1. Load the dataset using ImageDataGenerator, ensuring it is organized in subfolders where each subfolder represents a class label.
2. Check the number of subfolders (representing classes) and the total image count to ensure dataset consistency.
3. Preprocess the images by resizing them to a uniform size (e.g., 256x256/224 x 224), applying CLAHE for contrast enhancement, normalizing pixel values to the [0, 1] range, and optionally removing noise using filters like Gaussian Blur. (Or any other suitable techniques) **(You can read your relevant topics review paper for getting idea about what kind of preprocessing suitable for your dataset)**
4. Calculate and print the PSNR or SSIM before and after preprocessing to assess improvements in image quality.
5. Split the dataset into training, validation, and testing sets, using an 80-10-10 split or another suitable ratio.
6. Apply data augmentation (e.g., rotation, flipping, zooming) only on the training set on each image to increase dataset variability. Generate Two Versions per Image Instead of applying multiple
   augmentations per image, generating exactly two or three different augmented versions per image to reduce computational cost. **(Follow cotton leaf code)**
7. Train 7-8 transfer learning models (e.g., ResNet, VGG, Inception) for 40 epochs with early stopping (patience of 5), dropout (0.5), and L2 regularization. For transformer train 4-5 models like swin tiny, tiny vit, levit, deit, mobilevit etc. (Follow brain tumor and cotton leaf transformer code) **Pure Architecture like eye disease and brain tumor code}**
8. If the accuracy is not up to the mark you can use attention mechanisms like Self Attention, Multi Head Self Attention, Convolutional Block Attention Module, Multi Scale Attention. (Like cotton leaf code) but if you use these you have to train another code without attention mechanism to show that after using attention your models performance have increase.
9. Print the classification reports for training, validation, and testing, including metrics like accuracy, precision, recall, and F1-score. Mean PPV, Mean NPV, Specificity, Mcnemar score for binary classification, Bootstrap Accuracy Test.
10. Measure and report training time, testing time, and inference time, gpu usage,mem usage.
11. Print the model summary to check the architecture details post-training. (Follow eye disease code)
12. Experiment with hybrid models or ensemble methods to combine the strengths of different models for better performance. (If your accuracy is below 98/98.5 you can implement these techniques to increase like feature fusion, soft voting,

stacking) **(Follow eye disease and cotton leaf code in last block before grad cam)**

13. Apply Grad-CAM to the best-performing model to visualize the areas it focuses on, enhancing interpretability.
14. After training all models, perform 5-fold cross-validation on your best performing model or on all the models. **(Follow cv code)**

    OPTIONAL

15. Fine-tune hyperparameters like learning rate, batch size, and dropout rate to optimize model performance with optuna.

**DO NOT FOLLOW EXTACT CODE SHARED IN THE CLASSROOM. YOU NEED TO MODIFY IT**