In [1]:

```
from joblib import dump
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential#type:ignore
from tensorflow.keras.layers import Dense#type:ignore
from tensorflow.keras import regularizers#type:ignore
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("dataset.csv")
df = shuffle(df, random_state=42)
df.head()
```

Out[2]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	
373	Acne	skin_rash	pus_filled_pimples	scurring	NaN	NaN	NaN	
4915	Acne	skin_rash	pus_filled_pimples	blackheads	scurring	NaN	NaN	
4344	Impetigo	skin_rash	high_fever	blister	red_sore_around_nose	yellow_crust_ooze	NaN	
3554	Heart attack	vomiting	breathlessness	sweating	chest_pain	NaN	NaN	
3298	hepatitis A	joint_pain	vomiting	yellowish_skin	dark_urine	nausea	loss_of_appetite	ab

In [3]:

```
for col in df.columns:
    df[col] = df[col].str.replace('_',' ')
df.head()
```

Out[3]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Sy
373	Acne	skin rash	pus filled pimples	scurring	NaN	NaN	NaN	NaN	NaN	
4915	Acne	skin rash	pus filled pimples	blackheads	scurring	NaN	NaN	NaN	NaN	
4344	Impetigo	skin rash	high fever	blister	red sore around nose	yellow crust ooze	NaN	NaN	NaN	
3554	Heart attack	vomiting	breathlessness	sweating	chest pain	NaN	NaN	NaN	NaN	
3298	hepatitis A	joint pain	vomiting	yellowish skin	dark urine	nausea	loss of appetite	abdominal pain	diarrhoea	r

In [4]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Symp
count	4919	4919	4919	4919	4571	3714	2934	2268	1944	
unique	41	34	48	54	50	38	32	26	21	
top	Acne	vomiting	vomiting	fatigue	high fever	headache	nausea	abdominal pain	abdominal pain	yello
freq	120	822	870	726	378	348	390	264	276	

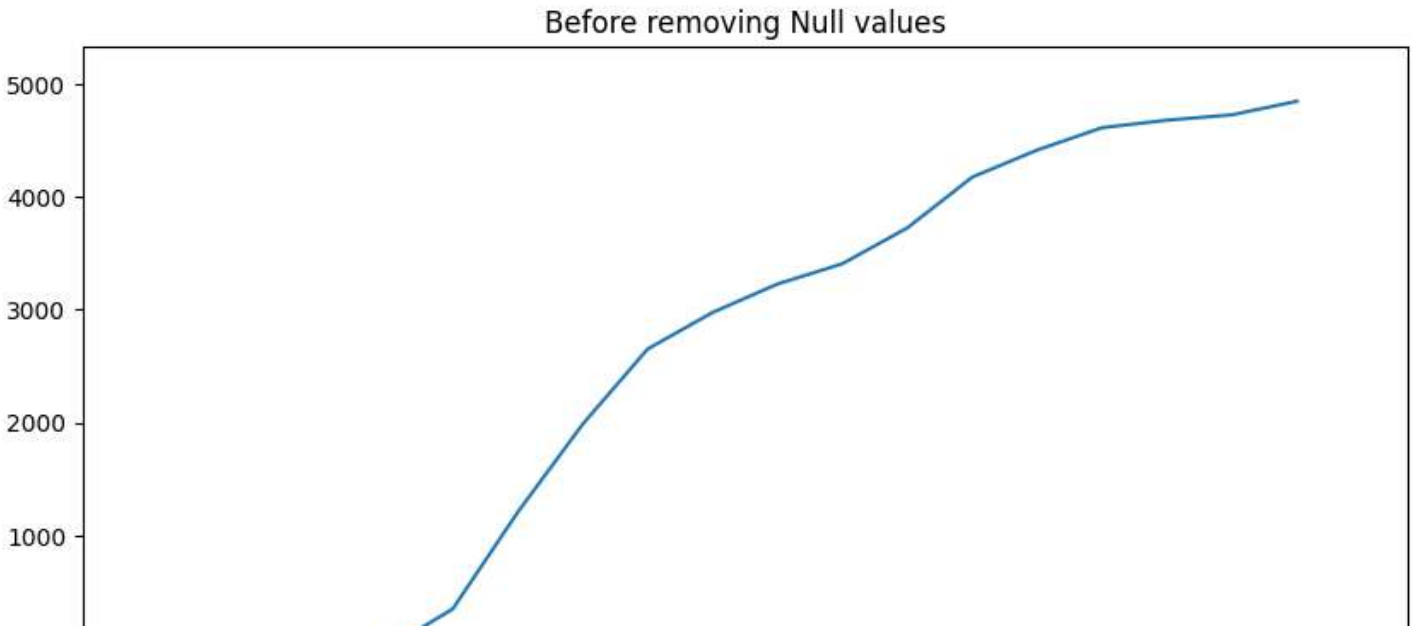
In [5]:

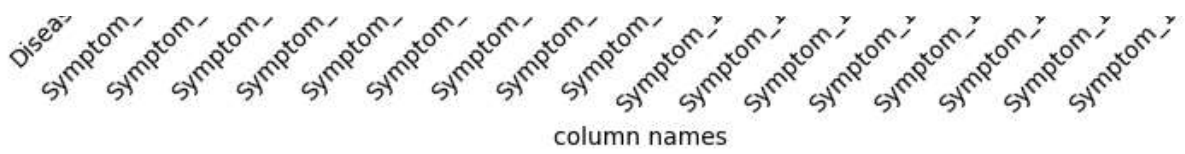
```
null_checker = df.apply(lambda x: sum(x.isnull())).to_frame(name='count')
print(null_checker)
```

```
            count
Disease         0
Symptom_1       0
Symptom_2       0
Symptom_3       0
Symptom_4     348
Symptom_5    1205
Symptom_6    1985
Symptom_7    2651
Symptom_8    2975
Symptom_9    3227
Symptom_10   3407
Symptom_11   3725
Symptom_12   4175
Symptom_13   4415
Symptom_14   4613
Symptom_15   4679
Symptom_16   4727
Symptom_17   4847
```

In [6]:

```
plt.figure(figsize=(10,5))
plt.plot(null_checker.index, null_checker['count'])
plt.xticks(null_checker.index, null_checker.index, rotation=45,
horizontalalignment='right')
plt.title('Before removing Null values')
plt.xlabel('column names')
plt.margins(0.1)
plt.show()
```





In [7]:

```
cols = df.columns
data = df[cols].values.flatten()

s = pd.Series(data)
s = s.str.strip()
s = s.values.reshape(df.shape)

df = pd.DataFrame(s, columns=df.columns)
df.head()
```

Out[7]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Sympt
0	Acne	skin rash	pus filled pimples	scurring	NaN	NaN	NaN	NaN	NaN	
1	Acne	skin rash	pus filled pimples	blackheads	scurring	NaN	NaN	NaN	NaN	
2	Impetigo	skin rash	high fever	blister	red sore around nose	yellow crust ooze	NaN	NaN	NaN	
3	Heart attack	vomiting	breathlessness	sweating	chest pain	NaN	NaN	NaN	NaN	
4	hepatitis A	joint pain	vomiting	yellowish skin	dark urine	nausea	loss of appetite	abdominal pain	diarrhoea	mild

In [8]:

```
df = df.fillna(0)
df.head()
```

Out[8]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Sympt
0	Acne	skin rash	pus filled pimples	scurring	0	0	0	0	0	
1	Acne	skin rash	pus filled pimples	blackheads	scurring	0	0	0	0	
2	Impetigo	skin rash	high fever	blister	red sore around nose	yellow crust ooze	0	0	0	
3	Heart attack	vomiting	breathlessness	sweating	chest pain	0	0	0	0	
4	hepatitis A	joint pain	vomiting	yellowish skin	dark urine	nausea	loss of appetite	abdominal pain	diarrhoea	mild

In [9]:

```
df1 = pd.read_csv('Symptom-severity.csv')
df1['Symptom'] = df1['Symptom'].str.replace('_',' ')
df1.head()
```

	Symptom	weight
0	itching	1
1	skin rash	3
2	nodal skin eruptions	4
3	continuous sneezing	4
4	shivering	5

