In [28]:
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, losses
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Model
```

In [29]:
```python
# Download the dataset
dataframe = pd.read_csv('http://storage.googleapis.com/download.tensorflow.or
g/data/ecg.csv', header=None)
raw_data = dataframe.values
dataframe.head()
```

Out[29]:

	0	1	2	3	4	5	6	7	8
0	-0.112522	-2.827204	-3.773897	-4.349751	-4.376041	-3.474986	-2.181408	-1.818287	-1.250522
1	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258	-0.754680
2	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490658	-1.183580
3	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131	-1.333884
4	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423	-1.594450

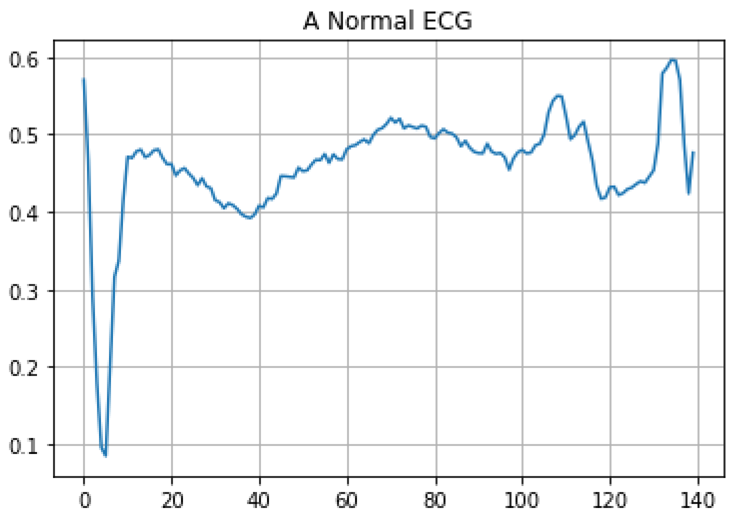
5 rows × 141 columns

In [30]:
```python
# The last element contains the labels
labels = raw_data[:, -1]

# The other data points are the electrocadriogram data
data = raw_data[:, 0:-1]

train_data, test_data, train_labels, test_labels = train_test_split(
    data, labels, test_size=0.2, random_state=21
)
```

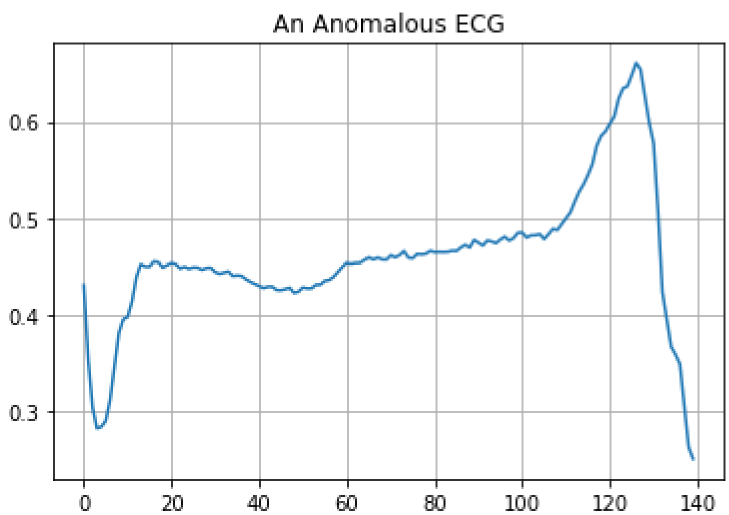
In [31]:
```python
min_val = tf.reduce_min(train_data)
max_val = tf.reduce_max(train_data)

train_data = (train_data - min_val) / (max_val - min_val)
test_data = (test_data - min_val) / (max_val - min_val)

train_data = tf.cast(train_data, tf.float32)
test_data = tf.cast(test_data, tf.float32)
```

In [32]:
```
train_labels = train_labels.astype(bool)
test_labels = test_labels.astype(bool)

