**Sutram Solutions Pvt Ltd** is looking to hire **3 dedicated interns**, each focusing on one of the following key technical areas critical to building an AI-powered Learning Management System:

| **Applied AI/ML & Natural Language Processing (NLP)** | **Data Engineering & Integration** | **Backend Development & LLM App Engineering** |
|---|---|---|
| If you are strong in model building, NLP tasks, or LLMs like GPT or BERT, this track is for you. The **Focus Areas are:**<br>• Tokenization, summarization, classification<br>• Transformers, HuggingFace, OpenAI APIs , LLM chaining<br>• Prompt engineering and fine-tuning models | If you are good with working on data pipelines, cleaning, and managing educational content in various formats, this is your strength area The **Focus Areas are**:<br>• Pandas, matplotlib,NumPy, file format handling (CSV, PDF, JSON)<br>• Vector DBs (FAISS, ChromaDB), embeddings<br>• Data pipelines for model feeding | If your expertise lies in building scalable Python APIs and integrating LLMs into usable backend systems, choose this track. The **Focus Areas are**:<br>• FastAPI, Flask, LangChain, LlamaIndex<br>• Deploying LLMs via REST APIs<br>• Docker, microservices, async & session handling |

## What You Should Do:

- **Review your strengths** across the above areas & **Pick one focus area** where you want to intern
- **Tailor your preparation and responses** accordingly, as screening tasks will be aligned to your chosen domain

✅ Only candidates who are **focused and aligned** with the required domain expertise will be considered for the next round.

➡ **If your submission is approved**, you will move to the **final round**, which will be a **virtual discussion** with our technical panel.

✅ This virtual discussion will be the **final step** for your selection.

We have provided **4 assignments** for you to choose from. **1 assignment is mandatory**, and for the remaining **3 assignments**, you can pick any 1 that interests you the most.

Mandatory Assignment (Must Submit First):

- **Assignment 1**: Build a Simple RNN in Keras (This is required to be completed and submitted first by all candidates as it will demonstrate your understanding of neural network building.)

Optional Assignments (Choose any 1, Submit After Mandatory Assignment):

- **Assignment 1**: Personalized Learning Path Recommendations
- **Assignment 2**: LLM-Powered Automated Quiz Generator
- **Assignment 4**: Instructor Support Tool for Content Enhancement and Feedback Generation

## Important Notes:

1. **First**, you must complete and submit **the mandatory assignment** (Assignment 1 - RNN task).
2. Once the mandatory assignment is submitted, you may start working on **any one of the remaining three assignments** that best aligns with your skills and interests
3. For Any doubts call us @ **6302519834 / 9556151572 or Message us**https://www.linkedin.com/company/sutram-solutions-pvtltd

# ✅ Assignment Submission Guidelines & Instructions

Please follow the steps below carefully for your assignment submission:

### 🗂 *Follow the Folder Structure*

- Use the **folder structure** shared with you.
- Ensure your code and files are properly organized according to this structure.

### 🌿 *Create a GitHub Branch*

- Push your code to GitHub in a **new branch** with the following naming convention: fName_lName_assignmentName
  - For example: ritu_raj_nlptasks

```
your_submission_name.zip (or .tar.gz)
├── README.md (or README.txt)
├── datasets/
│   ├── classification_data.txt (or .csv, .json - clearly labeled)
│   └── generation_data.txt
├── src/
│   ├── classification_model.py
│   └── generation_model.py
└── utils/
    └── preprocessing.py (optional - for shared preprocessing functions)
```

### 📝 *Add a README File*

Your project should include a README.md file at the root level containing the following:

- **Instructions** on how to run the program.
- **Sample Input and Output** to demonstrate how your program works.
- (Optional) **Screenshots** of the working application, if applicable.
- **Note**: The application must run after checkout or if it a Jupyter Notebook it must run on google colab.

### 🚀 *Final Submission*

- Once completed, make sure the code is committed and pushed to the correct branch.
- Share the link to the Github branch in the indeed message or through email career@sutramsolutions.com.

