In [ ]:
```python
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from tensorflow.keras.applications import VGG16
import PIL
import PIL.Image
```

In [ ]:
```python
# Configure GPU settings
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if physical_devices:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)

# Load all the images
train_dir = r"C:\Users\MMANTC-STL-08\Desktop\IND_Currency_Dataset"
test_dir = r"C:\Users\MMANTC-STL-08\Desktop\test"
TARGET_SIZE = 224
BATCH_SIZE = 64
```

In [31]:
```python
#Data augmentation and loading
train_datagen = ImageDataGenerator(validation_split=0.2, rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset="training",
    shuffle=True,
    target_size=(TARGET_SIZE, TARGET_SIZE)
)
```

```
Found 1931 images belonging to 2 classes.
```

In [32]:
```python
print(train_generator.class_indices)
```

```
{'fake': 0, 'real': 1}
```

In [33]:
```python
# Using a VGG model for training
base_model = VGG16(weights='imagenet', input_shape=(TARGET_SIZE, TARGET_SIZE,
base_model.trainable = False

validation_generator = train_datagen.flow_from_directory(
    train_dir,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset="validation",
    shuffle=False,
    target_size=(TARGET_SIZE, TARGET_SIZE)
)
```

```
Found 482 images belonging to 2 classes.
```

In [34]:
```
# Adding a model on top
inputs = tf.keras.Input(shape=(TARGET_SIZE, TARGET_SIZE, 3))
x = base_model(inputs)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(16, activation='relu')(x)
output = tf.keras.layers.Dense(len(train_generator.class_indices), activation=

vgg = tf.keras.Model(inputs=inputs, outputs=output)
vgg.summary()
```

Model: "functional_1"

Layer (type)	**Output Shape**	
input_layer_3 (InputLayer)	(None, 224, 224, 3)	
vgg16 (Functional)	(None, 7, 7, 512)	
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 512)	
dense_2 (Dense)	(None, 16)	
dense_3 (Dense)	(None, 2)	

Total params: 14,722,930 (56.16 MB)

Trainable params: 8,242 (32.20 KB)

Non-trainable params: 14,714,688 (56.13 MB)

In [35]:
```
# Compile the model
opt = tf.keras.optimizers.Adam()
cce = tf.keras.losses.CategoricalCrossentropy()
vgg.compile(optimizer=opt, loss=cce, metrics=['accuracy'])
```

In [36]:
```
# Define callbacks
checkpoint_filepath = 'model.weights.h5'
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True
)
```

In [37]:
```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_
```

In [38]:
```
# Create a new test generator BEFORE the training loop
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
```

```
    target_size=(TARGET_SIZE, TARGET_SIZE),
    shuffle=False  # We don't shuffle the test data for evaluation
)
```

```
Found 177 images belonging to 2 classes.
```

