

style: @import "custom.css";

Interactive AI Demos

Lab Session

CS 203: Software Tools and Techniques for AI

Prof. Nipun Batra, IIT Gandhinagar

Lab Objectives

1. **Streamlit Basics:** Build simple, interactive data apps.
2. **Model Integration:** Connect a backend ML model to a frontend UI.
3. **State Management:** Handle chat history and session variables.
4. **Deployment:** Deploy your app to the cloud (Streamlit Cloud or Hugging Face Spaces).

Setup

```
pip install streamlit gradio pandas numpy matplotlib  
pip install transformers torch  
pip install youtube-transcript-api
```

Exercise 1: Sentiment Analysis Dashboard (45 min)

Goal: Build a text analysis tool with visualization.

1. **Input:** Text area for user input.
2. **Model:** Use a simple rule-based sentiment or a pre-trained Transformer (e.g., `distilbert`).
3. **Output:**
 - Sentiment Score (Polarity).
 - Bar chart of confidence scores.
 - History table of previous queries (using `st.session_state`).

Hint:

```
import streamlit as st
from transformers import pipeline
```

Exercise 2: Image Classification App (45 min)

Goal: Upload an image and get a classification.

1. **Widget:** `st.file_uploader` (accepts png, jpg).
2. **Display:** Show the uploaded image.
3. **Model:** Use a pre-trained ResNet/EfficientNet from `torchvision` or `transformers`.
4. **Process:**
 - Resize image.
 - Normalize.
 - Inference.
5. **Output:** Top-3 predicted classes with probabilities.

Exercise 3: YouTube Video Summarizer (60 min)

Goal: A complete GenAI application.

1. **Input:** YouTube URL.

2. **Backend:**

- Extract video ID.
- Get transcript using `youtube_transcript_api`.
- Send transcript to an LLM (Gemini/OpenAI) for summarization.

3. **UI:**

- Show video thumbnail.
- "Summarize" button (with `st.spinner`).
- Display summary in a nice markdown box.
- **Bonus:** Add a "Chat with Video" feature using `st.chat_message`.

Exercise 4: Deployment (30 min)

Goal: Share your work with the world.

Option A: Streamlit Cloud

1. Push your code to GitHub.
2. Include `requirements.txt`.
3. Connect Streamlit Cloud to your repo.

Option B: Hugging Face Spaces

1. Create a new Space (SDK: Streamlit).
2. Upload `app.py` and `requirements.txt`.

Deliverable: A live public URL to your app.

Submission

Submit the following:

1. GitHub Repository Link.
2. Live App URL (if deployed).
3. Screenshots of your 3 apps running locally.