# Image Classification Project - Summary

## Overview

* **GitHub Repository:** [Project Repository](https://github.com/manishsoni123/Image-Classification) (https://github.com/manishsoni123/Image-Classification)
* **Docker Images:**
  + **Backend (FastAPI):** docker pull ms241/backend:latest
  + **Frontend (Streamlit):** docker pull ms241/frontend:latest
* **Model file:**
  + In Model Folder I store pkl file and .pth file both
* **Authentication for API:**
  + **Default credentials:**
    - Username admin
    - **Password**: password123

**Remember**:**-** For Access Stremlit UI You Need to Run Backend service First on port 8090

## Project Structure

* **backend/** → FastAPI-based REST API for model inference.
* **frontend/** → Streamlit UI for image classification.
* **model/** → Stores trained model files (.pkl and .pth).
* **docker-compose.yml** → Runs backend and frontend together.
* **EDA\_training\_evaluation.ipynb** → Data analysis and model training notebook.
  + **Perform EDA**
  + **Model Training**
  + **Model Evalution**

## Dataset

* Uses **CIFAR-10** (60,000 images across 10 classes).

## Data Preprocessing

* Resized images, normalized pixel values, and applied augmentation.
* Converted images to PyTorch tensors.
* Split dataset into training, validation, and testing sets.

## Model Training

* Used **ResNet-18** CNN model with PyTorch.
* Applied early stopping and learning rate scheduling.
* Evaluated model using Accuracy, Precision, Recall, and Confusion Matrix.

## Model Deployment

* **Backend:** FastAPI-based REST API with authentication.
* **Endpoints:**
  + /health-check → API health check.
  + /predict → Upload image for classification.
* **Frontend:** Streamlit UI for easy interaction.
* **Dockerized Deployment:** Backend & frontend containerized using Docker.

## **Running Instructions (Detailed Explanation)**

**🔹 Running Without Docker (Manual Execution)**

In your local system create any directory and clone this repository to this command ( git required )

**git clone** [**https://github.com/manishsoni123/Image-Classification.git**](https://github.com/manishsoni123/Image-Classification.git)

1. **Backend (FastAPI) Setup:**
   * Navigate to the backend folder:

cd backend

* + Install dependencies:

**Create environment and active it**

python -m venv env

**For Activation Linux**:- source env/bin/activate

**For Windows** :- env/Scripts/activate

**Install Requirements**

pip install -r requirements.txt

* + Start the FastAPI application:

uvicorn app:app --host 0.0.0.0 --port 8090

* + The API will be available at: [**http://localhost:8090**](http://localhost:8090)

1. **Frontend (Streamlit) Setup:**
   * Navigate to the frontend folder:

cd frontend

* + Install dependencies:

pip install -r requirements.txt

* + Start the Streamlit application:

streamlit run app.py --server.port=8501 --server.address=0.0.0.0

* + The UI will be available at: [**http://localhost:8501**](http://localhost:8501)

**🔹 Running With Docker (Containerized Execution)**

1. In your local system create any directory and clone this repository to this command ( git required )

**git clone** [**https://github.com/manishsoni123/Image-Classification.git**](https://github.com/manishsoni123/Image-Classification.git)

1. After that Go to that directory and run this command This command is automatically build docker image and run it ( docker installation is required for this )

**docker-compose up**

If you don’t want to build image then pull form this I already Build Docker images and Push Into Docker Hub Here is Link

* + **Backend (FastAPI):** docker pull ms241/backend:latest
  + **Frontend (Streamlit):** docker pull ms241/frontend:latest

Pull That image and Run this command **docker-compose up**

## Accessing the Application

* **Frontend (Streamlit UI):** http://localhost:8501
* **Backend (FastAPI API):** http://localhost:8090/helth-check

## Authentication

* Default credentials: admin/password123 (configurable via env variables).

## Bonus Features Implemented

✅ Basic logging & error handling  
✅ Streamlit UI for user-friendly predictions  
✅ Dockerized deployment for seamless execution

## Future Enhancements

* Deploy on AWS, GCP, Azure, or Render.
* Implement Grad-CAM or SHAP for model explainability.
* Optimize model performance for real-time classification.
* Add JWT authentication for improved security.

## Conclusion

This project demonstrates an end-to-end ML pipeline from data preprocessing and model training to API deployment and frontend UI integration.