## Dataset Requirements

- You must **create a synthetic dataset** both for testing and training using either curated educational sentences or scraping public domain text (no copyrighted data).
- For generation, include rich, grammatically correct sentence structures to train effectively.

Submitting your architecture design as a PPT or PDF for clearer visualization and presentation would be highly appreciated but not mandatory**.**

# Mandatory Assignment: Design and Implement a Simple (RNN)

Objective: This assignment aims to evaluate your understanding and practical implementation skills related to Recurrent Neural Networks (RNNs) for Natural Language Processing (NLP) tasks using the Keras library (TensorFlow backend preferred).

**Tasks:** You are required to design and implement a single, simple RNN model that can perform **two distinct NLP tasks**:

1. **Educational Text Classification:** Classify short educational text snippets into one of the following predefined categories: **Math, Science, History**.
2. **Next Word Generation (Auto-completion/Prediction):** Given the beginning of a sentence (at least 10 words), generate the next 20 words to simulate intelligent auto-completion or content prediction.

**Evaluation/Demonstration:**

- **Classification:** Demonstrate the model's ability to classify a few unseen text snippets from your test set (you can split your classification dataset into training and testing sets). Report the classification accuracy on your test set.
- **Next Word Generation:** Provide a starting sentence (that exists within your training corpus or is a reasonable starting point) and show the model's generated next 20 words. Briefly comment on the coherence of the generated text

**Evaluation Criteria:** Your submission will be evaluated based on the following:

- ✓ **Dataset Creation Quality:** Is the created dataset relevant and reasonably structured for the tasks?
- ✓ **Correctness of Implementation:** Does the code run without errors and perform the requested tasks?
- ✓ **Understanding of RNNs:** Does the model architecture demonstrate a basic understanding of RNN principles?
- ✓ **Model Implementation**: Correctness of the RNN implementation for both classification and text generation.
- ✓ **Text Preprocessing Skills:** Are the necessary text preprocessing steps implemented correctly?
- ✓ **Clarity of Code and Documentation:** Is the code well-commented and is the README file informative?
- ✓ **Problem-Solving Approach:** How effectively did you approach the challenges of this assignment?
- ✓ **Basic Evaluation/Demonstration:** Did you demonstrate the model's performance for both tasks?

## Plan of Attack / Implementation Plan

**Phase 1: Dataset Creation (Allocate significant initial time)**

- Text Classification: You are required to create your own dataset for both tasks. The labels are (Math, Science, History & English).
- Next Word Generation Dataset: Create a single, longer text corpus of educational content related to either Math, Science, or History (you can choose one). This corpus will be used to train your model to predict the next words in a sequence.

### Phase 2: Text Preprocessing (Crucial for model input)

4. **Classification Preprocessing:**
   - Read your classification dataset.
   - Tokenize the text snippets (break into words) & create a vocabulary.
   - Convert text to sequences , pad sequences & Prepare labels for training (e.g., one-hot encode).
   - Split your data into training and testing sets.
5. **Generation Preprocessing:**
   - Read your generation text corpus , Tokenize the text & Create a vocabulary.
   - Create sequences of words as input and the next word as the target.
   - Convert these sequences into numerical representations using the vocabulary.
   - Prepare the target next words (e.g., one-hot encode).

## Phase 3: Model Building (Keras RNN)

6. **Choose RNN Architecture:** Decide on a simple RNN type to start with (SimpleRNN). You can explore LSTM or GRU later if time permits.
7. **Build Classification Model:**
    - Create a sequential Keras model.
    - Add an Embedding layer (to convert word indices to vectors) if required add more RNN layers.
    - Add a Dense output layer activation, loss function optimizer (e.g., `adam`).
8. **Build Generation Model:**
    - Create a sequential Keras model.
    - Same as above

## Phase 4: Model Training and Evaluation

9. **Train Classification Model:** Train the model using your training data for a reasonable number of epochs.
10. **Evaluate Classification Model:** Evaluate the trained model on your test set and report the accuracy.
11. **Train Generation Model:** Train the model on your prepared sequences for a reasonable number of epochs.
12. **Demonstrate Next Word Generation:** Provide a starting sentence and use your trained model to predict words.

## Phase 5: Documentation and Submission

13. **Write README:** Create a README file explaining your approach, how to run your code, your model architectures, preprocessing steps, and your observations (accuracy for classification, coherence for generation).
14. **Submit:** Package your code, datasets, and README into GITHUB.

## Key Takeaways for Candidates:

- **Start Simple:** Don't aim for overly complex models initially. Focus on getting a basic RNN working for both tasks.
- **Preprocessing is Critical:** Pay close attention to the text preprocessing steps as they directly impact model performance.
- **Document Your Process:** Clearly explain your choices and steps in the README.
- **Demonstrate Functionality:** Ensure your code runs and produces the required outputs (classification accuracy and generated text).

Important Notes:

- You are encouraged to research and implement different variations of RNNs (like LSTM/GRU) and experiment with different architectures.
- Keep the code clean and follow best practices for writing maintainable Python code.

# Assignment 1: Personalized Learning Recommendations

**Objective**: Build a scalable and modular backend system for an LMS that delivers personalized learning recommendations using AI techniques, that takes student questionnaire input and generates personalized AI-driven learning paths and content recommendations.

| **Input Section:** | **Output Section:** |
|---|---|
| 1. The system will collect student responses to a set of predefined questions through a questionnaire format.<br>2. The questionnaire will include questions about learning preferences, subject strengths, emotional states, attention span, and more.<br>3. Each answer will correspond to a set of predefined options from which students can select the one that best describes them.<br>4. The input will include both cognitive (e.g., confidence level, subject preferences) and behavioral traits (e.g., participation, homework submission).<br>5. The data from the questionnaire will be used to build a complete student profile, capturing key characteristics.<br>6. The student's answers should be categorized into relevant parameters that reflect their academic behavior and preferences.<br>7. The input data will be provided in a structured format (e.g., JSON) to ensure ease of processing.<br>8. It will include critical inputs such as preferred learning styles (visual, auditory, etc.) and emotional states (calm, stressed, motivated).<br>9. The system should be capable of processing various types of input responses and convert them into actionable learning data.<br>10. This student input will be the foundation for generating a personalized learning experience and content recommendations. | 1. Based on the student's profile, the system will generate a personalized learning path recommendation.<br>2. The output will include a suggested schedule with content recommendations such as videos, readings, or interactive quizzes.<br>3. A learning path will be constructed to align with the student's strengths and weaknesses as reflected in their profile.<br>4. Each recommended content will be categorized by type (e.g., lessons, practice problems) and difficulty level.<br>5. The model will also suggest pacing strategies week wise, considering the student's attention span and motivation trends.<br>6. The system will provide an explanation for why specific topics or content formats are recommended to enhance student understanding.<br>7. The output will be returned in a structured format such as JSON or API response, including learning recommendations and content metadata.<br>8. The system may optionally generate additional study materials, like notes or quizzes, to reinforce learning.<br>9. All output recommendations must be accessible via a REST API endpoint for integration into the backend of an AI-powered LMS. |


student_recommendation_output.json


student_profile_input.json

# Student Profile Questionnaire,

**Subject:** (Please select one)

1. Math
2. Science

| | |
|---|---|
| **Academic Strengths and Weaknesses:** (Choose the option that best describes you)<br><br>1. I understand the main ideas well, but sometimes struggle with the step-by-step procedures.<br>2. I'm good at solving problems, but I find it hard to remember formulas and specific facts.<br>3. I feel pretty comfortable with most aspects of this subject.<br>4. I find the basic concepts challenging and need extra help to keep up. | **Preferred Learning Style:** (Choose the option that best describes how you learn)<br><br>1. I learn best when I see pictures, diagrams, and videos.<br>2. I learn best by listening to lectures and discussions.<br>3. I learn best by doing things myself, like experiments or hands-on activities.<br>4. I learn best by reading textbooks and writing notes. |
| **Attention Span:** (Choose the option that best describes your focus in class)<br><br>1. I can usually focus for a long time without getting distracted.<br>2. I can focus for a while, but I need short breaks to stay engaged.<br>3. I get distracted easily and find it hard to pay attention for long periods.<br>**4.** It's very hard for me to focus, even for short activities. | **Emotional State:** (Choose the option that best describes how you generally feel about learning this subject)<br><br>1. I usually feel positive and interested in learning this subject.<br>2. Sometimes I feel a bit worried or frustrated when learning this.<br>3. I often feel unmotivated or not interested in this subject.<br>4. I get quite stressed or upset when learning this subject and need a lot of support. |
| **Classroom Participation:** (Choose the option that best describes your participation in class)<br><br>1. I often raise my hand, ask questions, and share my ideas.<br>2. I participate when the teacher asks me directly.<br>3. I usually just listen and don't talk much in class.<br>4. I feel uncomfortable speaking in class. | **Homework Submission Timeliness:** (Choose the option that best describes when you usually hand in homework)<br><br>1. I always submit my homework on time.<br>2. I usually submit my homework on time, but sometimes it's a little late.<br>3. My homework is often late.<br>**4.** I rarely hand in my homework. |
| **Quiz Scores:** (Choose the option that best describes your quiz performance)<br><br>1. I usually get very high scores on quizzes.<br>2. I usually get good scores on quizzes.<br>3. My quiz scores are usually just average.<br>4. I often get low scores on quizzes. | **Confidence Level:** (Choose the option that best describes how confident you feel in this subject)<br><br>1. I feel very confident in my ability to do well in this subject.<br>2. I feel somewhat confident, but I sometimes doubt myself.<br>3. I don't feel very confident in this subject.<br>**4.** I have very little confidence in my ability to learn this subject. |

**Motivation Trends:** (Choose the option that best describes what motivates you to learn this subject)

1. I enjoy learning this subject because it's interesting to me.
2. I'm motivated by getting good grades and praise.
3. My interest in this subject goes up and down.
4. I don't feel very motivated to learn this subject.

**Subject Preferences:** (Choose the option that best describes your preference between Math and Science)

1. I like Math much more than Science.
2. I like Science much more than Math.
3. I enjoy both Math and Science equally.
4. I don't really enjoy either Math or Science.

**Interest Tags (hobbies, favorite subjects):** (Select up to two options that interest you the most)

- **Math Interests:**
  - Solving puzzles and logic problems
  - Seeing how math is used in real life
  - Thinking about abstract math ideas
  - Learning about the history of math and mathematicians
- **Science Interests:**
  - Doing experiments and hands-on projects
  - Learning about space and the planets
  - Learning about living things and the human body
  - Learning about the environment and nature

**Focus Level in Class:** (Choose the option that best describes your ability to concentrate in class)

1. I can usually pay close attention throughout the entire class.
2. I can usually focus, but my mind wanders sometimes.
3. I often find it hard to keep my attention on what's being taught.
4. I really struggle to focus in class.

**Need for Repetition:** (Choose the option that best describes how you learn new things)

1. I usually understand things quickly and don't need much repetition.
2. Repeating things a few times helps me remember them better.
3. I need to go over things many times to understand and remember them.
4. Even when I repeat things, it's hard for me to remember.

**Response Speed:** (Choose the option that best describes how quickly you answer questions in class)

1. I can usually answer questions quickly and think of answers easily.
2. I usually need a little time to think before I answer.
3. It takes me a long time to come up with answers.
4. I don't often answer questions out loud.

**Stress Level:** (Choose the option that best describes how stressed you feel about learning this subject)

1. I usually don't feel stressed about this subject.
2. I sometimes feel a bit stressed when there are tests or difficult assignments.
3. I often feel stressed about this subject.
4. I feel very stressed and worried about this subject.

**Self-Assessment Scores:** (Choose the option that best describes how well you think you understand this subject)

1. I think I have a very good understanding of this subject.
2. I think I have a pretty good understanding, but there are some areas I'm unsure about.
3. I don't think I understand this subject very well.
4. I feel like I don't understand this subject at all.

**Curiosity Index:** (Choose the option that best describes your interest in learning more about this subject)

1. I'm very curious and always want to learn more.

**Team Collaboration Skills:** (Choose the option that best describes how you work in groups)

1. I work very well with others and often help lead the group.

| | |
|---|---|
| 2. I'm somewhat curious and ask questions when I'm interested.<br>3. I don't usually feel very curious about this subject.<br>4. I rarely ask questions or try to learn more on my own. | 2. I can work well in a team and contribute my fair share.<br>3. I prefer to work by myself and sometimes find it hard to work in groups.<br>**4.** I often struggle to work effectively with others in a team. |
| **Creativity Score:** (Choose the option that best describes how creative you are in this subject)<br><br>1. I often come up with new and original ideas in this subject.<br>2. I can be creative and offer some interesting ideas.<br>3. I usually follow the standard ways of doing things.<br>4. I find it hard to think creatively in this subject. | **Problem-Solving Skills:** (Choose the option that best describes how you approach problems in this subject)<br><br>1. I'm very good at solving problems and can usually figure things out.<br>2. I can usually solve problems with some guidance.<br>3. I find it hard to solve problems on my own and need a lot of help.<br>**4.** I often get stuck when trying to solve problems. |
| **Memory Retention Rate:** (Choose the option that best describes how well you remember things in this subject)<br><br>1. I can remember things for a long time, even without much review.<br>2. I remember things well if I review them occasionally.<br>3. I tend to forget things easily if I don't review them often.<br>**4.** It's hard for me to remember information in this subject, even after studying. | **Preferred Feedback Tone:** (Choose the option that best describes the kind of feedback you prefer)<br><br>1. I prefer direct feedback that tells me exactly what I need to improve.<br>2. I prefer feedback that is encouraging and tells me what I'm doing well.<br>3. I like feedback that points out both my strengths and areas for improvement in a balanced way.<br>4. I prefer gentle and supportive feedback. |
| **Energy Levels:** (Choose the option that best describes your energy levels during learning)<br><br>1. I usually have a lot of energy when learning this subject.<br>2. My energy levels are usually pretty steady.<br>3. My energy levels go up and down.<br>4. I often feel tired or low energy when learning this subject. | **Behavioral Consistency:** (Choose the option that best describes your engagement in this subject)<br><br>1. I am consistently engaged and participate regularly.<br>2. I am usually engaged, but sometimes my participation varies.<br>3. My engagement in this subject is often inconsistent.<br>**4.** I am often not engaged in this subject. |
| **Adaptability to New Topics:** (Choose the option that best describes how you handle learning new things in this subject)<br><br>1. I usually pick up new topics quickly and enjoy learning them.<br>2. I can usually learn new topics, but it might take me a little time.<br>3. I often struggle when learning new topics.<br>**4.** I find it hard to adjust when we start learning something new. | **Language Proficiency:** (Choose the option that best describes your understanding of the language used in this subject)<br><br>1. I understand the language used in this subject very well.<br>2. I understand most of the language, but sometimes I struggle with specific terms.<br>3. I have some difficulty understanding the language used in this subject. |

| | |
|---|---|
| | 4. I struggle a lot with the language used in this subject. |
| **Visual vs Verbal Preference (in Processing Information):** (Choose the option that best describes how you understand information best)<br><br>1. I understand things best when I see them (pictures, diagrams).<br>2. I understand things best when I read or hear them explained.<br>3. I learn well through both seeing and hearing/reading.<br>4. I find it hard to understand information whether it's shown or explained. | **Completion Rate of Assignments:** (Choose the option that best describes how often you finish your assignments)<br><br>1. I always finish all my assignments.<br>2. I usually finish most of my assignments.<br>3. I only finish some of my assignments.<br>4. I rarely finish my assignments. |
| **Help-Seeking Behavior:** (Choose the option that best describes when you ask for help)<br><br>1. I ask for help as soon as I realize I'm struggling.<br>2. I usually try to figure things out on my own first, but I will ask for help if I'm stuck.<br>3. I don't usually ask for help, even when I'm finding things difficult.<br>4. I avoid asking for help. | **Improvement Trends:** (Choose the option that best describes how your understanding in this subject has changed over time)<br><br>1. I feel like I'm constantly getting better at this subject.<br>2. I've improved gradually over time.<br>3. My understanding has stayed about the same.<br>4. I feel like I'm understanding this subject less than I used to. |

## PLAN OF ATTACK / IMPLEMENTATION PLAN

1. **Understand the Requirements and Scope:**
   - Thoroughly review the input (student questionnaire data) and the desired output (personalized learning paths and content recommendations).
   - Identify the key functionalities required: data ingestion, student profiling, AI-driven recommendation, learning path generation, content recommendation, and API exposure.
   - Keep in mind the need for scalability and modularity in your design.
2. **Design the System Architecture:**
   - **Identify Core Modules/Services:** Break down the system into logical, independent modules (e.g., Questionnaire Ingestion Service, Student Profiling Service, AI Recommendation Engine.
   - **Define Inter-Module Communication:** Determine how these modules will interact with each other
   - **Choose Technologies (Conceptual):** Choose suitable technologies for each module (e.g., Python/Flask/Django for backend or specific AI/ML libraries).
3. **Outline the Student Profiling Process:**
   - **Data Ingestion:** Describe how the system will receive and process the student questionnaire data
   - **Data Mapping and Transformation:** Explain how the questionnaire responses will be mapped to relevant student profile attributes (referencing the provided input parameters).
4. **Design the AI Recommendation Engine:**
   - **AI Techniques Exploration:** Research and select potential AI/ML techniques suitable for personalization (e.g., collaborative filtering, content-based filtering, knowledge-based systems, LLMs for analysis).
   - **Model Training (Conceptual):** Briefly describe how these models would be trained (what data would be used).
5. **Develop the Learning Path Generation Logic:**
   - **Input to Path Generation:** Identify what data the path generation module will use (student profile, learning goals, content catalog).
   - **Path Construction Algorithm (Conceptual):** Describe the logic for creating a sequence of learning resources, considering prerequisites, difficulty, and student preferences.

    - **Pacing Strategy:** Explain how the system would incorporate the student's attention span and motivation trends into the suggested schedule.
6. **Develop the Content Recommendation Logic:**
    - **Input to Content Recommendation:** Identify what data the content recommendation module will use (student profile, current learning stage, content metadata).
    - **Recommendation Algorithm (Conceptual):** Describe how specific content items (videos, readings, quizzes) will be selected and ranked for recommendation.
    - **Explanation Generation:** Outline how the system will provide reasons for recommending specific content.
7. **Design the API for Frontend Integration:**
    - **Define Key Endpoints:** Identify the necessary API endpoints for the LMS frontend (e.g., submitting questionnaire data, retrieving learning path, retrieving content recommendations).
    - **Specify Request and Response Formats:** Define the structure of the data exchanged through the APIs (e.g., JSON).

**Key Takeaways for Candidates:**

- **Focus on Architecture:** The emphasis is on the design and high-level implementation plan, not on writing extensive code.
- **Modular Thinking:** Break down the system into independent, manageable components.
- **AI Justification:** Clearly explain why you chose specific AI techniques.
- **Scalability Awareness:** Demonstrate an understanding of the challenges of building a scalable system.
- **Clear Communication:** Present your design in a well-organized and understandable manner.

# Assignment 2: Interactive LLM-Powered Quiz Creation

**Objective**: Design a Python-based chatbot that allows users to interactively guide the creation of a quiz using a Large Language Model (LLM). The chatbot should engage in a conversation with the user, prompting them for the desired parameters of the quiz and then generating the quiz based on the collected information

| **Input Section:** The chatbot should ask the user a series of questions to gather the following quiz parameters:<br><br>✓ **Topic:** (e.g., "Quantum Physics")<br>✓ **Difficulty:** (e.g., "easy", "medium", "hard")<br>✓ **Number of Questions:** (e.g., 3)<br>✓ **Desired Question Types:** (e.g., "multiple choice", "short answer", "true/false") - Allow the user to specify one or more.<br>✓ **Specific Sub-topics (Optional):** (e.g., "Wave-particle duality", "Quantum entanglement") - Offer the user the option to provide these.<br>✓ **Context Keywords (Optional):** (e.g., "experiment", "theory", "equation") - Offer the user the option to provide these.<br>✓ **Target Audience (Optional):** (e.g., "Undergraduate Physics Students") - Offer the user the option to specify this.<br>✓ **Language (Optional):** (e.g., "en") - Default to English if not specified.<br>✓ **Include Explanations?** (yes/no)<br>✓ **Maximum Length per Question (Optional):** (e.g., in words or tokens) - Offer the user the option to set a limit.<br><br>**Note**: Leverage **OpenAI**, **Hugging Face Transformers**, or **LangChain** for prompt-based generation or you can use following dataset for training purpose<br><br>**SQuAD 2.0 , Cosmopedia , TriviaQA , HotpotQA** | **Output Section:**<br><br>Use system prompts to control tone, difficulty, and structure.<br><br>Use mock data to test multiple topics.<br><br>Deploy it using Streamlit or Flask for demo<br><br>Parse the LLM's response to extract the question, options, and answer and show in a readable manner to a student |
|---|---|
|  <br>llm_quiz_input.json |  <br>llm_quiz_output.json |

# Plan of Attack / Implementation Plan

1. **Set Up the Development Environment:**
    - Install Python and necessary libraries (e.g., `streamlit` or `flask` for the interface, the chosen LLM library like `openai`, `transformers`, or `langchain`).
    - Obtain API keys or set up access to your chosen LLM.
2. **Design the Chatbot Interaction Flow:**
    - Plan the sequence of questions the chatbot will ask to gather quiz parameters (topic, difficulty, number of questions, question types, optional parameters).
    - Consider how the chatbot will handle user input and guide the conversation.
    - Think about how to store the collected parameters.
3. **Implement the Chatbot Interface:**
    - Choose a framework (`streamlit` or `flask` or anything else).
    - Create the basic UI elements for the chatbot (e.g., a text input field for the user, a display area for the chatbot's messages and the generated quiz).
4. **Integrate with the LLM:**
    - Write the code to interact with the chosen LLM API or library.
    - Implement the function that takes the collected quiz parameters and constructs an effective prompt for the LLM.
    - Handle the LLM's response.
5. **Parse and Format the LLM Output:**
    - Analyze the LLM's text output to extract the individual quiz questions, multiple-choice options (if applicable), and the correct answer.
    - Structure this information into a readable format (e.g., displaying questions with options and indicating the answer).
6. **Implement Mock Data Testing (Optional but Recommended):**
    - Create some hardcoded input scenarios (different topics, difficulties) to test the chatbot's ability to generate diverse quizzes before fully relying on user input.
7. **Deploy for Demo (using Streamlit or Flask):**
    - If using `streamlit`, run the Python script directly.
    - If using `flask`, set up the necessary routes and run the Flask application.
8. **Document Your Code and Approach:**
    - Add comments to your Python code explaining the logic.
    - Write a README file outlining:
        - How to run the chatbot.
        - The LLM API/library used.
        - Examples of interacting with the chatbot.
        - Your prompt engineering strategies.
        - Any challenges faced and how you addressed them.

We are looking for a clear conversational flow, effective parameter gathering, and correct LLM integration. Prioritize evaluating prompt engineering quality for relevant question generation and the ability to parse and display quiz details clearly. Code clarity and understanding the LLM use case are key. Do not focus on a polished UI, perfect quiz quality, extensive error handling, or advanced LLM techniques. The emphasis is on their approach to interactive prompting and leveraging an LLM for content generation.

# Assignment 3: Content Curation and Recommendation System

**Objective**: This assignment focuses on empowering instructors with AI tools to efficiently discover, evaluate, and recommend relevant external learning resources (beyond the LMS's built-in content) to their students.

https://aws.amazon.com/blogs/publicsector/how-the-ai-for-teaching-learning-framework-on-aws-is-transforming-the-student-and-teacher-experience/

## Input Section: Input Functionality

- ✓ **Instructor Profile Management**
  Capture subject expertise, preferred content type (video, article, PPT), and grade level.
  Update teaching style preferences (project-based, theory-driven, etc.).
- ✓ **Curriculum Upload or Selection**
  Allow instructors to upload a syllabus or select from pre-loaded curriculum standards (e.g., CBSE, ICSE, IB, etc.).
- ✓ **Topic or Unit Selection**
  Let instructors choose specific topics, units, or learning outcomes for which they need content.
- ✓ **Learning Objective Input**
  Instructors can define what skill or competency they want to develop in students.
- ✓ **Content Format Preference**
  Allow selection of preferred content format: textual notes, videos, interactive tools, real-world examples, simulations, etc.
- ✓ **Student Performance Feedback (Optional)**
  System accepts anonymized past student performance data to tailor recommendations.

  - ✓ Use NLP/LLM models to extract key concepts and themes from instructor inputs.
  - ✓ Query internal database and/or external APIs (YouTube EDU, Open Educational Resources, Khan Academy, etc.) for relevant materials.
  - ✓ Rank results based on:
    - Alignment to syllabus
    - Pedagogical effectiveness
    - Instructor preferences
    - Student performance feedback (if provided)

  - ✓ Provide a curated list of recommended content with brief explanations ("Why we picked this").

## Output Section:

- **Recommendation Dashboard**
  - Present recommended content via a clean, searchable, and filterable interface.
- **Content Summary and Tags**
  - Show key highlights, difficulty level, estimated time, engagement type (visual, auditory), and license (CC, open, etc.).
- **Save, Bookmark, and Rate Content**
  - Let instructors mark useful content and rate it to improve future recommendations.
- **Feedback to Improve AI**
  - Capture instructor feedback on each recommendation to fine-tune future outputs.
- **Export/Share Options**
  - Allow instructors to download or share curated playlists/collections with students or other teachers.

| | |
|---|---|
| instructor_input_sample.json | instructor_output_sample.json |

## PLAN OF ATTACK / IMPLEMENTATION PLAN

# 1. Input Collection Module

- **Instructor Profile Form**: Collect teaching subject(s), grade level, preferred content types (e.g., videos, simulations, notes).
- **Curriculum Input**:
  - Upload syllabus PDF/CSV or select from standard frameworks (e.g., CBSE, IB, etc.).
- **Topic/Unit Selection**:
  - Instructor selects specific topic or learning outcome.
- **Learning Objective Definition**:
  - Input skills to be achieved (e.g., critical thinking, conceptual clarity).
- **Preferences**:
  - Select content formats (video, article, quiz).
  - Teaching style (project-based, flipped classroom, etc.).

# 2. NLP/LLM Engine for Content Understanding

- **Text Extraction & Summarization**:
  - Use libraries like spaCy or NLTK to extract key concepts from instructor inputs (topics, learning objectives, curriculum)..
- **Topic Modeling & Keyword Extraction**:
  - Use TF-IDF, BERT embeddings, or GPT for key concept identification.
- **Intent Detection**:
  - Understand if instructor wants concept explanations, practice content, or activity-based learning materials.
- **Semantic Search:** Explore embedding models from libraries like Sentence Transformers or integrate with LLM APIs (OpenAI, Hugging Face Transformers) to perform semantic searches across resource titles and descriptions.
- **Content Analysis:** For evaluating content quality or relevance, consider using LLMs for summarization or topic modeling of retrieved resources.

# 3. Content Search & Ranking Engine

- **Query Generator**: Develop Python modules using libraries like `requests` to interact with APIs of platforms
  - Use extracted topics/keywords to auto-generate queries to:
    - YouTube EDU
    - Khan Academy
    - Open Educational Resources (OER Commons)
    - TED-Ed, Wikibooks, etc.
- **Content Fetcher**:
  - Use APIs or web scraping (where permissible) to fetch metadata (title, description, tags, format).
- **Relevance Ranking**:
  - Rank based on:
    - Curriculum alignment
    - Instructor preferences
    - Pedagogical effectiveness
    - Prior feedback scores (if applicable)
  - Use scoring logic or an ML ranking model.