In [41]:

```
# Lists to store training/validation/test accuracy and loss after each epoch
train_acc_per_epoch = []
val_acc_per_epoch = []
train_loss_per_epoch = []
val_loss_per_epoch = []
test_acc_per_epoch = []
test_loss_per_epoch = []

EPOCHS = 30
NUM_STEPS = train_generator.samples // BATCH_SIZE
VAL_NUM_STEPS = validation_generator.samples // BATCH_SIZE

# Train the model
for epoch in range(EPOCHS):
    # Train the model for one epoch
    history = vgg.fit(
        train_generator,
        epochs=1,
        steps_per_epoch=NUM_STEPS,
        validation_data=validation_generator,
        validation_steps=VAL_NUM_STEPS,
        callbacks=[reduce_lr, model_checkpoint_callback]
    )

    # Store training and validation metrics
    train_acc_per_epoch.append(history.history['accuracy'][0])  # Training acc
    val_acc_per_epoch.append(history.history['val_accuracy'][0])  # Validation
    train_loss_per_epoch.append(history.history['loss'][0])  # Training loss f
    val_loss_per_epoch.append(history.history['val_loss'][0])  # Validation lo

    # Evaluate on the test set at the end of each epoch
    test_loss, test_accuracy = vgg.evaluate(test_generator, steps=test_generat

    # Store test metrics
    test_acc_per_epoch.append(test_accuracy)
    test_loss_per_epoch.append(test_loss)

    # Print progress for the current epoch
    print(f"Epoch {epoch+1}/{EPOCHS}")
    print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.4f}")
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.8762 - loss: 0.3098 - va
l_accuracy: 0.8705 - val_loss: 0.2960 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.4375 - loss: 0.8123
Epoch 1/30
Test Loss: 0.6366, Test Accuracy: 0.5781
30/30 ━━━━━━━━━━━━━━━━━━━━ 132s 4s/step - accuracy: 0.8755 - loss: 0.2710 - va
l_accuracy: 0.8750 - val_loss: 0.2663 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.4896 - loss: 0.7384
Epoch 2/30
Test Loss: 0.5759, Test Accuracy: 0.6172
30/30 ━━━━━━━━━━━━━━━━━━━━ 132s 4s/step - accuracy: 0.8929 - loss: 0.2272 - va
l_accuracy: 0.9040 - val_loss: 0.2467 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.6771 - loss: 0.5606
Epoch 3/30
Test Loss: 0.4474, Test Accuracy: 0.7578
30/30 ━━━━━━━━━━━━━━━━━━━━ 130s 4s/step - accuracy: 0.9115 - loss: 0.2033 - va
l_accuracy: 0.9174 - val_loss: 0.2326 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.7083 - loss: 0.5230
Epoch 4/30
Test Loss: 0.4153, Test Accuracy: 0.7812
30/30 ━━━━━━━━━━━━━━━━━━━━ 128s 4s/step - accuracy: 0.9258 - loss: 0.1881 - va
l_accuracy: 0.9286 - val_loss: 0.2324 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.7917 - loss: 0.4416
Epoch 5/30
Test Loss: 0.3558, Test Accuracy: 0.8438
30/30 ━━━━━━━━━━━━━━━━━━━━ 130s 4s/step - accuracy: 0.9396 - loss: 0.1674 - va
l_accuracy: 0.9308 - val_loss: 0.2337 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.8281 - loss: 0.4023
Epoch 6/30
Test Loss: 0.3254, Test Accuracy: 0.8672
30/30 ━━━━━━━━━━━━━━━━━━━━ 129s 4s/step - accuracy: 0.9499 - loss: 0.1565 - va
l_accuracy: 0.9219 - val_loss: 0.2447 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.8594 - loss: 0.3393
Epoch 7/30
Test Loss: 0.2816, Test Accuracy: 0.8906
30/30 ━━━━━━━━━━━━━━━━━━━━ 130s 4s/step - accuracy: 0.9505 - loss: 0.1453 - va
l_accuracy: 0.9174 - val_loss: 0.2526 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.8594 - loss: 0.3162
Epoch 8/30
Test Loss: 0.2629, Test Accuracy: 0.8906
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9522 - loss: 0.1395 - va
l_accuracy: 0.9196 - val_loss: 0.2429 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.8490 - loss: 0.3289
Epoch 9/30
Test Loss: 0.2675, Test Accuracy: 0.8828
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9436 - loss: 0.1406 - va
l_accuracy: 0.9107 - val_loss: 0.2643 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9010 - loss: 0.2838
Epoch 10/30
Test Loss: 0.2354, Test Accuracy: 0.9219
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.9606 - loss: 0.1208 - va
l_accuracy: 0.9129 - val_loss: 0.2563 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9010 - loss: 0.2848
Epoch 11/30
```

```
Test Loss: 0.2326, Test Accuracy: 0.9219
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.9597 - loss: 0.1090 - va
l_accuracy: 0.8951 - val_loss: 0.3013 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9219 - loss: 0.2286
Epoch 12/30
Test Loss: 0.1958, Test Accuracy: 0.9375
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9740 - loss: 0.0987 - va
l_accuracy: 0.8973 - val_loss: 0.2682 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9010 - loss: 0.2600
Epoch 13/30
Test Loss: 0.2114, Test Accuracy: 0.9219
30/30 ━━━━━━━━━━━━━━━━━━━━ 126s 4s/step - accuracy: 0.9624 - loss: 0.1111 - va
l_accuracy: 0.8973 - val_loss: 0.2821 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9115 - loss: 0.2383
Epoch 14/30
Test Loss: 0.1947, Test Accuracy: 0.9297
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.9625 - loss: 0.1028 - va
l_accuracy: 0.8973 - val_loss: 0.2832 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9115 - loss: 0.2376
Epoch 15/30
Test Loss: 0.1919, Test Accuracy: 0.9297
30/30 ━━━━━━━━━━━━━━━━━━━━ 129s 4s/step - accuracy: 0.9633 - loss: 0.0955 - va
l_accuracy: 0.8817 - val_loss: 0.3190 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9219 - loss: 0.1994
Epoch 16/30
Test Loss: 0.1655, Test Accuracy: 0.9375
30/30 ━━━━━━━━━━━━━━━━━━━━ 126s 4s/step - accuracy: 0.9739 - loss: 0.0856 - va
l_accuracy: 0.8750 - val_loss: 0.3220 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.9219 - loss: 0.1942
Epoch 17/30
Test Loss: 0.1601, Test Accuracy: 0.9375
30/30 ━━━━━━━━━━━━━━━━━━━━ 126s 4s/step - accuracy: 0.9704 - loss: 0.0883 - va
l_accuracy: 0.8348 - val_loss: 0.3893 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9427 - loss: 0.1528
Epoch 18/30
Test Loss: 0.1335, Test Accuracy: 0.9531
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9757 - loss: 0.0844 - va
l_accuracy: 0.8705 - val_loss: 0.3165 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9219 - loss: 0.1886
Epoch 19/30
Test Loss: 0.1524, Test Accuracy: 0.9375
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9757 - loss: 0.0718 - va
l_accuracy: 0.8214 - val_loss: 0.3997 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9531 - loss: 0.1411
Epoch 20/30
Test Loss: 0.1220, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 126s 4s/step - accuracy: 0.9813 - loss: 0.0705 - va
l_accuracy: 0.8125 - val_loss: 0.4134 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9531 - loss: 0.1352
Epoch 21/30
Test Loss: 0.1166, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9802 - loss: 0.0727 - va
l_accuracy: 0.8438 - val_loss: 0.3803 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9531 - loss: 0.1485
```

```
Epoch 22/30
Test Loss: 0.1224, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 126s 4s/step - accuracy: 0.9820 - loss: 0.0587 - va
l_accuracy: 0.8438 - val_loss: 0.3782 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.9531 - loss: 0.1478
Epoch 23/30
Test Loss: 0.1211, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.9793 - loss: 0.0629 - va
l_accuracy: 0.8683 - val_loss: 0.3341 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9479 - loss: 0.1735
Epoch 24/30
Test Loss: 0.1369, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9747 - loss: 0.0672 - va
l_accuracy: 0.8304 - val_loss: 0.3964 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9531 - loss: 0.1332
Epoch 25/30
Test Loss: 0.1093, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9835 - loss: 0.0561 - va
l_accuracy: 0.8571 - val_loss: 0.3556 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9479 - loss: 0.1522
Epoch 26/30
Test Loss: 0.1207, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 127s 4s/step - accuracy: 0.9794 - loss: 0.0575 - va
l_accuracy: 0.8661 - val_loss: 0.3384 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 4s/step - accuracy: 0.9479 - loss: 0.1645
Epoch 27/30
Test Loss: 0.1286, Test Accuracy: 0.9609
30/30 ━━━━━━━━━━━━━━━━━━━━ 129s 4s/step - accuracy: 0.9844 - loss: 0.0577 - va
l_accuracy: 0.8348 - val_loss: 0.3911 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9583 - loss: 0.1361
Epoch 28/30
Test Loss: 0.1085, Test Accuracy: 0.9688
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9787 - loss: 0.0599 - va
l_accuracy: 0.8304 - val_loss: 0.4076 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9583 - loss: 0.1261
Epoch 29/30
Test Loss: 0.1012, Test Accuracy: 0.9688
30/30 ━━━━━━━━━━━━━━━━━━━━ 125s 4s/step - accuracy: 0.9827 - loss: 0.0524 - va
l_accuracy: 0.8326 - val_loss: 0.4039 - learning_rate: 0.0010
2/2 ━━━━━━━━━━━━━━━━━━━━ 7s 3s/step - accuracy: 0.9583 - loss: 0.1296
Epoch 30/30
Test Loss: 0.1030, Test Accuracy: 0.9688
```

In [46]:

```python
# Plot the training vs. validation accuracy and loss
plt.figure(figsize=(25, 15))

# Plot 1: Training vs Validation Accuracy
plt.subplot(2, 2, 1)
plt.plot(range(1, EPOCHS+1), train_acc_per_epoch, label='Training Accuracy', c
#plt.plot(range(1, EPOCHS+1), val_acc_per_epoch, label='Validation Accuracy',
plt.title('Training Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
```

```
plt.grid()

# Plot 2: Training vs Validation Loss
plt.subplot(2, 2, 2)
plt.plot(range(1, EPOCHS+1), train_loss_per_epoch, label='Training Loss', colo
#plt.plot(range(1, EPOCHS+1), val_loss_per_epoch, label='Validation Loss', col
plt.title('Training  Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()

# Plot 3: Test Accuracy per Epoch
plt.subplot(2, 2, 3)
plt.plot(range(1, EPOCHS+1), test_acc_per_epoch, label='Test Accuracy', color=
plt.title('Test Accuracy ')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.grid()

# Plot 4: Test Loss per Epoch
plt.subplot(2, 2, 4)
plt.plot(range(1, EPOCHS+1), test_loss_per_epoch, label='Test Loss', color='re
plt.title('Test Loss ')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()

plt.tight_layout()
plt.show()
plt.figure(figsize=(25, 15))

# Plot 5: Training vs Test Accuracy
plt.subplot(2, 2, 3)
plt.plot(range(1, EPOCHS+1), train_acc_per_epoch, label='Training Accuracy', c
plt.plot(range(1, EPOCHS+1), test_acc_per_epoch, label='Test Accuracy', color=
plt.title('Training vs Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.grid()

# Plot 6: Training vs Test Loss
plt.subplot(2, 2, 4)
plt.plot(range(1, EPOCHS+1), train_loss_per_epoch, label='Training Loss', colo
plt.plot(range(1, EPOCHS+1), test_loss_per_epoch, label='Test Loss', color='re
plt.title('Training vs Test Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.grid()
```

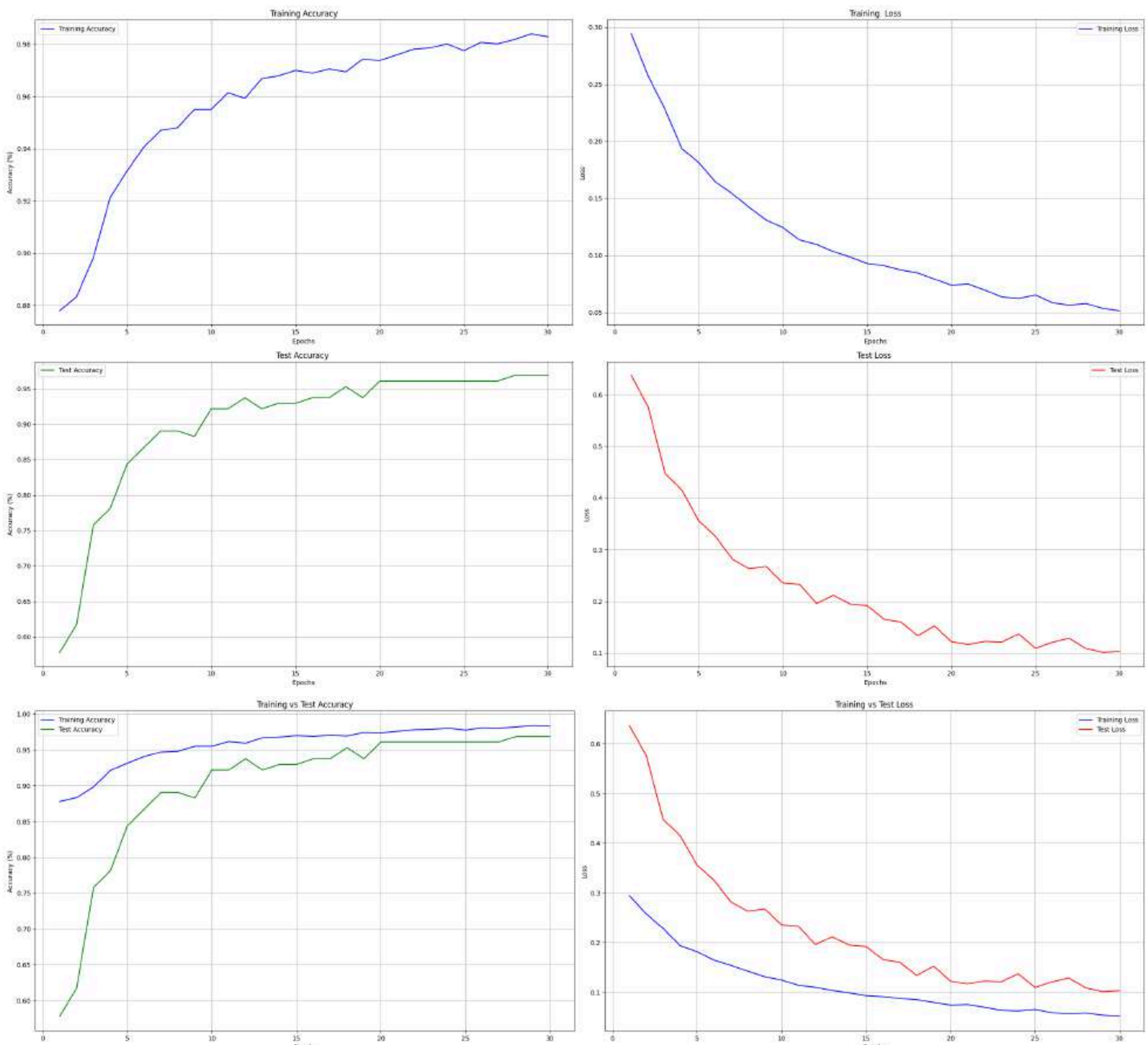
```python
# Adjust layout to ensure all plots fit nicely
plt.tight_layout()

# Show the plots
plt.show()

# Print final model loss and accuracy
final_loss = history.history['loss'][-1]  # Final training loss (from the last
final_accuracy = history.history['accuracy'][-1]  # Final training accuracy (f

print(f'Final Model Loss: {final_loss:.4f}')
print(f'Final Model Accuracy: {final_accuracy:.4f}')

# Print final model loss and accuracy
final_loss = history.history['loss'][-1]
final_accuracy = history.history['accuracy'][-1]
print(f'Final Model Loss: {final_loss:.4f}')
print(f'Final Model Accuracy: {final_accuracy:.4f}')
```



```
Final Model Loss: 0.0516
Final Model Accuracy: 0.9829
Final Model Loss: 0.0516
Final Model Accuracy: 0.9829
```

