In [1]:
```python
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
```

In [2]:
```python
(x_train, _), (x_test, _) = keras.datasets.mnist.load_data()

print(f"Original x_train shape: {x_train.shape}")
print(f"Original x_test shape: {x_test.shape}")
```

```
Original x_train shape: (60000, 28, 28)
Original x_test shape: (10000, 28, 28)
```

In [3]:
```python
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

print(f"Processed x_train shape: {x_train.shape}")
print(f"Processed x_test shape: {x_test.shape}")

input_dim = x_train.shape[1] # 784

encoding_dim = 32
```

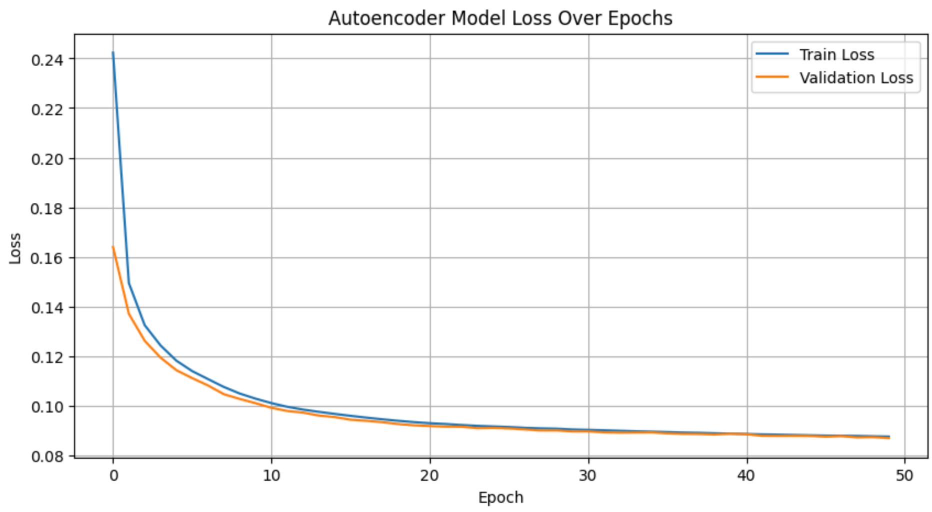
```
Processed x_train shape: (60000, 784)
Processed x_test shape: (10000, 784)
```

In [4]:
```python
input_img = keras.Input(shape=(input_dim,))
encoded = layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(encoding_dim, activation='relu')(encoded) # Bottleneck layer

decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded = layers.Dense(input_dim, activation='sigmoid')(decoded)

autoencoder = keras.Model(input_img, decoded)

encoder = keras.Model(input_img, encoded)

autoencoder.summary()
encoder.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 784)]             0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

 dense_3 (Dense)             (None, 64)                2112

 dense_4 (Dense)             (None, 128)               8320

 dense_5 (Dense)             (None, 784)               101136

=================================================================
Total params: 222,384
Trainable params: 222,384
Non-trainable params: 0
_________________________________________________________________
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 784)]             0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 32)                2080

=================================================================
Total params: 110,816
Trainable params: 110,816
Non-trainable params: 0
_________________________________________________________________
```

