**Title: Real-Time Handwritten Digit Recognition using CNN, OpenCV & Streamlit with Dockerized Deployment**

**Project Overview**

This project involves building a real-time digit recognition system using the MNIST dataset and deploying it via Docker using a Streamlit web application. The system should capture handwritten digits via webcam, recognize them using a trained CNN model, and provide results live on a web interface.

**Project Timeline & Task Breakdown**

**Phase 1: Dataset Familiarization & Preprocessing**

**Timeline:** Day 1  
**Tasks:**

* Download the MNIST dataset from keras.datasets or UCI.
* Visualize sample digits with their corresponding labels.
* Normalize the pixel values (0–255 scaled to 0–1).
* Reshape the data as required for CNN input.
* Split into training and test sets.

**Phase 2: Model Development with CNN**

**Timeline:** Day 2  
**Tasks:**

* Build a CNN architecture using Keras/TensorFlow:
  + Input layer suitable for 28x28 grayscale images.
  + Conv2D, MaxPooling, Dropout, and Dense layers.
* Train the model with training data.
* Evaluate accuracy and confusion matrix on the test set.
* Save the trained model (.h5 or .pt format).

**Phase 3: Real-time Camera Integration using OpenCV**

**Timeline:** Day 3  
**Tasks:**

* Capture live video using OpenCV.
* Provide a bounding box area for handwritten input.
* Extract the region of interest (ROI) from each frame.
* Preprocess ROI (resize to 28x28, grayscale, invert colors).
* Pass preprocessed ROI to the trained CNN model.
* Display prediction on the video feed.

**Phase 4: Streamlit App Development**

**Timeline:** Day 4  
**Tasks:**

* Design a Streamlit app UI:
  + Live webcam feed.
  + Real-time digit prediction display.
  + Option to upload an image for prediction.
* Integrate CNN model prediction logic.
* Display confidence score or probability.

**Phase 5: Docker Containerization**

**Timeline:** Day 5  
**Tasks:**

* Write a Dockerfile to:
  + Set base image (e.g., python:3.9-slim).
  + Install required libraries.
  + Copy model, app files.
  + Run Streamlit app in container.
* Build and test Docker image.
* Run the app using:  
  docker run -p 8501:8501 handwritten-app

**Phase 6: Final Testing & Enhancements**

**Timeline:** Day 6  
**Tasks:**

* Test Streamlit app on different devices.
* Enhance UI/UX (color themes, loading indicators).
* Add an optional "Draw digit" canvas using streamlit-drawable-canvas.
* Optional: Upload predictions to a simple SQLite or CSV file.

**Appendices**

**Appendix A: Tools & Libraries**

* Python, Keras/TensorFlow
* OpenCV
* Streamlit
* Docker
* streamlit-drawable-canvas (optional)

**Appendix B: Deliverables**

* Trained CNN model file
* Streamlit app code
* Dockerfile
* Screenshot of running app
* Sample video/gif of webcam demo
* README file with steps to run

**Appendix C: Evaluation Criteria**

* Accuracy of model
* Real-time response quality
* UI/UX design
* Docker integration
* Code modularity and documentation