In [10]:

```
df1['Symptom'].unique()
```

Out[10]:

```
array(['itching', 'skin rash', 'nodal skin eruptions',
       'continuous sneezing', 'shivering', 'chills', 'joint pain',
       'stomach pain', 'acidity', 'ulcers on tongue', 'muscle wasting',
       'vomiting', 'burning micturition', 'spotting urination', 'fatigue',
       'weight gain', 'anxiety', 'cold hands and feets', 'mood swings',
       'weight loss', 'restlessness', 'lethargy', 'patches in throat',
       'irregular sugar level', 'cough', 'high fever', 'sunken eyes',
       'breathlessness', 'sweating', 'dehydration', 'indigestion',
       'headache', 'yellowish skin', 'dark urine', 'nausea',
       'loss of appetite', 'pain behind the eyes', 'back pain',
       'constipation', 'abdominal pain', 'diarrhoea', 'mild fever',
       'yellow urine', 'yellowing of eyes', 'acute liver failure',
       'fluid overload', 'swelling of stomach', 'swelled lymph nodes',
       'malaise', 'blurred and distorted vision', 'phlegm',
       'throat irritation', 'redness of eyes', 'sinus pressure',
       'runny nose', 'congestion', 'chest pain', 'weakness in limbs',
       'fast heart rate', 'pain during bowel movements',
       'pain in anal region', 'bloody stool', 'irritation in anus',
       'neck pain', 'dizziness', 'cramps', 'bruising', 'obesity',
       'swollen legs', 'swollen blood vessels', 'puffy face and eyes',
       'enlarged thyroid', 'brittle nails', 'swollen extremeties',
       'excessive hunger', 'extra marital contacts',
       'drying and tingling lips', 'slurred speech', 'knee pain',
       'hip joint pain', 'muscle weakness', 'stiff neck',
       'swelling joints', 'movement stiffness', 'spinning movements',
       'loss of balance', 'unsteadiness', 'weakness of one body side',
       'loss of smell', 'bladder discomfort', 'foul smell ofurine',
       'continuous feel of urine', 'passage of gases', 'internal itching',
       'toxic look (typhos)', 'depression', 'irritability', 'muscle pain',
       'altered sensorium', 'red spots over body', 'belly pain',
       'abnormal menstruation', 'dischromic patches',
       'watering from eyes', 'increased appetite', 'polyuria',
       'family history', 'mucoid sputum', 'rusty sputum',
       'lack of concentration', 'visual disturbances',
       'receiving blood transfusion', 'receiving unsterile injections',
       'coma', 'stomach bleeding', 'distention of abdomen',
       'history of alcohol consumption', 'blood in sputum',
       'prominent veins on calf', 'palpitations', 'painful walking',
       'pus filled pimples', 'blackheads', 'scurring', 'skin peeling',
       'silver like dusting', 'small dents in nails',
       'inflammatory nails', 'blister', 'red sore around nose',
       'yellow crust ooze', 'prognosis'], dtype=object)
```

In [11]:

```
vals = df.values
symptoms = df1['Symptom'].unique()

for i in range(len(symptoms)):
    vals[vals == symptoms[i]] = df1[df1['Symptom'] == symptoms[i]]['weight'].values[0]

d = pd.DataFrame(vals, columns=cols)
d.head(20)
```

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	S
0	Acne	3	2	2	0	0	0	0	0	
1	Acne	3	2	2	2	0	0	0	0	
2	Impetigo	3	7	4	2	3	0	0	0	
3	Heart attack	5	4	3	7	0	0	0	0	
4	hepatitis A	3	5	3	4	5	4	4	6	
5	Hypertension	7	4	4	3	0	0	0	0	
6	Gastroenteritis	5	3	6	0	0	0	0	0	
7	Arthritis	2	4	5	5	2	0	0	0	
8	Peptic ulcer diseae	5	5	4	4	5	4	0	0	
9	Migraine	3	5	3	5	4	4	3	2	
10	Osteoarthristis	3	5	3	2	5	2	0	0	
11	Hepatitis D	3	5	4	3	4	5	4	4	
12	Heart attack	5	3	7	0	0	0	0	0	
13	Hepatitis B	1	4	2	3	4	4	4	4	
14	Hepatitis E	3	5	4	7	3	4	5	4	
15	Varicose veins	4	4	4	5	5	6	0	0	
16	Hypothyroidism	4	3	5	3	2	4	5	6	
17	Heart attack	4	3	7	0	0	0	0	0	
18	AIDS	3	6	7	5	0	0	0	0	
19	Hepatitis E	3	5	4	7	4	5	4	4	

