1. 1st workflow

Deepfake Detector Project

├── Goal: Detect digitally manipulated videos (Deepfakes)

├── Main Components

│ ├── Deepfake Detector Django Application

│ │ ├── Framework: Django

│ │ │ └── Purpose: Web application framework

│ │ ├── Backend Services: Firebase

│ │ │ ├── Firebase Authentication

│ │ │ │ └── Purpose: User account management

│ │ │ └── Firestore

│ │ │ └── Purpose: Application data storage

│ │ ├── Programming Language: Python

│ │ ├── Core Python Libraries

│ │ │ ├── django

│ │ │ ├── firebase-admin

│ │ │ ├── Pillow (PIL)

│ │ ├── Frontend Technologies (via Django)

│ │ │ ├── HTML

│ │ │ ├── CSS

│ │ │ └── JavaScript

│ │ └── Workflow

│ │ ├── User Interaction (Browser)

│ │ ├── Authentication (Firebase)

│ │ ├── Video Upload

│ │ ├── Backend Processing (Django + ML Model)

│ │ ├── Prediction Retrieval

│ │ └── Results Display

│ └── Machine Learning (ML) Model

│ ├── Framework: PyTorch

│ │ └── Purpose: Deep learning model development

│ ├── Key Python Libraries

│ │ ├── OpenCV (`cv2`)

│ │ │ ├── Purpose: Video processing (frame extraction)

│ │ ├── face\_recognition

│ │ │ ├── Purpose: Face detection

│ │ ├── Pillow (PIL)

│ │ │ ├── Purpose: Image manipulation

│ │ ├── NumPy

│ │ │ ├── Purpose: Numerical operations

│ │ ├── Matplotlib

│ │ │ ├── Purpose: Plotting training data

│ │ ├── Seaborn

│ │ │ ├── Purpose: Statistical visualizations

│ │ ├── pandas

│ │ │ ├── Purpose: Data manipulation (training labels)

│ │ ├── torch

│ │ │ └── Purpose: Core PyTorch tensors and functions

│ │ └── torchvision

│ │ └── Purpose: Datasets and model architectures (ResNeXt)

│ ├── Model Architecture

│ │ ├── CNN (ResNeXt - Pre-trained)

│ │ │ └── Purpose: Spatial feature extraction

│ │ └── LSTM

│ │ └── Purpose: Temporal feature analysis

│ └── Training Process

│ ├── Data Preparation

│ ├── Batching

│ ├── Loss Function (CrossEntropyLoss)

│ ├── Optimizer (Adam)

│ └── Evaluation (Accuracy, Confusion Matrix)

1. 2nd Workflow

Deepfake Detector Project

├── Goal: Detect digitally manipulated videos (Deepfakes)

├── Main Components:

│ ├── 1. Deepfake Detector Django Application (User Interface & Backend Logic)

│ │ ├── Framework: Django (Python Web Framework)

│ │ │ ├── Purpose: Provides structure for building the web application

│ │ │ ├── Handles: User requests, routing, web page rendering, integration with backend logic

│ │ ├── Backend Services: Firebase (Cloud Platform)

│ │ │ ├── Firebase Authentication

│ │ │ │ ├── Purpose: Manages user accounts (signup, login)

│ │ │ │ ├── Data Storage: Likely stores user credentials (password hashes in Firestore)

│ │ │ ├── Firestore (NoSQL Database)

│ │ │ │ ├── Purpose: Stores application data

│ │ │ │ ├── Example Data: User information, potentially analysis metadata

│ │ ├── Programming Language: Python

│ │ ├── Core Python Libraries Used:

│ │ │ ├── django (as mentioned above)

│ │ │ ├── firebase-admin

│ │ │ │ ├── Purpose: Allows the Django application to interact with Firebase services securely

│ │ │ ├── Pillow (PIL)

│ │ │ │ ├── Purpose: Basic image manipulation (potentially for user profile pictures or video frame previews)

│ │ ├── Frontend Technologies (Rendered by Django):

│ │ │ ├── HTML

│ │ │ │ ├── Purpose: Structure of web pages

│ │ │ ├── CSS

│ │ │ │ ├── Purpose: Styling and visual presentation of web pages

│ │ │ ├── JavaScript

│ │ │ │ ├── Purpose: Interactive elements and dynamic behavior on web pages

│ │ ├── Workflow:

│ │ │ ├── User Accesses Application (via Browser)

│ │ │ ├── User Logs In/Signs Up (Firebase Authentication handles this)

│ │ │ ├── User Uploads Video

│ │ │ ├── Django Backend Receives Video

│ │ │ ├── Django Backend Processes Video (likely calls the trained ML model)

│ │ │ ├── Receives Prediction from ML Model

│ │ │ ├── Django Backend Displays Results to User

│ ├── 2. Machine Learning (ML) Model for Deepfake Detection (The "Brain")

│ │ ├── Core Deep Learning Framework: PyTorch

│ │ │ ├── Purpose: Building, training, and evaluating the neural network

│ │ │ ├── Components: Defines the CNN (ResNeXt) and LSTM layers

│ │ ├── Key Python Libraries for Data Handling & ML:

│ │ │ ├── OpenCV (`cv2`)

│ │ │ │ ├── Purpose: Reading video files, extracting frames

│ │ │ ├── face\_recognition

│ │ │ │ ├── Purpose: Detecting faces in video frames

│ │ │ ├── Pillow (PIL)

│ │ │ │ ├── Purpose: Processing image frames (resizing, etc.)

│ │ │ ├── NumPy

│ │ │ │ ├── Purpose: Numerical operations, handling multi-dimensional arrays (tensors)

│ │ │ ├── Matplotlib

│ │ │ │ ├── Purpose: Plotting and visualizing training progress (loss curves)

│ │ │ ├── Seaborn

│ │ │ │ ├── Purpose: Advanced statistical visualizations (confusion matrices)

│ │ │ ├── pandas

│ │ │ │ ├── Purpose: Data manipulation and analysis (handling labeled training data, likely from CSV files)

│ │ ├── Model Architecture:

│ │ │ ├── Convolutional Neural Network (CNN) - ResNeXt (Pre-trained)

│ │ │ │ ├── Purpose: Extracts spatial features from individual video frames

│ │ │ │ ├── Pre-trained: Learned general visual features from a large dataset

│ │ │ ├── Long Short-Term Memory (LSTM)

│ │ │ │ ├── Purpose: Processes sequences of features from multiple frames to identify temporal inconsistencies

│ │ ├── Training Process:

│ │ │ ├── Data Preparation: Loading and preprocessing video data (framing, face detection, resizing, normalization)

│ │ │ ├── Batching: Feeding data to the model in batches

│ │ │ ├── Loss Function: CrossEntropyLoss (measures the difference between predictions and true labels)

│ │ │ ├── Optimizer: Adam (adjusts model parameters to minimize loss)

│ │ │ ├── Evaluation: Using metrics like Accuracy and Confusion Matrix to assess model performance

│ │ ├── Model Output: Prediction (Real or Fake video)