AI Smart Interviewer & Proctoring System

A Syllabus-Compliant Deep Learning Implementation

1. Project Overview & Story

This project began as a standard "Voice-to-Text" bot using pre-built libraries. However, upon reviewing the **Deep Learning Course Syllabus**, we realized that simply *using* existing AI (like Whisper) does not demonstrate the required understanding of how Neural Networks actually "learn."

The Goal: We are transforming this application from a "Wrapper" (which just calls external tools) into a **"White Box" Deep Learning Project**. We will build, define, and train our own Neural Networks from scratch to satisfy academic requirements.

---

2. The Syllabus vs. The Project Plan

To ensure this project gets full marks, we have mapped every feature we plan to build directly to your course modules.

Course Module	Required Concept	Our Implementation Plan
Module 1	Perceptrons, MLPs, Vectorization	**The "Interviewer Brain":** We will build a Multi-Layer Perceptron (MLP) to classify user answers into topics (e.g., "Frontend" vs "Backend").
Module 2	Feedforward NNs, Backpropagation	**Manual Training Loop:** We will code the feedforward and backpropagation steps to train our text classifier.
Module 3	Optimization (GD, Adam, RMSProp)	**Custom Optimizers:** We will experiment with different optimizers (SGD vs Adam) to train our "Brain" model and plot the loss curves.
Module 4	Convolutional Neural Networks (CNNs)	**The "Proctoring Eye":** We will build a Custom CNN (LeNet style) to analyze webcam video and detect if the candidate is looking away (cheating).
Module 6	Regularization (Dropout, Data Augmentation)	**Anti-Overfitting:** We will apply Dropout layers to our CNN to prevent it from memorizing specific background images.

---

3. Current State vs. Future State

❌ Phase 1: Current State (The "Black Box")

Currently, the app works, but it relies entirely on pre-trained external libraries:

1. **Hearing:** Uses `OpenAI Whisper` (Pre-trained).
2. **Thinking:** Uses `Sentence-Transformers` (Pre-trained) and hardcoded `if/else` logic for questions.
3. **Speaking:** Uses Windows TTS (Not AI).

Critique: This is an "Engineering" project, not a "Deep Learning" project. It shows we know how to *code*, but not how to *train* AI.

✅ Phase 2: Planned State (The "White Box")

We are moving to a system where **we own the models**:

1. **Adaptive Questioning (The Brain):** Instead of a fixed list of questions, a Neural Network (MLP) will take the vector of your answer and predict the best *Next Topic ID*.
    - **Advanced Multi-Intent Support:** Unlike basic bots that treat topics in silos, our model recognizes **Mixed Concepts**. If a user says *"I prefer Java over Python,"* the model detects both labels `['Java', 'Python']`. This allows the bot to ask sophisticated comparative questions (e.g., *"How does Java's memory management differ from Python's?"*).
    - *Example:* You answer "I use React with SQL." -> Model detects both -> Maps to "Ask about Full Stack Integration".
2. **Real-time Proctoring (The Eye):** A Computer Vision model (CNN) will run in the background. It will warn the user: *"Warning: Eye contact lost. Suspicious behavior detected."*

---

4. How We Will Build It (Abstract Plan)

The "Brain" (Text Classification MLP)

- **Data Source:** We will create a small synthetic dataset.

    - *Input:* "I am good at SQL", "Tables are normalized", "I use MongoDB"
    - *Label:* "Database_Topic"
- **Architecture:** We will use PyTorch/TensorFlow to define `Linear` layers with `ReLU` activation.
- **The "Deep" Part:** We will write the training loop to minimize `CrossEntropyLoss`.

The "Eye" (Vision CNN)

- **Data Source:** We will write a script to capture 100 webcam photos of you "Looking Straight" and 100 "Looking Away."
- **Architecture:** We will define `Conv2d` layers (to find edges/eyes) -> `MaxPool` (to reduce size) -> `Linear` (to classify).
- **The "Deep" Part:** We will implement `Dropout` to ensure the model learns "Face Direction" and not just "Background colors."

---

5. Industry Research: How Real Bots Work

We researched how companies like **HireVue**, **Mettl**, and **Pymetrics** build these systems.

A. Multimodal Deep Learning

Real industry bots don't just listen; they watch. They use **Multimodal Learning**, meaning they combine three streams of data:

1. **Text (NLP):** What did you say? (Correctness)
2. **Audio (Signal Processing):** How did you say it? (Confidence, Tonal variation)
3. **Video (Computer Vision):** What was your body language? (Micro-expressions, Eye contact)

B. Adaptive Testing (CAT - Computerized Adaptive Testing)

Real bots use **Item Response Theory (IRT)**. They don't have a fixed question list.

- They assign a difficulty score to every question.
- If you get a question right, the "next question model" selects a harder question to maximize "Information Gain."
- Our planned **MLP Intent Classifier** mimics this by dynamically choosing topics based on your input.

C. Cheating Detection

Industry proctoring uses **Object Detection (YOLO)**.

- They detect: Cell phones, Second person in the frame, Books.
- They track: Gaze vectors (mathematical lines drawn from your pupils to see where you are looking).
- Our planned **CNN Proctor** is a simplified, academic version of this Gaze Tracking.

---

6. Conclusion

By implementing the **Proctoring CNN** and the **Adaptive MLP**, this project transforms from a simple script into a robust demonstration of **Deep Learning fundamentals**. It moves beyond simply "detecting text" to actually "understanding context" (via MLP) and "seeing the world" (via CNN), perfectly matching the modules in your syllabus.