In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [19]:
```
from keras.datasets import fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

In [20]:
```
train_images
```

Out[20]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
```

In [21]:
```
test_images
```

Out[21]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
```

Normalize the Images

In [22]:
```
train_images = train_images/255.0
test_images = test_images/255.0
```

In [23]:
```
train_images.shape
```

Out[23]:
```
(60000, 28, 28)
```

Out[23]:
```
(60000, 28, 28)
```

In [24]:
```
test_images.shape
```

Out[24]:
```
(10000, 28, 28)
```

Out[24]:
```
(10000, 28, 28)
```

Define the model structure

In [25]:
```
from keras.models import Sequential
from keras.layers import Dense, Flatten

model = Sequential()
model
```

Out[25]:
```
<Sequential name=sequential_1, built=False>
```

Out[25]:
```
<Sequential name=sequential_1, built=False>
```

In [26]:
```
model.add(Flatten(input_shape=(28, 28), name="flatten"))
model.add(Dense(128, activation="relu", name="dense_1"))
model.add(Dense(10, activation="softmax", name="dense_output"))
```

```
D:\py\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
t to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
D:\py\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
t to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

Compile the model

In [27]:
```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=['accuracy'])
```

Train the model

In [28]:
```
history = model.fit(train_images, train_labels, epochs = 3, verbose = 1)
```

```
Epoch 1/3
Epoch 1/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7840 - loss: 0.6257
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7840 - loss: 0.6257
Epoch 2/3
Epoch 2/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8600 - loss: 0.3944
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8600 - loss: 0.3944
Epoch 3/3
Epoch 3/3
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8768 - loss: 0.3397
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8768 - loss: 0.3397
```

Evaluate the model

In [29]:
```
loss, accuracy = model.evaluate(test_images, test_labels)
print(loss, " ", accuracy)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8701 - loss: 0.3666
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8701 - loss: 0.3666
0.3740958571434021   0.8654999732971191
0.3740958571434021   0.8654999732971191
```

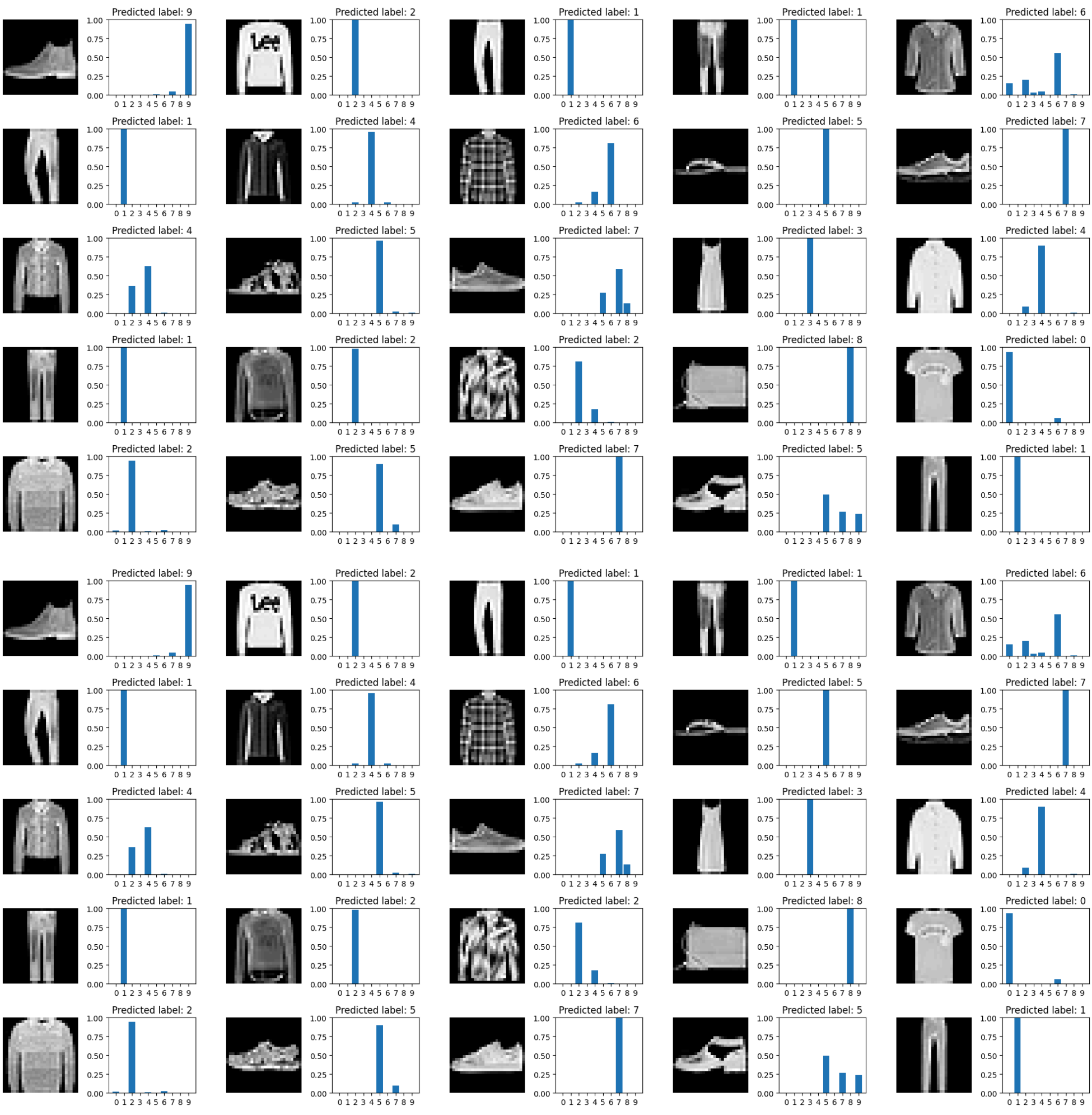
Make Predictions

In [31]:
```
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis = 1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```

Display the Predictions

In [40]:
```
rows = 5
cols = 5
num_images = rows * cols

plt.figure(figsize=(2 * 2 * cols, 2 * rows))

for i in range(num_images):

    # plot the images
    plt.subplot(rows, 2 * cols, 2 * i + 1)
    plt.imshow(test_images[i], cmap='gray')
    plt.axis('off')

    # plot the bar chart
    plt.subplot(rows, 2 * cols, 2 * i + 2)
    plt.bar(range(10), predictions[i])
    plt.xticks(range(10))
    plt.ylim([0, 1])
    plt.title(f"Predicted label: {predicted_labels[i]}")
    plt.tight_layout()

plt.show()
```



In [ ]:
```
```