Requirments

Goal:

An AI-powered web platform that can take a **video**, extract **audio**, **visual**, and **text-based insights**, and display them in an interactive dashboard.

It will use **Computer Vision**, **Automatic Speech Recognition**, and **Natural Language Processing** — all integrated via Flask backend and Hugging Face models.

1. Functional Requirements

A. **User Interface (Frontend)**

1. Video Upload Page:
    - Form to upload `.mp4` / `.mov` / `.avi` files
    - Display upload progress bar
    - Send video to Flask backend via POST request
2. Dashboard Page:
    - Shows video preview
    - Shows:
        - Transcribed text
        - Summary of the meeting/event
        - Detected objects/actions in scenes
        - Sentiment analysis (positive/neutral/negative chart)
        - Frequently mentioned keywords
    - Implemented using **HTML, CSS, JS** and **Chart.js or D3.js**

B. **Backend (Flask)**

1. **Video Handling:**

    - Receive uploaded video and save it temporarily
    - Extract audio using MoviePy or FFmpeg
    - Extract frames every few seconds for image analysis

2. **AI Processing Pipeline:**
    - **Audio → Text:** Use Whisper or any Hugging Face ASR model
    - **Video Frames → Vision Models:** Detect people, gestures, or scenes using Hugging Face CV models (like `facebook/detr-resnet-50`)
    - **Text → Insights:**
        - Summarization → `facebook/bart-large-cnn` or `t5-base`
        - Sentiment → `distilbert-base-uncased-finetuned-sst-2-english`
        - Keyword extraction → simple NLP or embedding-based filtering

3. **Result Packaging:**
    - Combine results into a JSON structure
    - Send JSON to frontend via REST API

C. **Database (Optional but Recommended)**

- **SQLite / PostgreSQL** for storing:
    - Uploaded video metadata (filename, date, status)
    - Transcripts, summaries, and analytics results
    - User session data (if login is added)

D. **Visualization (Frontend Integration)**

- Use **Chart.js** or **Plotly.js** to display:
    - Sentiment pie chart
    - Keyword frequency bar chart
    - Timeline chart (e.g., when certain actions or topics occurred

2. Non-Functional Requirements

- **Scalability:** Modular pipeline so components (audio, video, text) can run independently.
- **Performance:** Handle videos up to ~2 minutes efficiently.
- **Accuracy:** Use pre-trained Hugging Face models to ensure reliability.
- **Usability:** Simple and responsive web UI.
- **Security:** Validate uploaded files and sanitize all inputs.

3. Team Work Division (3-Person Setup)

👱‍♂️ Team Member 1 – Backend & AI Pipeline Lead

Responsibilities:

- Set up Flask server and REST routes (`/upload`, `/process`, `/results`)
- Integrate Hugging Face models:
  - ASR (Whisper)
  - Vision (Object Detection or Image Captioning)
  - NLP (Summarization + Sentiment)
- Handle frame/audio extraction (MoviePy/FFmpeg)
- Combine outputs into a unified JSON response

Key Skills: Python, Flask, Hugging Face, MoviePy, REST APIs

👩 Team Member 2 – Frontend & Visualization Lead

Responsibilities:

- Build upload + dashboard pages (HTML, CSS, JS)
- Design clean UI and responsive layout
- Integrate JS with Flask API (fetch and display JSON results)
- Build data visualizations:
  - Sentiment pie chart
  - Word cloud / keyword bar chart

    - Summary display card

Key Skills: HTML, CSS, JavaScript, Chart.js / D3.js

👨‍💻 Team Member 3 – Data & Integration Engineer

Responsibilities:

- Handle data storage (SQLite/PostgreSQL)
- Manage backend logic for storing and retrieving analysis results
- Create schema for user uploads, transcripts, and insights
- Optimize model outputs for accuracy and consistency
- Assist others with connecting backend JSON → frontend JS

Key Skills: Python, SQL, Flask, Data handling

🧱 4. Suggested Folder Structure

```
insightxr/
│
├── backend/              ← Flask backend (AI + APIs)
│   ├── app.py
│   ├── requirements.txt
│   ├── models/           ← ML model loaders (Hugging Face, Whisper)
│   │   ├── asr_model.py
│   │   ├── vision_model.py
│   │   └── nlp_model.py
│   ├── utils/            ← Helper scripts for processing
│   │   ├── audio_utils.py
│   │   ├── video_utils.py
│   │   └── text_utils.py
│   ├── routes/           ← Flask Blueprints (upload, process, results)
│   │   ├── upload_routes.py
│   │   ├── process_routes.py
│   │   └── results_routes.py
│   ├── static/           ← (Optional) static files (if needed)
│   └── instance/         ← (Optional) DB or config files
│
└── frontend/             ← React frontend
    ├── package.json
    ├── public/
    │   └── index.html
    └── src/
        ├── App.js
        ├── index.js
        ├── components/
        │   ├── UploadForm.jsx
        │   ├── Dashboard.jsx
        │   ├── InsightCard.jsx
        │   ├── Charts/
        │   │   ├── SentimentChart.jsx
        │   │   └── KeywordCloud.jsx
        ├── pages/
        │   ├── Home.jsx
        │   └── Results.jsx
        ├── styles/
        │   ├── App.css
        │   ├── Dashboard.css
        │   └── UploadForm.css
        └── services/
            └── api.js        ← Handles Flask API calls (axios/fetch)
```

🧰 5. Required Python Packages

```
flask
transformers
torch
opencv-python
moviepy
ffmpeg-python
pandas
```

```
numpy
chart-studio
```

🧠 6. Project Milestones

Week	Goal	Owner
1	Setup Flask + HTML Upload Page	Frontend Lead
2	Implement video/audio extraction	Backend Lead
3	Integrate ASR model (Whisper)	Backend Lead
4	Add NLP summarization + sentiment	Data Engineer
5	Add vision model + frame analysis	Backend Lead
6	Design dashboard + connect APIs	Frontend Lead
7	Polish UI + test full pipeline	Everyone
8	Deploy to Render / Vercel + Final Report	Everyone