In [12]:

```
d = d.replace('dischromic  patches', 0)
d = d.replace('spotting  urination',0)
df = d.replace('foul smell of urine',0)
df.head(20)
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_10828\1341861434.py:1: FutureWarning: Downcast
ing behavior in `replace` is deprecated and will be removed in a future version. To retai
n the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the
future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  d = d.replace('dischromic  patches', 0)
C:\Users\Asus\AppData\Local\Temp\ipykernel_10828\1341861434.py:2: FutureWarning: Downcast
ing behavior in `replace` is deprecated and will be removed in a future version. To retai
n the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the
future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  d = d.replace('spotting  urination',0)
C:\Users\Asus\AppData\Local\Temp\ipykernel_10828\1341861434.py:3: FutureWarning: Downcast
ing behavior in `replace` is deprecated and will be removed in a future version. To retai
n the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the
future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df = d.replace('foul smell of urine',0)
```

Out[12]:

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	S
0	Acne	3	2	2	0	0	0	0	0	
1	Acne	3	2	2	2	0	0	0	0	
2	Impetigo	3	7	4	2	3	0	0	0	
3	Heart attack	5	4	3	7	0	0	0	0	

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	S
6	Gastroenteritis	5	3	6	0	0	0	0	0	
7	Arthritis	2	4	5	5	2	0	0	0	
8	Peptic ulcer diseae	5	5	4	4	5	4	0	0	
9	Migraine	3	5	3	5	4	4	3	2	
10	Osteoarthristis	3	5	3	2	5	2	0	0	
11	Hepatitis D	3	5	4	3	4	5	4	4	
12	Heart attack	5	3	7	0	0	0	0	0	
13	Hepatitis B	1	4	2	3	4	4	4	4	
14	Hepatitis E	3	5	4	7	3	4	5	4	
15	Varicose veins	4	4	4	5	5	6	0	0	
16	Hypothyroidism	4	3	5	3	2	4	5	6	
17	Heart attack	4	3	7	0	0	0	0	0	
18	AIDS	3	6	7	5	0	0	0	0	
19	Hepatitis E	3	5	4	7	4	5	4	4	

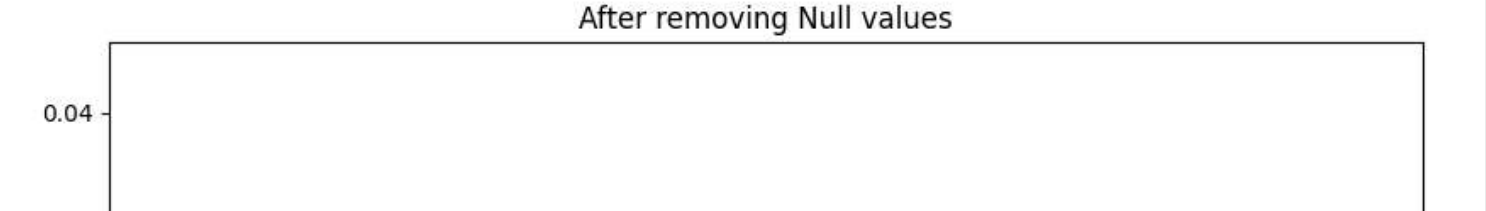
In [13]:

