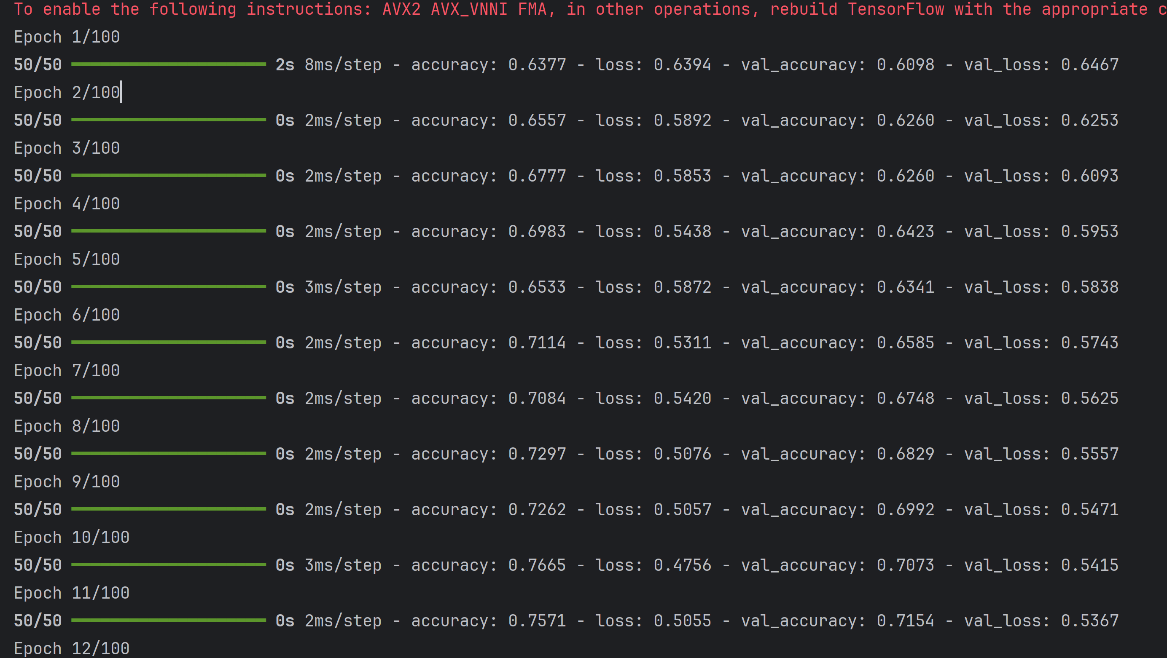
**PRACTICAL 11**

# AIM: Implementation of Neural network-based application.

**INPUT:**

# Importing necessary libraries  
import tensorflow as tf  
from tensorflow.keras import layers, models  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.metrics import accuracy\_score  
import pandas as pd  
  
# Load the Pima Indians Diabetes dataset from a CSV file  
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"  
column\_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']  
data = pd.read\_csv(url, names=column\_names)  
  
# Split data into input features (X) and output labels (y)  
X = data.iloc[:, 0:8].values # First 8 columns are features  
y = data.iloc[:, 8].values # The 9th column is the label (Outcome)  
  
# Split into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalize the feature data (important for neural networks)  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Building the neural network model  
model = models.Sequential()  
  
# Input layer and one hidden layer with 12 neurons and ReLU activation  
model.add(layers.Dense(12, input\_dim=8, activation='relu'))  
  
# Second hidden layer with 8 neurons  
model.add(layers.Dense(8, activation='relu'))  
  
# Output layer with 1 neuron (for binary classification) and sigmoid activation  
model.add(layers.Dense(1, activation='sigmoid'))  
  
# Compile the model  
model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
  
# Train the model  
history = model.fit(X\_train, y\_train, epochs=100, batch\_size=10, validation\_split=0.2, verbose=1)  
  
# Evaluate the model on the test set  
\_, test\_accuracy = model.evaluate(X\_test, y\_test)  
print(f'Test accuracy: {test\_accuracy:.4f}')  
  
# Make predictions  
y\_pred = (model.predict(X\_test) > 0.5).astype("int32")  
  
# Print accuracy score  
print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

**OUTPUT:**