INTEL AI HACKATHON PLAN  
**Core Features of the Project:**

The **Ultimate Learning Assistant** project involves the following core features:

1. **PDF Upload and Text Extraction**:  
   * Ability for users to upload study material (PDFs).
   * Extract and clean text from PDFs.
2. **Contextual Understanding and Answer Generation**:  
   * Fine-tuned transformer model (T5) to answer questions based on extracted content.
   * Provide explanations, examples, and help with dry running algorithms.
3. **Quiz Generation and Evaluation**:  
   * Automatically generate quizzes from the uploaded PDF content.
   * Evaluate the user's answers and provide feedback for improvement.
4. **Voice Activation**:  
   * Use speech-to-text functionality to activate the assistant via voice commands.
   * Convert responses from the model into speech for a fully interactive voice experience.
5. **Interactive Chat Interface**:  
   * Chat-based interaction for users to ask questions, receive explanations, and practice learning material.
6. **Feedback and Learning Path Optimization**:  
   * Analyze user answers to quizzes and provide tailored feedback.
   * Suggest areas for improvement and create personalized learning paths.

### **Detailed Hour-by-Hour Work Assignment for Each Team Member**

This schedule is designed for a **4-member team** (Anirudh, Apoorv, Nishanth, and Dhanush) working 8 hours each, for a total of 48 hours. Tasks are broken down into hourly chunks for efficient work division and to ensure each team member is responsible for specific components of the project.

### 

### **Anirudh: *PDF Handling, Text Extraction, and Data Preprocessing***

#### **Hour 1:**

* Research and finalize the tools and libraries for **PDF extraction** (e.g., PyMuPDF).
* Set up the environment and install dependencies (e.g., PyMuPDF).

#### **Hour 2:**

* Implement the **PDF upload functionality**: Create an interface for uploading PDF files.
* Extract text from a sample PDF and verify the accuracy of the extraction.

#### **Hour 3:**

* **Preprocess extracted text** by cleaning and removing irrelevant content (headers, footers, images, etc.).
* Implement a function that **splits the content** into logical sections (e.g., chapters, topics).

#### **Hour 4:**

* Test the text extraction and preprocessing on multiple sample PDFs.
* Store preprocessed text into a structured format (JSON, CSV) for future use by the model.

#### **Hour 5:**

* Set up automated testing for text extraction on new PDFs.
* Create a sample report showing the extracted content and the structured format.

#### **Hour 6:**

* Finalize the text preprocessing pipeline and ensure compatibility with the input format for the transformer model.
* Document the code and workflow for future reference.

#### **Hour 7:**

* **Collaborate with Nishanth** to ensure smooth integration with the model's input pipeline.
* Handle any feedback or issues related to text extraction and preprocessing.

#### **Hour 8:**

* Prepare and optimize the code for production deployment.
* Test with multiple PDFs to ensure robustness.

### 

### **Apoorv: *Model Fine-Tuning and Answer Generation***

#### **Hour 1:**

* Research suitable base models for the task (e.g., T5 or BART).
* Set up the development environment and install libraries (Transformers, Hugging Face).

#### **Hour 2:**

* Fine-tune the **T5 model** for **question answering** based on sample datasets like SQuAD.
* Create a dataset using text from the uploaded PDFs, ensuring the format is compatible with T5 (question-answer pairs).

#### **Hour 3:**

* Preprocess the data for fine-tuning the model (tokenization, truncation, padding).
* Start fine-tuning the T5 model on the preprocessed data.

#### **Hour 4:**

* Monitor the fine-tuning process, adjust hyperparameters, and check for overfitting.
* Begin testing the model with basic queries related to the PDF content.

#### **Hour 5:**

* Evaluate the fine-tuned model's performance on a validation dataset.
* Implement additional functionality to generate explanations, examples, and code outputs from the model.

#### **Hour 6:**

* Test the model’s ability to handle a variety of user queries (concept explanations, examples, dry runs).
* Begin collaborating with **Apoorv** to integrate the model into the larger system.

#### **Hour 7:**

* **Debugging**: Work on any issues related to model inference, ensuring it handles context well from long-form text.
* Implement additional features for model output (e.g., creating bullet points, structured explanations).

#### **Hour 8:**

* Finalize the model's deployment for user interactions.
* Write documentation on how the model processes user queries and generates responses.

### **Nishanth: *Voice Activation, User Interface, and Front-End Integration***

#### **Hour 1:**

* Research available **speech-to-text** libraries and frameworks (e.g., Google Speech API, Intel’s Speech SDK).
* Set up the speech-to-text environment and install required dependencies.

#### **Hour 2:**

* Implement a basic voice-command interface that activates the assistant when the user speaks.
* Test speech recognition using a microphone.

#### **Hour 3:**

* Integrate the voice activation system with the existing model, so commands (e.g., "Explain dynamic programming") can trigger the model’s responses.

#### **Hour 4:**

* Design a basic **chat interface** for the assistant (using libraries like Flask, React, or simple command-line-based for testing).
* Connect the chat interface with the **model backend** to send and receive queries.

#### **Hour 5:**

* Enhance the **user interface** to show answers, explanations, and quiz options.
* Add buttons for voice commands (e.g., "Explain", "Test me", etc.).

#### **Hour 6:**

* Integrate **text-to-speech** functionality to read out answers and explanations to the user using libraries like **gTTS** or Intel's speech SDK.
* Ensure smooth synchronization between text and voice outputs.

#### **Hour 7:**

* Conduct **usability testing**: Make sure voice commands, chat interactions, and the assistant’s responses work seamlessly.
* Gather feedback and debug issues in the UI/UX.

#### **Hour 8:**

* Finalize the **front-end integration**, ensuring the chat interface and voice activation are fully functional.
* Document the integration process and prepare the user interface for deployment.

### **Dhanush: *Quiz Generation, Answer Evaluation, and Feedback System***

#### **Hour 1:**

* Research techniques for **quiz generation** from study material, focusing on question creation.
* Write initial functions for generating multiple-choice questions based on the extracted PDF content.

#### **Hour 2:**

* Implement **quiz generation** logic that produces questions and stores them in a structured format.
* Develop functions for evaluating the correctness of answers, comparing user answers to correct answers.

#### **Hour 3:**

* Set up a **feedback mechanism** that evaluates the correctness of user responses and provides suggestions for improvement.
* Develop features to **track user progress** (e.g., number of correct/incorrect answers).

#### **Hour 4:**

* Test the quiz generation and evaluation system with sample content.
* Debug any issues with question generation or answer evaluation.

#### **Hour 5:**

* Integrate the **quiz generation system** with the model’s backend, enabling users to quiz themselves on the material.
* Develop options for the assistant to provide hints or explanations after each question.

#### **Hour 6:**

* Work with **Nishanth** to integrate the quiz system with the voice interface. Allow users to take quizzes via voice commands.

#### **Hour 7:**

* Test the entire quiz flow with a group of test users, gathering feedback on the experience.
* Ensure feedback is actionable, providing personalized suggestions for improvement.

#### **Hour 8:**

* Finalize the **feedback system** that suggests areas for improvement based on quiz performance.
* Prepare for integration into the main system, ensuring everything functions together.

### **Final Integration (Collaboration of All Team Members):**

After each member completes their tasks, the final integration of all components should follow:

* Integrate **text extraction**, **model responses**, **quiz generation**, **voice activation**, and the **feedback system** into a unified application.
* Test the complete flow from **PDF upload** to **voice interaction** and **quiz evaluation**.
* Conduct debugging, usability tests, and optimize the system for production.

.