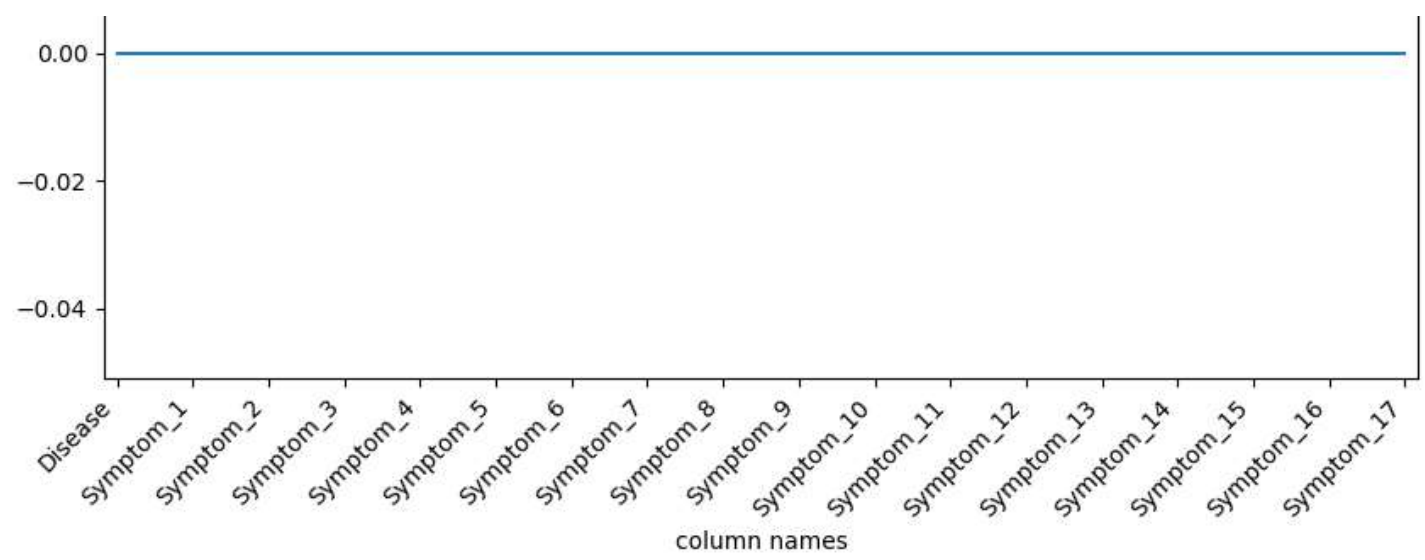
```
null_checker = df.apply(lambda x: sum(x.isnull())).to_frame(name='count')
print(null_checker)
```

```
             count
Disease          0
Symptom_1        0
Symptom_2        0
Symptom_3        0
Symptom_4        0
Symptom_5        0
Symptom_6        0
Symptom_7        0
Symptom_8        0
Symptom_9        0
Symptom_10       0
Symptom_11       0
Symptom_12       0
Symptom_13       0
Symptom_14       0
Symptom_15       0
Symptom_16       0
Symptom_17       0
```

In [14]:

```
plt.figure(figsize=(10,5))
plt.plot(null_checker.index, null_checker['count'])
plt.xticks(null_checker.index, null_checker.index, rotation=45,
horizontalalignment='right')
plt.title('After removing Null values')
plt.xlabel('column names')
plt.margins(0.01)
plt.show()
```





In [15]:

```
print("Number of symptoms used to identify the disease ",len(df1['Symptom'].unique()))
print("Number of diseases that can be identified ",len(df['Disease'].unique()))
```

```
Number of symptoms used to identify the disease  132
Number of diseases that can be identified  41
```

In [16]:

```
data = df.iloc[:,1:].values
labels = df['Disease'].values
```

In [17]:

```
x_train, x_test, y_train, y_test = train_test_split(data, labels, train_size = 0.8,rando
m_state=42)
```

In [18]:

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(3935, 17) (984, 17) (3935,) (984,)
```

In [19]:

```
null_checker = df.apply(lambda x: sum(x.isnull())).to_frame(name='count')
print(null_checker)
```

```
            count
Disease         0
Symptom_1       0
Symptom_2       0
Symptom_3       0
Symptom_4       0
Symptom_5       0
Symptom_6       0
Symptom_7       0
Symptom_8       0
Symptom_9       0
Symptom_10      0
Symptom_11      0
Symptom_12      0
Symptom_13      0
Symptom_14      0
Symptom_15      0
Symptom_16      0
Symptom_17      0
```

```
ann_model.add(tf.keras.Input(shape=(x_train.shape[1],)))
ann_model.add(Dense(units=100, activation='relu',kernel_regularizer=regularizers.l2(0.000
1)))
ann_model.add(Dense(units=50, activation='relu',kernel_regularizer=regularizers.l2(0.000
1)))
ann_model.add(Dense(units=len(df['Disease'].unique()), activation='softmax',kernel_regula
rizer=regularizers.l2(0.0001)))
```

In [21]:

```
ann_model.compile(tf.keras.optimizers.Adam(learning_rate=0.001), loss='sparse_categorical
_crossentropy', metrics=['accuracy'])
```

In [22]:

```
label_encoder = LabelEncoder()

# Encode categorical variables
for col in range(x_train.shape[1]):
    if isinstance(x_train[0, col], str):
        x_train[:, col] = label_encoder.fit_transform(x_train[:, col])
        x_test[:, col] = label_encoder.transform(x_test[:, col])
```

In [23]:

```
label_encoder.fit(df['Disease'].unique())
y_train = label_encoder.transform(y_train)
y_test = label_encoder.transform(y_test)
```

In [24]:

```
ann_model.fit(x_train, y_train, epochs=50, batch_size=32, validation_split=0.2, verbose=
1)
```

```
Epoch 1/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1265 - loss: 3.4498 - val_accuracy:
0.4028 - val_loss: 2.2993
Epoch 2/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.5310 - loss: 1.8972 - val_accuracy:
0.6277 - val_loss: 1.3842
Epoch 3/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7229 - loss: 1.1403 - val_accuracy:
0.7992 - val_loss: 0.9646
Epoch 4/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8137 - loss: 0.8055 - val_accuracy:
0.8577 - val_loss: 0.7543
Epoch 5/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8522 - loss: 0.6324 - val_accuracy:
0.8653 - val_loss: 0.6297
Epoch 6/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 960us/step - accuracy: 0.8866 - loss: 0.5048 - val_accuracy
: 0.8742 - val_loss: 0.5403
Epoch 7/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8939 - loss: 0.4563 - val_accuracy:
0.8945 - val_loss: 0.4962
Epoch 8/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 991us/step - accuracy: 0.9100 - loss: 0.3780 - val_accuracy
: 0.9085 - val_loss: 0.4159
Epoch 9/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9247 - loss: 0.3360 - val_accuracy:
0.9136 - val_loss: 0.3831
Epoch 10/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 995us/step - accuracy: 0.9301 - loss: 0.3098 - val_accuracy
: 0.8996 - val_loss: 0.3552
Epoch 11/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9322 - loss: 0.2870 - val_accuracy:
0.9263 - val_loss: 0.3059
Epoch 12/50
```

```
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9431 - loss: 0.2447 - val_accuracy:
0.9352 - val_loss: 0.2596
Epoch 14/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9451 - loss: 0.2314 - val_accuracy:
0.9466 - val_loss: 0.2327
Epoch 15/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9501 - loss: 0.2034 - val_accuracy:
0.9479 - val_loss: 0.2400
Epoch 16/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 966us/step - accuracy: 0.9568 - loss: 0.1883 - val_accuracy
: 0.9543 - val_loss: 0.2138
Epoch 17/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 981us/step - accuracy: 0.9651 - loss: 0.1704 - val_accuracy
: 0.9593 - val_loss: 0.1951
Epoch 18/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9604 - loss: 0.1843 - val_accuracy:
0.9479 - val_loss: 0.1906
Epoch 19/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 990us/step - accuracy: 0.9684 - loss: 0.1603 - val_accuracy
: 0.9708 - val_loss: 0.1730
Epoch 20/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 980us/step - accuracy: 0.9658 - loss: 0.1618 - val_accuracy
: 0.9797 - val_loss: 0.1545
Epoch 21/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 985us/step - accuracy: 0.9681 - loss: 0.1539 - val_accuracy
: 0.9657 - val_loss: 0.1664
Epoch 22/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9789 - loss: 0.1307 - val_accuracy:
0.9809 - val_loss: 0.1601
Epoch 23/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9796 - loss: 0.1420 - val_accuracy:
0.9835 - val_loss: 0.1396
Epoch 24/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9829 - loss: 0.1325 - val_accuracy:
0.9809 - val_loss: 0.1437
Epoch 25/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9786 - loss: 0.1260 - val_accuracy:
0.9720 - val_loss: 0.1421
Epoch 26/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9835 - loss: 0.1170 - val_accuracy:
0.9797 - val_loss: 0.1402
Epoch 27/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9850 - loss: 0.1249 - val_accuracy:
0.9695 - val_loss: 0.1348
Epoch 28/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9775 - loss: 0.1299 - val_accuracy:
0.9809 - val_loss: 0.1291
Epoch 29/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9832 - loss: 0.1100 - val_accuracy:
0.9873 - val_loss: 0.1167
Epoch 30/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9867 - loss: 0.1051 - val_accuracy:
0.9860 - val_loss: 0.1172
Epoch 31/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9769 - loss: 0.1241 - val_accuracy:
0.9848 - val_loss: 0.1172
Epoch 32/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9829 - loss: 0.1163 - val_accuracy:
0.9848 - val_loss: 0.1112
Epoch 33/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9828 - loss: 0.1041 - val_accuracy:
0.9759 - val_loss: 0.1212
Epoch 34/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9843 - loss: 0.1052 - val_accuracy:
0.9873 - val_loss: 0.0997
Epoch 35/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9823 - loss: 0.1052 - val_accuracy:
0.9848 - val_loss: 0.1060
Epoch 36/50
```

```
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9891 - loss: 0.0889 - val_accuracy:
0.9809 - val_loss: 0.1050
Epoch 38/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9844 - loss: 0.0970 - val_accuracy:
0.9835 - val_loss: 0.0968
Epoch 39/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9817 - loss: 0.1065 - val_accuracy:
0.9822 - val_loss: 0.0950
Epoch 40/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 985us/step - accuracy: 0.9838 - loss: 0.1005 - val_accuracy
: 0.9860 - val_loss: 0.0972
Epoch 41/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9886 - loss: 0.0885 - val_accuracy:
0.9873 - val_loss: 0.0882
Epoch 42/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9826 - loss: 0.0979 - val_accuracy:
0.9848 - val_loss: 0.0989
Epoch 43/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9844 - loss: 0.1032 - val_accuracy:
0.9733 - val_loss: 0.1039
Epoch 44/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9865 - loss: 0.0939 - val_accuracy:
0.9898 - val_loss: 0.0926
Epoch 45/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9839 - loss: 0.0917 - val_accuracy:
0.9860 - val_loss: 0.0881
Epoch 46/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9832 - loss: 0.0957 - val_accuracy:
0.9860 - val_loss: 0.0864
Epoch 47/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9845 - loss: 0.0920 - val_accuracy:
0.9886 - val_loss: 0.0855
Epoch 48/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9856 - loss: 0.0932 - val_accuracy:
0.9898 - val_loss: 0.0810
Epoch 49/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9899 - loss: 0.0870 - val_accuracy:
0.9911 - val_loss: 0.0815
Epoch 50/50
99/99 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9849 - loss: 0.0879 - val_accuracy:
0.9848 - val_loss: 0.0924
```

Out[24]:

```
<keras.src.callbacks.history.History at 0x12161933a70>
```

In [25]:

```
pred_probs = ann_model.predict(x_test)
preds = np.argmax(pred_probs, axis=1)
```

```
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```

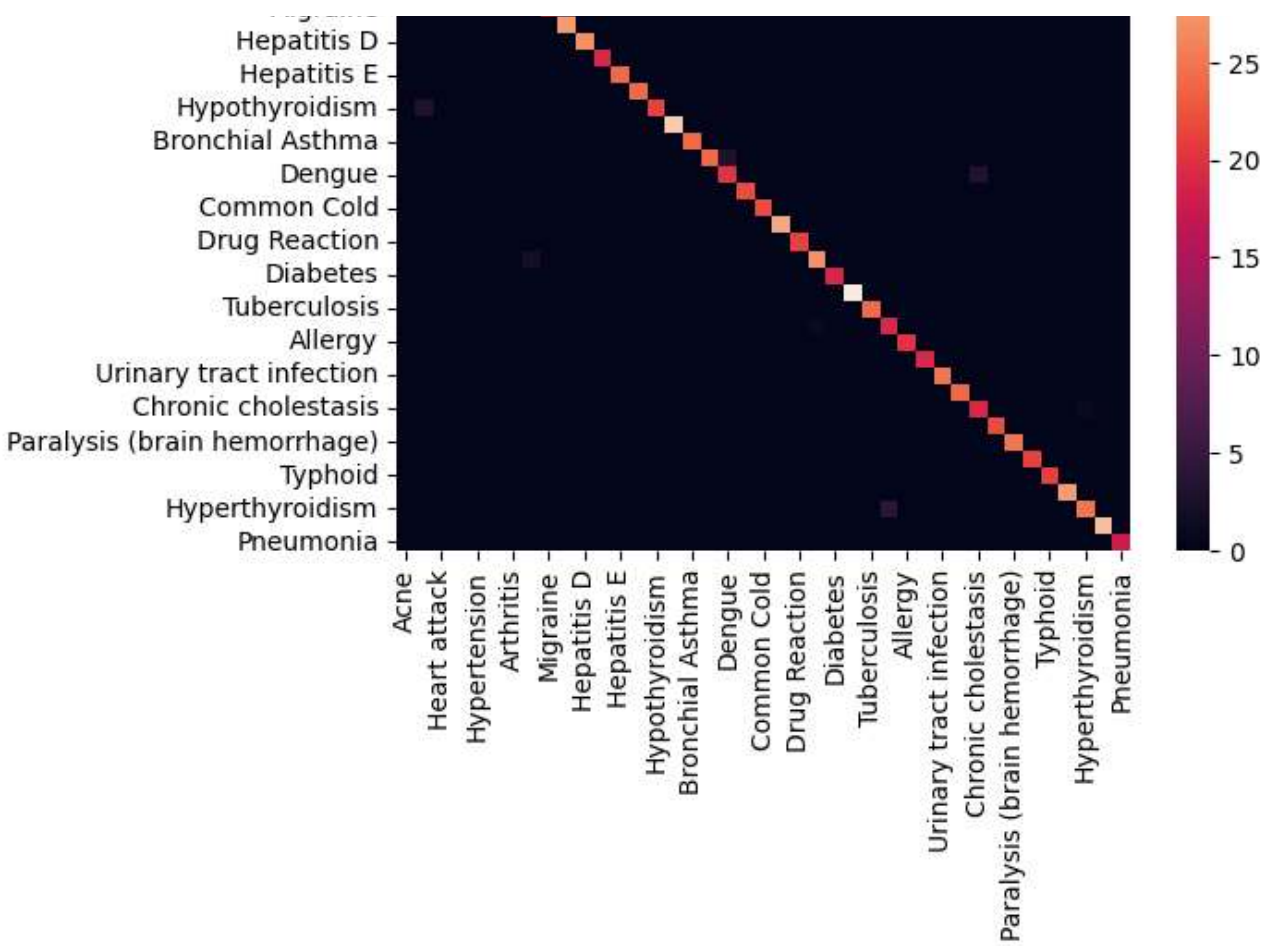
In [26]:

```
conf_mat = confusion_matrix(y_test, preds)
df_cm = pd.DataFrame(conf_mat, index=df['Disease'].unique(), columns=df['Disease'].uniqu
e())
print('F1-score% =', f1_score(y_test, preds, average='macro') * 100, '|', 'Accuracy% =',
accuracy_score(y_test, preds) * 100)
sns.heatmap(df_cm)
```

```
F1-score% = 98.04617591539801 | Accuracy% = 98.06910569105692
```

Out[26]:

```
<Axes: >
```



In [27]:

```
def predict_single(symptoms_list):
    # Create a DataFrame from the input symptoms
    input_data = pd.DataFrame([symptoms_list], columns=symptoms_list)

    # Convert input symptoms to weights using the symptom severity CSV
    encoded_input = []
    for symptom in symptoms_list:
        if symptom in df1['Symptom'].values:
            encoded_input.append(df1[df1['Symptom'] == symptom]['weight'].values[0])
        else:
            encoded_input.append(0)  # If symptom is not found, treat it as zero

    # Convert the list to a numpy array (similar to the input data shape)
    input_array = np.array(encoded_input).reshape(1, -1)

    # Pad the input array to match the number of columns in the training set (if needed)
    if input_array.shape[1] < x_train.shape[1]:
        padded_input = np.zeros((1, x_train.shape[1]))
        padded_input[:, :input_array.shape[1]] = input_array
    else:
        padded_input = input_array

    # Predict the disease
    prediction_probs = ann_model.predict(padded_input)
    predicted_class = np.argmax(prediction_probs, axis=1)
    predicted_disease = label_encoder.inverse_transform(predicted_class)[0]

    return predicted_disease
```

In [28]:

```
symptoms = ['stomach pain', 'acidity', 'ulcers on tongue', 'vomiting', 'cough', 'chest pa
in']
```

In [30]:

```
symptomss = ['vomiting', 'yellowish skin', 'abdominal pain', 'swelling of stomach', 'dist
ention of abdomen', 'history of alcohol consumption', 'fluid overload']
```

In [31]:

```
predicted_disease = predict_single(symptomss)
print(f"Predicted Disease: {predicted_disease}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
Predicted Disease: Alcoholic hepatitis
```

In [32]:

```
def gradio_predict(*symptoms):
    symptom_list = list(symptoms)
    return predict_single(symptom_list)
```

In [ ]:

```
import gradio as gr
inputs = [gr.Textbox(placeholder="Symptom", label=f"Symptom {i+1}") for i in range(17)]

interface = gr.Interface(
    fn=gradio_predict,
    inputs=inputs,
    outputs="text",
    title="Disease Prediction Based on Symptoms",
    description="Enter symptoms to predict the most likely disease. Each input field can
accept one symptom."
)

# Launch the Gradio interface
interface.launch()
```