In [47]:

```python
# Function to test multiple images
import cv2
# Function to process images
def process_jpg_image(img):
    img = tf.convert_to_tensor(img[:, :, :3])
    img = np.expand_dims(img, axis=0)
    img = tf.image.resize(img, [224, 224])
    return img
def test_multiple_images(image_paths):
    predictions = []
    num_images = len(image_paths)

    # Calculate the number of rows and columns for subplots
    num_cols = 4
    num_rows = (num_images + num_cols - 1) // num_cols  # Ceiling division

    plt.figure(figsize=(15, num_rows * 5))  # Adjust height based on the numbe

    for i, path in enumerate(image_paths):
        img = cv2.imread(path)
        if img is None:
            print(f"Error loading image: {path}")
            continue

        processed_img = process_jpg_image(img)
        pred = vgg.predict(processed_img)
        prediction = int(np.argmax(pred))
        predictions.append(prediction)

        # Display the image and prediction
        plt.subplot(num_rows, num_cols, i + 1)
        plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        plt.title(f"Predicted: {class_names[prediction]}")
        plt.axis('off')

    plt.tight_layout()
    plt.show()
    return predictions  # Return predictions for further analysis
```

In [52]:

```python
# Define class names
class_names = list(train_generator.class_indices.keys())
print("Class names:", class_names)

# Example usage with multiple image paths
test_image_paths = [
          r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_3.jpg",
          r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_4.jpg",
          r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_5.jpg",
```

```
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_9.jpg" ,
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA200_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_1.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_2.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA500_12.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\New Rs 10_ Rs 50_ Rs 200 nc
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\New Rs 100 Currency Note Is
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\Reserve Bank of India - Hom
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\Rs. 100 Banknote _ INDIA ..
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\Yellow Rs 200 Notes Are Her
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\2f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\3f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\4f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\5f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\6f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\7f.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\download (2).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\download.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (1).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (3).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\images (6).jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_148.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_149.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_150.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_151.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_152.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_153.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_154.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10NEW_155.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_1.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_2.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_11.jpg",
```

```
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA10OLD_12.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA20_12.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50NEW_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA50OLD_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100NEW_11.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_3.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_4.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_5.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_6.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_7.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_8.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_9.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_10.jpg",
        r"C:\Users\MMANTC-STL-08\Desktop\dataset\INDIA100OLD_11.jpg"

]

predictions = test_multiple_images(test_image_paths)
```

```
Class names: ['fake', 'real']
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 118ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 90ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 111ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 95ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
```

```
1/1 0s 100ms/step
1/1 0s 90ms/step
1/1 0s 81ms/step
1/1 0s 79ms/step
1/1 0s 91ms/step
1/1 0s 75ms/step
1/1 0s 85ms/step
1/1 0s 92ms/step
1/1 0s 114ms/step
1/1 0s 103ms/step
1/1 0s 101ms/step
1/1 0s 77ms/step
1/1 0s 81ms/step
1/1 0s 86ms/step
1/1 0s 87ms/step
1/1 0s 114ms/step
1/1 0s 101ms/step
1/1 0s 89ms/step
1/1 0s 120ms/step
1/1 0s 132ms/step
1/1 0s 93ms/step
1/1 0s 115ms/step
1/1 0s 125ms/step
1/1 0s 91ms/step
1/1 0s 120ms/step
1/1 0s 96ms/step
1/1 0s 94ms/step
1/1 0s 78ms/step
1/1 0s 115ms/step
1/1 0s 94ms/step
1/1 0s 84ms/step
1/1 0s 110ms/step
1/1 0s 102ms/step
1/1 0s 84ms/step
1/1 0s 104ms/step
1/1 0s 118ms/step
1/1 0s 88ms/step
1/1 0s 113ms/step
1/1 0s 108ms/step
1/1 0s 103ms/step
1/1 0s 101ms/step
1/1 0s 87ms/step
1/1 0s 109ms/step
1/1 0s 89ms/step
```


Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real
Predicted: real

```
In [ ]:
```