normal_train_data = train_data[train_labels]
normal_test_data = test_data[test_labels]

anomalous_train_data = train_data[~train_labels]
anomalous_test_data = test_data[~test_labels]
```

In [33]:
```
plt.grid()
plt.plot(np.arange(140), normal_train_data[0])
plt.title("A Normal ECG")
plt.show()
```



In [34]:
```
plt.grid()
plt.plot(np.arange(140), anomalous_train_data[0])
plt.title("An Anomalous ECG")
plt.show()
```



In [35]:
```python
class AnomalyDetector(Model):
  def __init__(self):
    super(AnomalyDetector, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Dense(32, activation="relu"),
      layers.Dense(16, activation="relu"),
      layers.Dense(8, activation="relu")])

    self.decoder = tf.keras.Sequential([
      layers.Dense(16, activation="relu"),
      layers.Dense(32, activation="relu"),
      layers.Dense(140, activation="sigmoid")])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = AnomalyDetector()
```

In [36]:
```python
autoencoder.compile(optimizer='adam', loss='mae')
```

In [37]:

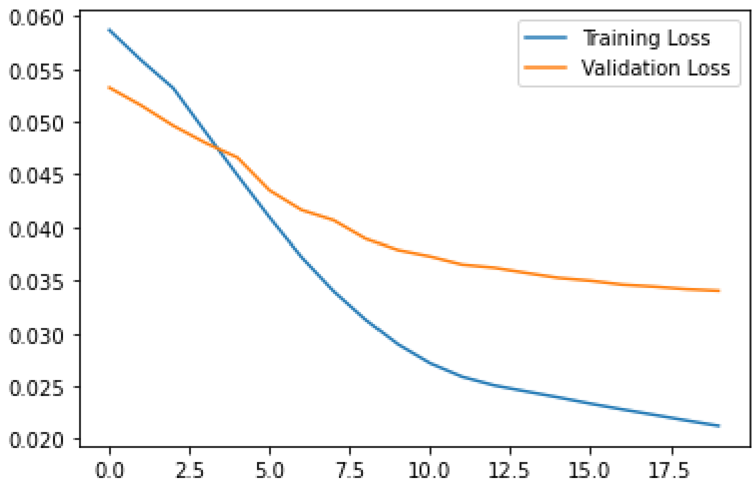
```
history = autoencoder.fit(normal_train_data, normal_train_data,
          epochs=20,
          batch_size=512,
          validation_data=(test_data, test_data),
          shuffle=True)
```

```
Epoch 1/20
5/5 [==============================] - 1s 45ms/step - loss: 0.0587 - val_los
s: 0.0532
Epoch 2/20
5/5 [==============================] - 0s 12ms/step - loss: 0.0558 - val_los
s: 0.0515
Epoch 3/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0531 - val_los
s: 0.0496
Epoch 4/20
5/5 [==============================] - 0s 12ms/step - loss: 0.0490 - val_los
s: 0.0480
Epoch 5/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0449 - val_los
s: 0.0466
Epoch 6/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0409 - val_los
s: 0.0435
Epoch 7/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0372 - val_los
s: 0.0416
Epoch 8/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0340 - val_los
s: 0.0407
Epoch 9/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0313 - val_los
s: 0.0389
Epoch 10/20
5/5 [==============================] - 0s 12ms/step - loss: 0.0289 - val_los
s: 0.0379
Epoch 11/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0271 - val_los
s: 0.0372
Epoch 12/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0259 - val_los
s: 0.0365
Epoch 13/20
5/5 [==============================] - 0s 15ms/step - loss: 0.0250 - val_los
s: 0.0362
Epoch 14/20
5/5 [==============================] - 0s 15ms/step - loss: 0.0245 - val_los
s: 0.0357
Epoch 15/20
5/5 [==============================] - 0s 14ms/step - loss: 0.0239 - val_los
s: 0.0352
Epoch 16/20
5/5 [==============================] - 0s 10ms/step - loss: 0.0233 - val_los
s: 0.0350
Epoch 17/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0228 - val_los
s: 0.0346
Epoch 18/20
5/5 [==============================] - 0s 12ms/step - loss: 0.0223 - val_los
s: 0.0344
Epoch 19/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0217 - val_los
s: 0.0342
```

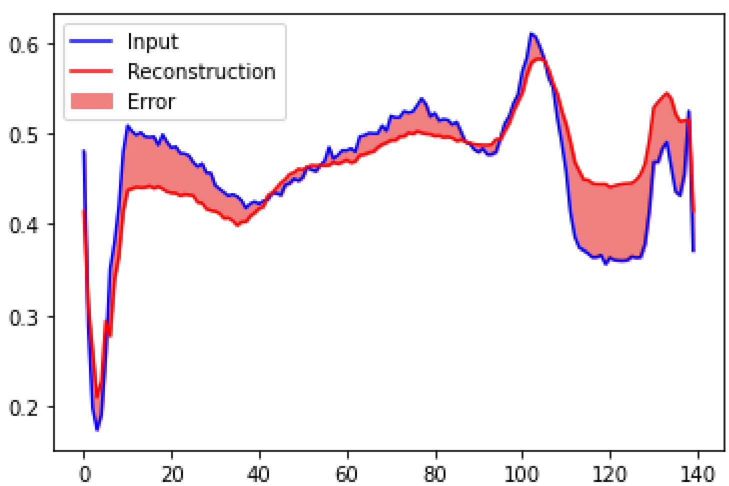
```
Epoch 20/20
5/5 [==============================] - 0s 13ms/step - loss: 0.0212 - val_los
s: 0.0340
```

In [38]:
```python
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.legend()
```

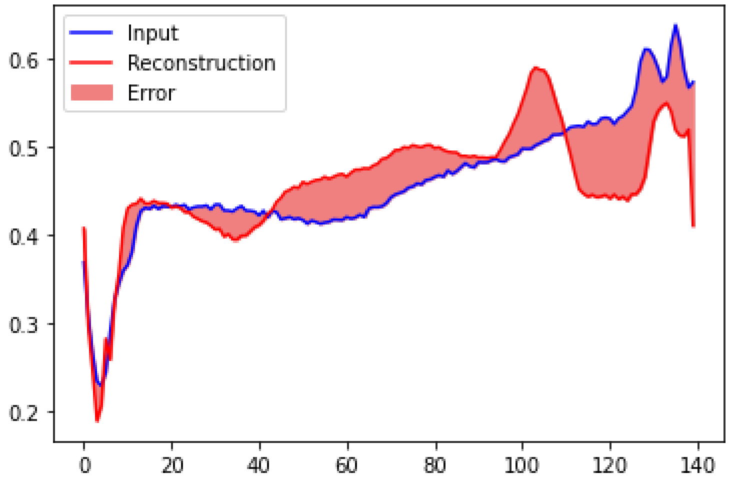
Out[38]:
```
<matplotlib.legend.Legend at 0x274c62dc5e0>
```



In [39]:
```python
encoded_data = autoencoder.encoder(normal_test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()

plt.plot(normal_test_data[0], 'b')
plt.plot(decoded_data[0], 'r')
plt.fill_between(np.arange(140), decoded_data[0], normal_test_data[0], color
='lightcoral')
plt.legend(labels=["Input", "Reconstruction", "Error"])
plt.show()
```

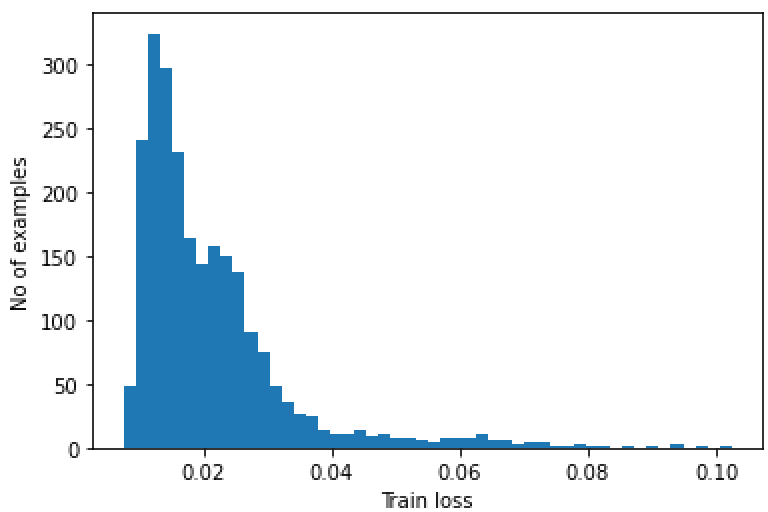


In [40]:
```
encoded_data = autoencoder.encoder(anomalous_test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()

plt.plot(anomalous_test_data[0], 'b')
plt.plot(decoded_data[0], 'r')
plt.fill_between(np.arange(140), decoded_data[0], anomalous_test_data[0], colo
r='lightcoral')
plt.legend(labels=["Input", "Reconstruction", "Error"])
plt.show()
```



In [41]:
```
reconstructions = autoencoder.predict(normal_train_data)
train_loss = tf.keras.losses.mae(reconstructions, normal_train_data)

plt.hist(train_loss[None,:], bins=50)
plt.xlabel("Train loss")
plt.ylabel("No of examples")
plt.show()
```

```
74/74 [==============================] - 0s 2ms/step
```

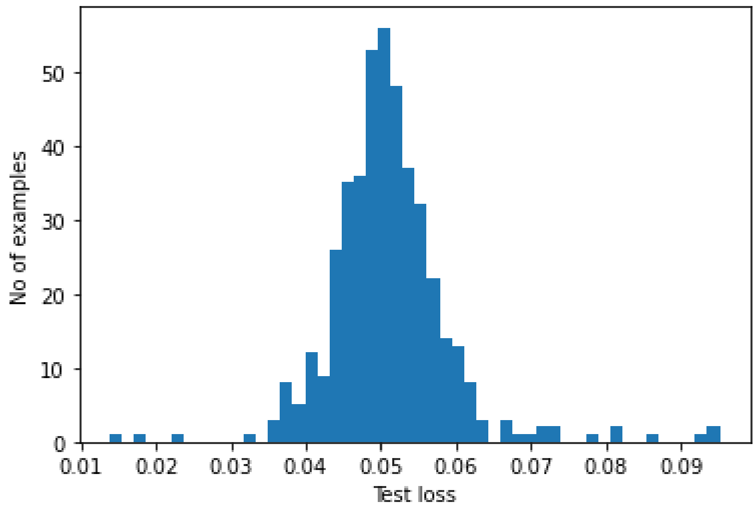


In [42]:
```
threshold = np.mean(train_loss) + np.std(train_loss)
print("Threshold: ", threshold)
```

```
Threshold:  0.03318568
```

In [43]:
```
reconstructions = autoencoder.predict(anomalous_test_data)
test_loss = tf.keras.losses.mae(reconstructions, anomalous_test_data)

plt.hist(test_loss[None, :], bins=50)
plt.xlabel("Test loss")
plt.ylabel("No of examples")
plt.show()
```

```
14/14 [==============================] - 0s 2ms/step
```



In [44]:
```
def predict(model, data, threshold):
  reconstructions = model(data)
  loss = tf.keras.losses.mae(reconstructions, data)
  return tf.math.less(loss, threshold)

def print_stats(predictions, labels):
  print("Accuracy = {}".format(accuracy_score(labels, predictions)))
  print("Precision = {}".format(precision_score(labels, predictions)))
  print("Recall = {}".format(recall_score(labels, predictions)))
```

In [45]:
```
preds = predict(autoencoder, test_data, threshold)
print_stats(preds, test_labels)
```

```
Accuracy = 0.944
Precision = 0.9921875
Recall = 0.9071428571428571
```

In [ ]:

In [ ]: