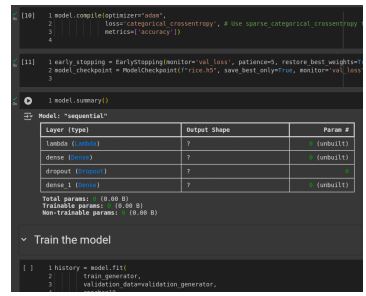
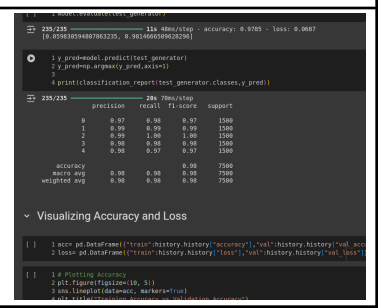
Project Development Phase
Model Performance Test

Date	7 March 2025
Team ID	PNT2025TMID00864
Project Name	GrainPalette A Deep Learning Odyssey In Rice Type Classification Through Transfer Learning
Maximum Marks	

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Sequential Model Architecture: Lambda Layer, Dense Layer, Dropout Layer, Dense Layer.	
2.	Accuracy	Training Accuracy - 97.85% Validation Accuracy -98%	
3.	Fine Tunning Result(if Done)	Validation Accuracy - N/A	N/A

[10]
```
model.compile(optimizer="adam",
              loss='categorical_crossentropy', # Use sparse_categorical_crossentropy f
              metrics=['accuracy'])

```

[11]
```
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=Tr
model_checkpoint = ModelCheckpoint(f"rice.h5", save_best_only=True, monitor='val_loss'

```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lambda (Lambda)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)
dropout (Dropout)	?	0
dense_1 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

Train the model

[ ]
```
history = model.fit(
        train_generator,
        validation_data=validation_generator,
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 11s 48ms/step - accuracy: 0.9785 - loss: 0.0687
[0.059830594807863235, 0.9814666509628296]
```

```
y_pred=model.predict(test_generator)
y_pred=np.argmax(y_pred,axis=1)

print(classification_report(test_generator.classes,y_pred))
```

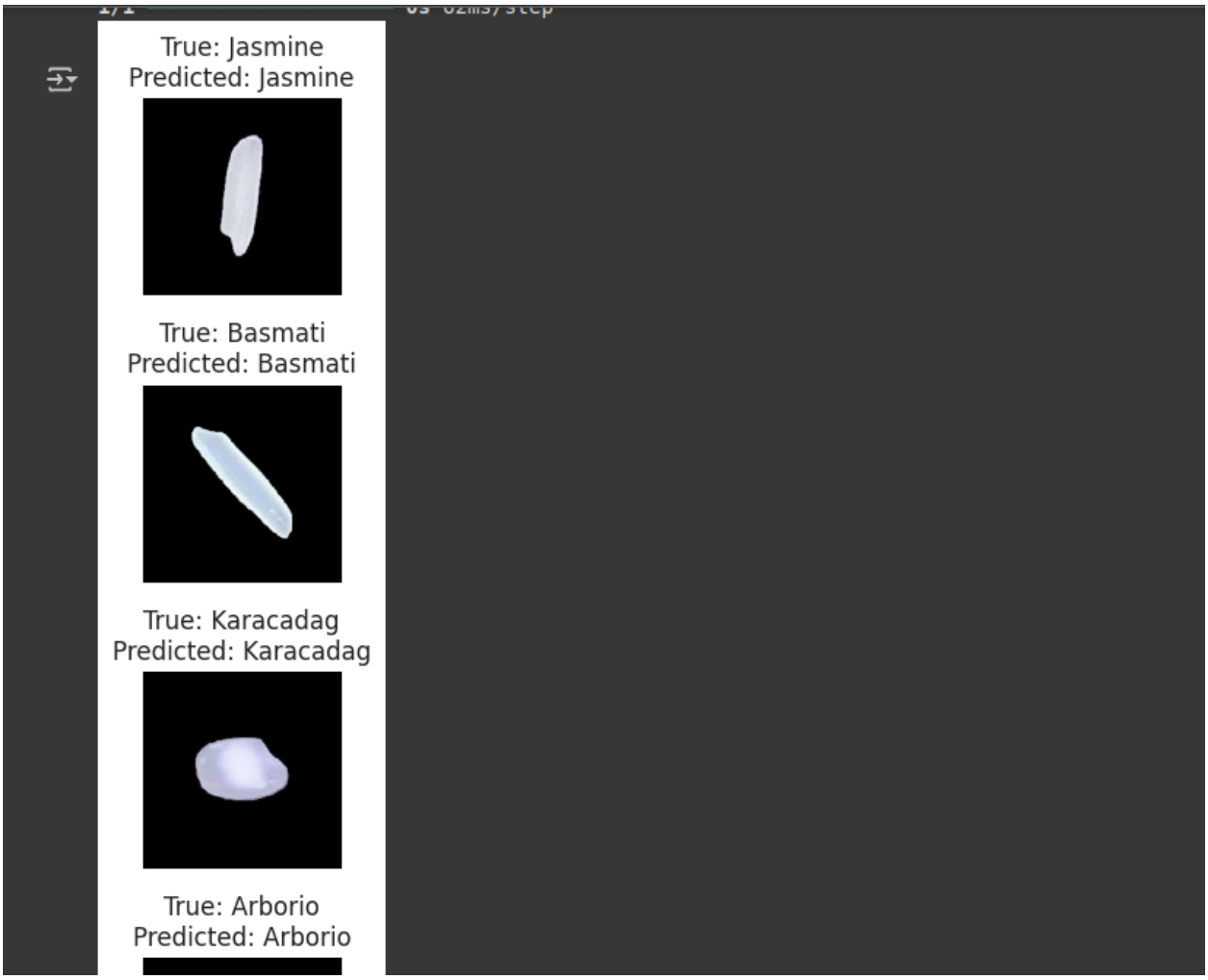
```
235/235 ━━━━━━━━━━━━━━━━━━━━ 20s 70ms/step
              precision    recall  f1-score   support

           0       0.97      0.98      0.97      1500
           1       0.99      0.99      0.99      1500
           2       0.99      1.00      1.00      1500
           3       0.98      0.98      0.98      1500
           4       0.98      0.97      0.97      1500

    accuracy                           0.98      7500
   macro avg       0.98      0.98      0.98      7500
weighted avg       0.98      0.98      0.98      7500
```

Visualizing Accuracy and Loss

[ ]
```
acc= pd.DataFrame({"train":history.history["accuracy"],"val":history.history["val_accu
loss= pd.DataFrame({"train":history.history["loss"],"val":history.history["val_loss"]}
```

[ ]
```
# Plotting Accuracy
plt.figure(figsize=(10, 5))
sns.lineplot(data=acc, markers=True)
plt.title("Training Accuracy vs Validation Accuracy")
```


Training Loss vs Validation Loss
train
val
Loss
0.14
0.12
0.10
0.08
0.06
0
1
2
3
4
5
6
7
Epochs


True: Jasmine
Predicted: Jasmine
True: Basmati
Predicted: Basmati
True: Karacadag
Predicted: Karacadag
True: Arborio
Predicted: Arborio