develop an Artificial Intelligence **Personalized Learning Assistant** (**AI Tutor**) running the small language models in edge devises (local laptop or CPU), prompt engineering techniques and Retrieval Augmentation Generation (RAG) and Agentic AI to enhance model performance at inference.

The goal is to create an AI tutor running efficiently with low CPU resources or offline on local laptop that helps students learn specific subjects, such as Deep Learning, by utilizing conversational AI and innovative tools. The assistant will guide students by answering questions, generating mind maps, quizzes, and study tips grounded on uploaded learning materials, such as PDF and PPTX files. The uploaded materials with information of course syllabus and learning objects and key study notes enrich the users prompts and facilitate the RAG grounding of AI responses. Users will focus on chatting with the AI and learning the topics without the need to crave users prompt to ensure good responses from the LLM.

### Problem Statement

Traditional learning resources often lack the interactivity and personalization required to cater to diverse student needs. With limited access to personalized tutoring, students struggle to grasp complex concepts at their own pace. A digital learning assistant capable of adapting to individual learning objectives can bridge this gap.

The AI Tutor is running on small LLM that do not require intensive computing power and able to run on local PC/Laptop CPU. It is capable to run offline, thereby ensuring privacy and security of the data loaded to the LLM via RAG and users chatting with the AI Tutor.

## 2. Scope of the Project

To design and implement a conversational AI-powered learning assistant that leverages RAG and prompt engineering to:

The AI Tutor will include the following simplified features:

1. **Contextual Question Answering**:
   * Enable users to ask questions related to uploaded content.
   * Provide precise and context-aware responses using a pre-trained language model.
2. **Simplified Study Guidance**:
   * Generate basic study suggestions based on user-defined objectives.
3. **Quiz Generation with Feedback**:
   * Create basic multiple-choice quizzes from uploaded content.
4. **Multi-document types Integration**:
   * Parse PDF and PPTX files containing text and images to extract key content.

### **3. Methodology**

The project will follow a simplified phased development approach:

1. **Requirement Analysis and data preparation**:
   * Gather functional requirements.
   * Define basic use cases and personas (e.g., students).
2. **Additional Feature Integration**:
   * Add basic quiz generation and mind map modules.
   * Implement document parsing for multimedia integration.

## 4. Technology Stack

| **Component** | **Technology** | **Role** |
| --- | --- | --- |
| **Frontend** | Gradio | Interactive user interface for Q&A, mind maps, quizzes, and tips. |
| **LLM** | LLaMA (Meta’s model) or Phi models, LangChain Framework etc. | Core conversational agent for RAG, question answering, and tutoring. |
| **Vector Database** | Milvus, Weaviate, or FAISS etc | Open-source options for embedding storage and fast similarity search. |
| **Document Parsing** | PyMuPDF, pdfplumber, python-pptx etc | Parse PDFs and PPTX files for content extraction. |

## 5. Workflow

#### **Frontend with Gradio**

* Build an intuitive Gradio interface with tabs for:
  1. **Content Upload**: Allow users to upload PDF/PPTX files.
  2. **Interactive Q&A**: Users can ask questions, and the AI provides contextual answers.
  3. **Mind Maps**: Display auto-generated visualizations.
  4. **Quizzes**: Provide auto-generated quizzes and instant feedback.

#### **Backend with LLM and LangChain Framework**

* Use LLM as the conversational agent
* Integrate with a RAG (Retrieval-Augmented Generation) pipeline to enable contextual Q&A using the uploaded content.

#### **Document Processing (LangChain’s Document loaders)**

* Extract text and metadata using:
  + PyMuPDF or pdfplumber for PDFs.
  + docling parses PDF, DOCX, PPTX, HTML, and other formats into a rich unified representation including document layout, tables etc., making them ready for generative AI workflows like RAG.
* Summarize or structure extracted data to generate embeddings.

#### **Vector Search with Open-Source Database**

* Store embeddings in an open-source vector database:
  1. **Milvus**: Scalable and customizable.
  2. **FAISS**: Lightweight and fast for local development.
* Use embeddings for semantic similarity search to retrieve relevant information during Q&A.

#### **Quiz and Study Plan Generation**

* Use the LLM to:
  1. Generate multiple-choice questions, fill-in-the-blanks, or open-ended questions.
  2. Provide feedback on answers and suggest additional resources or study tips.
  3. Create personalized study plans based on learning objectives and progress.

## References