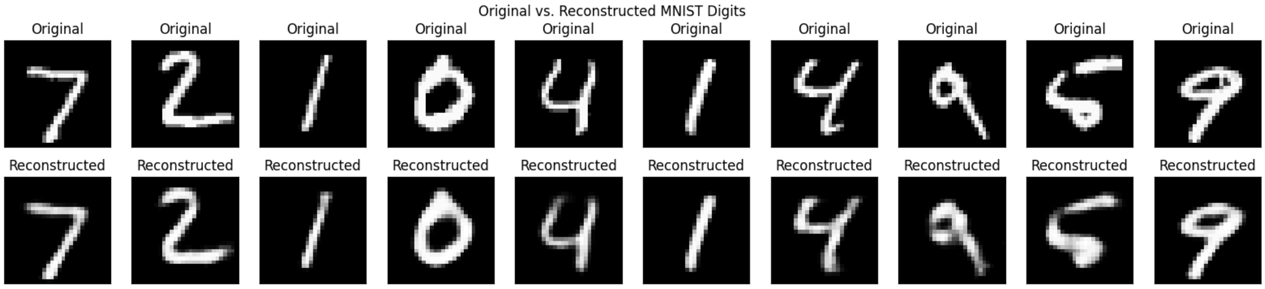
In [5]:
```python
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
history = autoencoder.fit(x_train, x_train,
                          epochs=50,
                          batch_size=256,
                          shuffle=True,
                          validation_data=(x_test, x_test),
                          verbose=1)

plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Autoencoder Model Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()
```

```
Epoch 1/50
235/235 [==============================] - 3s 7ms/step - loss: 0.2423 - val_loss: 0.1639
Epoch 2/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1494 - val_loss: 0.1370
Epoch 3/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1324 - val_loss: 0.1261
Epoch 4/50
235/235 [==============================] - 2s 6ms/step - loss: 0.1242 - val_loss: 0.1193
Epoch 5/50
235/235 [==============================] - 2s 7ms/step - loss: 0.1181 - val_loss: 0.1143
Epoch 6/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1139 - val_loss: 0.1111
Epoch 7/50
235/235 [==============================] - 2s 6ms/step - loss: 0.1107 - val_loss: 0.1081
Epoch 8/50
235/235 [==============================] - 2s 6ms/step - loss: 0.1075 - val_loss: 0.1046
Epoch 9/50
235/235 [==============================] - 2s 6ms/step - loss: 0.1048 - val_loss: 0.1027
Epoch 10/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1028 - val_loss: 0.1009
Epoch 11/50
235/235 [==============================] - 1s 6ms/step - loss: 0.1010 - val_loss: 0.0991
Epoch 12/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0995 - val_loss: 0.0978
Epoch 13/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0984 - val_loss: 0.0971
Epoch 14/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0974 - val_loss: 0.0960
Epoch 15/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0966 - val_loss: 0.0953
Epoch 16/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0958 - val_loss: 0.0943
Epoch 17/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0951 - val_loss: 0.0938
Epoch 18/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0944 - val_loss: 0.0933
Epoch 19/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0938 - val_loss: 0.0925
Epoch 20/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0933 - val_loss: 0.0920
Epoch 21/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0929 - val_loss: 0.0917
Epoch 22/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0925 - val_loss: 0.0915
Epoch 23/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0921 - val_loss: 0.0915
Epoch 24/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0918 - val_loss: 0.0909
Epoch 25/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0916 - val_loss: 0.0910
Epoch 26/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0913 - val_loss: 0.0908
Epoch 27/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0910 - val_loss: 0.0904
Epoch 28/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0908 - val_loss: 0.0899
Epoch 29/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0907 - val_loss: 0.0900
Epoch 30/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0903 - val_loss: 0.0895
Epoch 31/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0902 - val_loss: 0.0896
Epoch 32/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0900 - val_loss: 0.0892
Epoch 33/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0898 - val_loss: 0.0891
Epoch 34/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0896 - val_loss: 0.0891
Epoch 35/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0895 - val_loss: 0.0892
Epoch 36/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0893 - val_loss: 0.0888
Epoch 37/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0891 - val_loss: 0.0886
Epoch 38/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0890 - val_loss: 0.0885
Epoch 39/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0888 - val_loss: 0.0883
Epoch 40/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0887 - val_loss: 0.0886
Epoch 41/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0885 - val_loss: 0.0884
Epoch 42/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0884 - val_loss: 0.0878
Epoch 43/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0883 - val_loss: 0.0877
Epoch 44/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0881 - val_loss: 0.0877
Epoch 45/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0880 - val_loss: 0.0877
Epoch 46/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0879 - val_loss: 0.0874
Epoch 47/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0878 - val_loss: 0.0877
Epoch 48/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0877 - val_loss: 0.0872
Epoch 49/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0876 - val_loss: 0.0873
Epoch 50/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0875 - val_loss: 0.0869
```



In [7]:
```
encoded_imgs = encoder.predict(x_test)
decoded_imgs = autoencoder.predict(x_test)

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax.set_title("Original")

    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    ax.set_title("Reconstructed")
plt.suptitle('Original vs. Reconstructed MNIST Digits')
plt.show()
```

```
313/313 [==============================] - 0s 761us/step
313/313 [==============================] - 0s 1ms/step
```



In [8]:
```
test_loss = autoencoder.evaluate(x_test, x_test, verbose=0)
print(f"Final Test Reconstruction Loss (Binary Cross-entropy): {test_loss:.4f}")

mse = np.mean(np.power(x_test - decoded_imgs, 2))
print(f"Mean Squared Error (MSE) on Test Set: {mse:.4f}")
```

```
Final Test Reconstruction Loss (Binary Cross-entropy): 0.0869
Mean Squared Error (MSE) on Test Set: 0.0084
```

In [ ]: