In [2]:

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
from tensorflow.keras.utils import to_categorical

texts = [
    "I love this product", "This is amazing", "Absolutely terrible", "I hate it", "Not bad",
    "Could be better", "Really good job", "Awful experience", "Okayish", "Best ever",
    "Disappointed", "So happy with it", "Neutral feeling", "Worst purchase",
    "I enjoy it", "I dislike this", "Quite good", "Terrible experience", "Very happy",
    "I wouldn't recommend", "Quite bad", "Totally amazing", "Not my favorite", "Mediocre",
    "Excellent service", "Totally awful", "Totally awesome", "Worst decision ever",
    "Love it!", "Do not buy", "Best thing ever", "Very bad product"
]

labels = [1, 1, 0, 0, 2, 2, 1, 0, 2, 1, 0, 1, 2, 0, 1, 1, 0, 0, 1, 2, 1, 0, 2, 2, 0, 2, 0, 1, 1, 2, 0, 1]

tokenizer = Tokenizer(num_words=1000, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)

max_len = 10
padded = pad_sequences(sequences, maxlen=max_len)

labels_cat = to_categorical(labels, num_classes=3)

x_train, x_test, y_train, y_test = train_test_split(padded, labels_cat, test_size=0.2, random_state=42)

print("Dataset prepared and split!")
```

```
Dataset prepared and split!
```

In [3]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense, Dropout

vocab_size = len(tokenizer.word_index) + 1

model = Sequential([
    Embedding(vocab_size, 64, input_length=max_len),
    SimpleRNN(128, return_sequences=False),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])

model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

print("RNN Model defined and compiled!")
```

```
RNN Model defined and compiled!
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

In [4]:

```
history = model.fit(x_train, y_train, epochs=20, batch_size=4, validation_data=(x_test, y_test))

print("Model trained!")
```

```
Epoch 1/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 97ms/step - accuracy: 0.4666 - loss: 1.1060 - val_accuracy: 0.2857 - val_loss: 1.1239
Epoch 2/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.5916 - loss: 1.0071 - val_accuracy: 0.4286 - val_loss: 1.0871
Epoch 3/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.2703 - loss: 1.1548 - val_accuracy: 0.2857 - val_loss: 1.0935
Epoch 4/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5425 - loss: 1.0164 - val_accuracy: 0.2857 - val_loss: 1.1275
Epoch 5/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.3866 - loss: 1.0162 - val_accuracy: 0.2857 - val_loss: 1.1216
Epoch 6/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.4950 - loss: 0.9801 - val_accuracy: 0.2857 - val_loss: 1.1176
Epoch 7/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.6122 - loss: 0.8935 - val_accuracy: 0.4286 - val_loss: 1.0734
Epoch 8/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7627 - loss: 0.8604 - val_accuracy: 0.4286 - val_loss: 1.0607
Epoch 9/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.7716 - loss: 0.8596 - val_accuracy: 0.2857 - val_loss: 1.0521
Epoch 10/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.6601 - loss: 0.8339 - val_accuracy: 0.4286 - val_loss: 1.0407
Epoch 11/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6928 - loss: 0.7358 - val_accuracy: 0.4286 - val_loss: 1.0329
Epoch 12/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.8146 - loss: 0.6728 - val_accuracy: 0.5714 - val_loss: 1.0333
Epoch 13/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9018 - loss: 0.5411 - val_accuracy: 0.4286 - val_loss: 1.0445
Epoch 14/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.8964 - loss: 0.4850 - val_accuracy: 0.4286 - val_loss: 1.0647
Epoch 15/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.8737 - loss: 0.4653 - val_accuracy: 0.2857 - val_loss: 1.1092
Epoch 16/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9900 - loss: 0.4112 - val_accuracy: 0.4286 - val_loss: 1.0422
Epoch 17/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9154 - loss: 0.3378 - val_accuracy: 0.5714 - val_loss: 0.9815
Epoch 18/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9707 - loss: 0.3541 - val_accuracy: 0.5714 - val_loss: 0.9822
Epoch 19/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9603 - loss: 0.2089 - val_accuracy: 0.5714 - val_loss: 1.0055
Epoch 20/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - accuracy: 0.9447 - loss: 0.2319 - val_accuracy: 0.4286 - val_loss: 1.0155
Model trained!
```

In [5]:

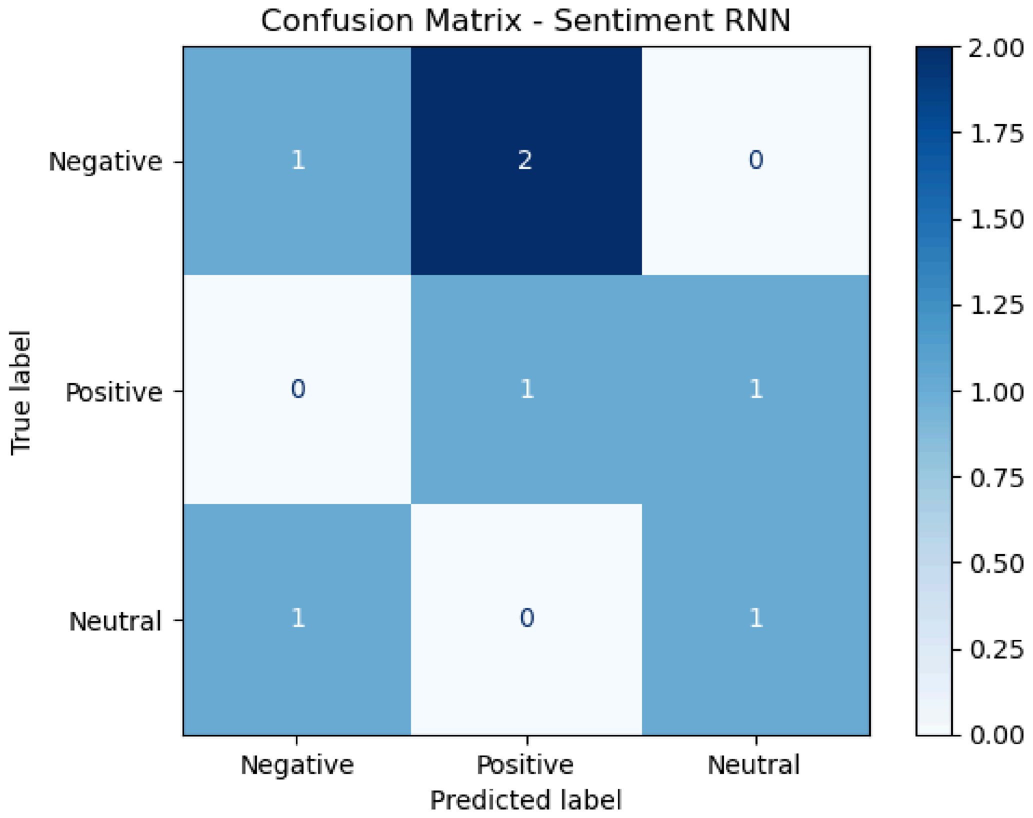
```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
print(f"Test Accuracy: {test_acc * 100:.2f}%")
```

```
Test Accuracy: 42.86%
```

In [6]:

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import numpy as np

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)

cm = confusion_matrix(y_true_classes, y_pred_classes)

unique_labels = np.unique(np.concatenate((y_true_classes, y_pred_classes)))
label_map = ["Negative", "Positive", "Neutral"]
used_labels = [label_map[i] for i in unique_labels]

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=used_labels)
disp.plot(cmap="Blues")
plt.title("Confusion Matrix - Sentiment RNN")
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 270ms/step
```



```
In [ ]:
```