NNDL LAB

LAB 6

Name: Kushal Sourav B

Regno: 2347125

In [21]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

data = pd.read_csv('./HistoricalQuotes.csv')
data = data[[' Close/Last']]

data
```

Out[21]:

	Close/Last
0	$273.36
1	$273.52
2	$292.65
3	$288.08
4	$298.18
...	...
2513	$31.2786
2514	$30.1014
2515	$29.9043
2516	$29.8357
2517	$29.8557

2518 rows × 1 columns

- cleaned the Close/Last column by removing dollar signs and commas, converting it to a numeric format.
- applied Min-Max Scaling to normalize the Close prices between 0 and 1 for better model performance.
- extracted the normalized Close prices into a NumPy array for easy input into a model.

In [25]:
```
data['Close'] = data[' Close/Last'].replace('[\$,]', '', regex=True).astype(floa

scaler = MinMaxScaler(feature_range=(0, 1))
data['Close'] = scaler.fit_transform(data[['Close']])
```

```
close_prices = data['Close'].values
close_prices
```

Out[25]:
```
array([8.18942624e-01, 8.19480684e-01, 8.83812549e-01, ...,
       2.30693463e-04, 0.00000000e+00, 6.72575693e-05])
```

- split the data into training and testing sets, using 80% of the close_prices for training and the remaining 20% for testing.
- The training data consists of the first 80% of the close_prices, and the testing data includes the remaining 20%.
- assigned the first 80% of the close_prices to train_data and the rest to test_data for model evaluation.

In [26]:
```
train_size = int(len(close_prices) * 0.8)
train_data = close_prices[:train_size]
test_data = close_prices[train_size:]


train_size, train_data, test_data
```

Out[26]:
```
(2014,
 array([0.81894262, 0.81948068, 0.88381255, ..., 0.16123556, 0.16026033,
        0.15688299]),
 array([1.52246924e-01, 1.50637787e-01, 1.47745711e-01, 1.46136238e-01,
        1.47005878e-01, 1.40890147e-01, 1.40933528e-01, 1.38752365e-01,
        1.44416462e-01, 1.41120841e-01, 1.36710762e-01, 1.36590371e-01,
        1.28668438e-01, 1.24897306e-01, 1.22562460e-01, 1.20501688e-01,
        1.18310772e-01, 1.18824956e-01, 1.18964180e-01, 1.17297201e-01,
        1.14544348e-01, 1.13271499e-01, 1.14246734e-01, 1.01635939e-01,
        1.04998145e-01, 1.01582806e-01, 1.05161918e-01, 1.05815325e-01,
        1.03696711e-01, 1.01347405e-01, 1.02106742e-01, 1.02663971e-01,
        1.02995215e-01, 1.02269842e-01, 1.02591670e-01, 1.00492561e-01,
        9.82871851e-02, 9.72255244e-02, 9.42325626e-02, 9.42904041e-02,
        9.30989362e-02, 9.49676878e-02, 9.34305160e-02, 9.11340063e-02,
        9.01227888e-02, 8.98850333e-02, 8.32840391e-02, 8.27123498e-02,
        8.17132386e-02, 8.23135124e-02, 8.64548972e-02, 8.79103510e-02,
        8.87655310e-02, 8.73437060e-02, 8.65894124e-02, 8.74829292e-02,
        8.84726243e-02, 8.68823191e-02, 8.60321834e-02, 8.32793311e-02,
        7.89556783e-02, 8.03583349e-02, 7.43293664e-02, 7.59721325e-02,
        8.05456472e-02, 7.69426592e-02, 7.97913536e-02, 8.09781134e-02,
        8.45138438e-02, 8.64643133e-02, 8.18669222e-02, 8.44418782e-02,
        8.47300769e-02, 8.95632058e-02, 9.48234203e-02, 9.17009876e-02,
        9.19458052e-02, 9.33054170e-02, 9.05861934e-02, 9.01540636e-02,
        9.41269682e-02, 9.42086861e-02, 9.40835870e-02, 9.21189934e-02,
        9.07593817e-02, 9.46024792e-02, 8.84053668e-02, 8.95773299e-02,
        9.11676351e-02, 1.02514996e-01, 1.01434167e-01, 1.02399649e-01,
        9.58803730e-02, 9.28827031e-02, 9.19700179e-02, 8.64548972e-02,
        7.73223282e-02, 8.09589450e-02, 8.13816588e-02, 7.86193904e-02,
        7.96282540e-02, 8.28566173e-02, 8.73003249e-02, 9.03941731e-02,
        9.14750022e-02, 9.33535061e-02, 9.38962747e-02, 9.27048069e-02,
        9.76626986e-02, 9.82922294e-02, 9.74178810e-02, 9.20709043e-02,
        8.84484116e-02, 8.66902987e-02, 8.44418782e-02, 8.21934577e-02,
        8.10117422e-02, 8.42111847e-02, 8.41102984e-02, 8.20976156e-02,
        7.93639317e-02, 8.27170578e-02, 8.45427645e-02, 8.70215423e-02,
        8.70121262e-02, 8.39421545e-02, 7.92055401e-02, 8.03872556e-02,
        7.91476986e-02, 7.09039384e-02, 7.07068737e-02, 7.55204979e-02,
        8.24339035e-02, 8.24530719e-02, 8.38604365e-02, 8.07763407e-02,
        7.91957878e-02, 7.43868716e-02, 7.93447633e-02, 6.93523062e-02,
        7.91574510e-02, 8.09589450e-02, 8.82610993e-02, 8.65029864e-02,
        9.02694103e-02, 8.72569438e-02, 8.79009350e-02, 8.82708516e-02,
        9.34688529e-02, 9.11101299e-02, 8.86118475e-02, 8.57244800e-02,
        8.55371677e-02, 8.07090831e-02, 7.92438770e-02, 7.49777293e-02,
        7.15428853e-02, 7.16629400e-02, 6.96115842e-02, 6.97316389e-02,
        7.24747389e-02, 7.12691470e-02, 6.86555178e-02, 6.75363519e-02,
        6.45719745e-02, 6.09256054e-02, 6.01427273e-02, 6.07288770e-02,
        5.91819529e-02, 5.64482690e-02, 5.87928679e-02, 5.46514830e-02,
        5.59438372e-02, 5.11493814e-02, 5.35225647e-02, 5.58765797e-02,
        5.66406257e-02, 5.93739733e-02, 5.65683238e-02, 5.62320359e-02,
        5.89176307e-02, 5.92777949e-02, 5.91819529e-02, 6.20642760e-02,
        6.46584005e-02, 6.59373032e-02, 6.56531399e-02, 6.67675979e-02,
        6.17616170e-02, 6.06037779e-02, 6.14589579e-02, 5.92539185e-02,
        6.03155792e-02, 6.07097086e-02, 6.32604519e-02, 6.29433325e-02,
        6.11515908e-02, 5.97872710e-02, 6.32459915e-02, 6.61622797e-02,
        6.64793992e-02, 6.75457679e-02, 6.66569592e-02, 6.62053246e-02,
        6.62487057e-02, 6.76036094e-02, 6.69451578e-02, 6.60230566e-02,
        6.78726397e-02, 6.62487057e-02, 6.78820558e-02, 6.80118629e-02,
        6.92554553e-02, 6.81443603e-02, 6.41637211e-02, 6.19778501e-02,
        5.90904826e-02, 5.69816215e-02, 5.93645572e-02, 6.11468828e-02,
        5.93548049e-02, 5.85860508e-02, 6.06326987e-02, 6.20834444e-02,
        6.20642760e-02, 6.24725295e-02, 6.35775713e-02, 6.51964610e-02,
```

```
6.70931245e-02, 6.71519749e-02, 6.82711408e-02, 6.80212789e-02,
6.85499234e-02, 6.53935257e-02, 6.26167970e-02, 6.35822794e-02,
6.26695942e-02, 5.85238376e-02, 6.04309260e-02, 5.82067182e-02,
6.56144668e-02, 6.95204502e-02, 6.87661565e-02, 6.62103689e-02,
6.89965137e-02, 7.05770666e-02, 7.03850462e-02, 7.26142984e-02,
7.24027733e-02, 6.88283698e-02, 6.74785104e-02, 6.93523062e-02,
6.69259894e-02, 6.43893702e-02, 6.42646074e-02, 6.23380143e-02,
6.80791205e-02, 7.17974552e-02, 7.41178413e-02, 7.25662092e-02,
7.22201690e-02, 7.11010030e-02, 6.99912532e-02, 7.17301976e-02,
7.03080363e-02, 6.87133593e-02, 6.61286510e-02, 6.46584005e-02,
6.50814506e-02, 6.54224465e-02, 6.26793465e-02, 6.11324224e-02,
6.45480981e-02, 6.48554652e-02, 6.36784577e-02, 6.17807854e-02,
5.66261653e-02, 5.94893200e-02, 6.24486530e-02, 6.33182934e-02,
6.70796730e-02, 6.57345216e-02, 6.51292035e-02, 6.37938044e-02,
6.41852435e-02, 6.11418385e-02, 5.99937518e-02, 6.01235589e-02,
5.88214523e-02, 5.79951931e-02, 5.46276066e-02, 5.51559148e-02,
5.59391292e-02, 5.60255552e-02, 5.56458862e-02, 5.51273303e-02,
5.58765797e-02, 5.54178830e-02, 5.44594627e-02, 5.36907087e-02,
5.39980758e-02, 5.35706539e-02, 5.35370251e-02, 5.41998485e-02,
5.36668322e-02, 5.32811101e-02, 5.38830653e-02, 5.25379139e-02,
5.34697676e-02, 5.21679973e-02, 5.25089932e-02, 5.16682736e-02,
4.91461147e-02, 5.18939227e-02, 5.09956979e-02, 5.08971655e-02,
4.79835676e-02, 5.02077754e-02, 4.70227933e-02, 4.78393001e-02,
4.40298314e-02, 4.45534316e-02, 4.71690785e-02, 4.76472798e-02,
5.17906823e-02, 5.24511517e-02, 5.15145900e-02, 5.27346423e-02,
5.20190218e-02, 5.25664984e-02, 4.99387452e-02, 4.82862267e-02,
4.57976966e-02, 4.42601886e-02, 4.63068364e-02, 4.75511015e-02,
4.76566958e-02, 4.80363648e-02, 4.73782495e-02, 4.83629003e-02,
4.88481637e-02, 4.83484399e-02, 5.24370276e-02, 5.08705988e-02,
4.48991355e-02, 4.38566432e-02, 4.30882255e-02, 4.15604698e-02,
4.09406913e-02, 3.86105528e-02, 3.85960924e-02, 3.84760376e-02,
3.35278983e-02, 3.53919418e-02, 3.59827995e-02, 3.77217440e-02,
3.74769265e-02, 3.95444241e-02, 4.00999717e-02, 3.84666216e-02,
3.79043483e-02, 3.59925519e-02, 3.57329377e-02, 3.19570977e-02,
3.25334951e-02, 2.94826918e-02, 2.84452438e-02, 2.79552724e-02,
2.62112836e-02, 2.60478477e-02, 2.59758821e-02, 2.35209808e-02,
2.39820315e-02, 2.08115097e-02, 1.99274089e-02, 1.64542280e-02,
1.61656931e-02, 1.57429792e-02, 1.50993243e-02, 1.63533417e-02,
1.49311804e-02, 1.77512903e-02, 1.95958291e-02, 1.97111758e-02,
2.12436395e-02, 2.07153313e-02, 1.86350547e-02, 1.93365512e-02,
2.06289054e-02, 1.98601513e-02, 2.42897348e-02, 2.54136088e-02,
2.46166066e-02, 2.53897324e-02, 2.60048029e-02, 2.55003711e-02,
2.54616980e-02, 2.32519506e-02, 2.36649120e-02, 2.50342761e-02,
2.65331111e-02, 2.42271853e-02, 2.45443047e-02, 2.41041040e-02,
2.18059128e-02, 2.06769945e-02, 1.76453596e-02, 1.97209282e-02,
2.04654695e-02, 2.10788585e-02, 2.06326045e-02, 2.32687649e-02,
2.43906212e-02, 2.36554960e-02, 2.39315883e-02, 1.91109020e-02,
1.82987669e-02, 1.90386001e-02, 2.05038063e-02, 2.27330584e-02,
2.85605905e-02, 2.77918365e-02, 2.88968783e-02, 2.98431923e-02,
3.12266805e-02, 2.94588153e-02, 3.13342927e-02, 3.02756585e-02,
2.80561587e-02, 2.44242500e-02, 2.18250812e-02, 2.14551646e-02,
2.00138349e-02, 1.65019809e-02, 1.94471899e-02, 2.02206519e-02,
2.26345261e-02, 2.60720604e-02, 2.64705615e-02, 2.49717266e-02,
2.30740543e-02, 2.13781547e-02, 1.69388188e-02, 1.74725076e-02,
1.82123409e-02, 1.60792671e-02, 1.38886881e-02, 1.89713426e-02,
2.09026437e-02, 2.17961605e-02, 2.16041401e-02, 2.37853031e-02,
2.55770447e-02, 2.29012023e-02, 2.16858581e-02, 1.29760028e-02,
1.79675233e-02, 2.26442784e-02, 2.39389866e-02, 2.76236926e-02,
2.50968257e-02, 2.87236901e-02, 2.53416432e-02, 2.55531683e-02,
2.91369879e-02, 2.97759348e-02, 2.76811978e-02, 2.41982646e-02,
```

```
1.71698486e-02, 1.83613164e-02, 1.85197080e-02, 1.92501252e-02,
1.76984931e-02, 1.61320643e-02, 1.60648067e-02, 1.58246972e-02,
1.49409327e-02, 1.52530078e-02, 1.47438680e-02, 1.42394363e-02,
1.30288000e-02, 1.25627051e-02, 1.29686045e-02, 1.13090240e-02,
1.05930672e-02, 8.55146364e-03, 9.85794193e-03, 9.37267856e-03,
7.63844214e-03, 6.43755824e-03, 7.59035298e-03, 7.33578308e-03,
7.49451094e-03, 7.20126794e-03, 8.52725092e-03, 7.99894271e-03,
7.68182327e-03, 6.80747487e-03, 4.91451059e-03, 4.85229733e-03,
8.93516807e-04, 2.30693463e-04, 0.00000000e+00, 6.72575693e-05]))
```

- defined a function create_sequences to transform the data into sequences: each sequence (x) is a window of sequence_length (default 60) previous values, with the following value as the target (y).
- applied this function to create x_train, y_train, x_test, and y_test by generating sequences from train_data and test_data, preparing the data for time-series modeling.
- Finally converted x_train, y_train, x_test, and y_test to float32 format for compatibility with neural networks.

In [27]:
```
import numpy as np

def create_sequences(data, sequence_length=60):
    x, y = [], []
    for i in range(sequence_length, len(data)):
        x.append(data[i-sequence_length:i])
        y.append(data[i])
    return np.array(x), np.array(y)

x_train, y_train = create_sequences(train_data)
x_test, y_test = create_sequences(test_data)


x_train = x_train.astype(np.float32)
y_train = y_train.astype(np.float32)
x_test = x_test.astype(np.float32)
y_test = y_test.astype(np.float32)


x_train, y_train,x_test, y_test
```

Out[27]:
```
(array([[0.8189426 , 0.81948066, 0.88381255, ..., 0.79277945, 0.779866  ,
         0.772165  ],
        [0.81948066, 0.88381255, 0.8684442 , ..., 0.779866  , 0.772165  ,
         0.78800416],
        [0.88381255, 0.8684442 , 0.90240926, ..., 0.772165  , 0.78800416,
         0.79839545],
        ...,
        [0.16979644, 0.17125224, 0.17376934, ..., 0.15440892, 0.15580216,
         0.16157655],
        [0.17125224, 0.17376934, 0.16724065, ..., 0.15580216, 0.16157655,
         0.16123556],
        [0.17376934, 0.16724065, 0.16931118, ..., 0.16157655, 0.16123556,
         0.16026032]], dtype=float32),
 array([0.78800416, 0.79839545, 0.8003795 , ..., 0.16123556, 0.16026032,
        0.15688299], dtype=float32),
 array([[0.15224692, 0.15063779, 0.14774571, ..., 0.08688232, 0.08603218,
         0.08327933],
        [0.15063779, 0.14774571, 0.14613624, ..., 0.08603218, 0.08327933,
         0.07895568],
        [0.14774571, 0.14613624, 0.14700587, ..., 0.08327933, 0.07895568,
         0.08035833],
        ...,
        [0.02137815, 0.01693882, 0.01747251, ..., 0.00491451, 0.0048523 ,
         0.00089352],
        [0.01693882, 0.01747251, 0.01821234, ..., 0.0048523 , 0.00089352,
         0.00023069],
        [0.01747251, 0.01821234, 0.01607927, ..., 0.00089352, 0.00023069,
         0.        ]], dtype=float32),
 array([7.89556801e-02, 8.03583339e-02, 7.43293688e-02, 7.59721324e-02,
        8.05456489e-02, 7.69426599e-02, 7.97913522e-02, 8.09781104e-02,
        8.45138431e-02, 8.64643157e-02, 8.18669200e-02, 8.44418779e-02,
        8.47300738e-02, 8.95632058e-02, 9.48234200e-02, 9.17009860e-02,
        9.19458047e-02, 9.33054164e-02, 9.05861929e-02, 9.01540667e-02,
        9.41269696e-02, 9.42086875e-02, 9.40835848e-02, 9.21189934e-02,
        9.07593817e-02, 9.46024805e-02, 8.84053633e-02, 8.95773321e-02,
        9.11676362e-02, 1.02514997e-01, 1.01434164e-01, 1.02399647e-01,
        9.58803743e-02, 9.28827003e-02, 9.19700190e-02, 8.64548981e-02,
        7.73223266e-02, 8.09589475e-02, 8.13816562e-02, 7.86193907e-02,
        7.96282515e-02, 8.28566179e-02, 8.73003229e-02, 9.03941765e-02,
        9.14750025e-02, 9.33535025e-02, 9.38962772e-02, 9.27048102e-02,
        9.76627022e-02, 9.82922316e-02, 9.74178836e-02, 9.20709074e-02,
        8.84484127e-02, 8.66902992e-02, 8.44418779e-02, 8.21934566e-02,
        8.10117424e-02, 8.42111856e-02, 8.41102973e-02, 8.20976123e-02,
        7.93639347e-02, 8.27170610e-02, 8.45427662e-02, 8.70215446e-02,
        8.70121270e-02, 8.39421526e-02, 7.92055428e-02, 8.03872570e-02,
        7.91476965e-02, 7.09039420e-02, 7.07068741e-02, 7.55205005e-02,
        8.24339017e-02, 8.24530721e-02, 8.38604346e-02, 8.07763413e-02,
        7.91957900e-02, 7.43868724e-02, 7.93447644e-02, 6.93523064e-02,
        7.91574493e-02, 8.09589475e-02, 8.82610977e-02, 8.65029842e-02,
        9.02694091e-02, 8.72569457e-02, 8.79009366e-02, 8.82708505e-02,
        9.34688523e-02, 9.11101326e-02, 8.86118487e-02, 8.57244805e-02,
        8.55371654e-02, 8.07090849e-02, 7.92438760e-02, 7.49777257e-02,
        7.15428889e-02, 7.16629401e-02, 6.96115866e-02, 6.97316378e-02,
        7.24747404e-02, 7.12691471e-02, 6.86555207e-02, 6.75363541e-02,
        6.45719767e-02, 6.09256066e-02, 6.01427257e-02, 6.07288778e-02,
        5.91819547e-02, 5.64482696e-02, 5.87928668e-02, 5.46514839e-02,
        5.59438355e-02, 5.11493832e-02, 5.35225645e-02, 5.58765791e-02,
        5.66406250e-02, 5.93739748e-02, 5.65683246e-02, 5.62320352e-02,
        5.89176305e-02, 5.92777953e-02, 5.91819547e-02, 6.20642751e-02,
        6.46584034e-02, 6.59373030e-02, 6.56531379e-02, 6.67675957e-02,
```

```
6.17616177e-02, 6.06037788e-02, 6.14589565e-02, 5.92539199e-02,
6.03155792e-02, 6.07097074e-02, 6.32604510e-02, 6.29433319e-02,
6.11515902e-02, 5.97872697e-02, 6.32459894e-02, 6.61622807e-02,
6.64793998e-02, 6.75457716e-02, 6.66569620e-02, 6.62053227e-02,
6.62487075e-02, 6.76036105e-02, 6.69451579e-02, 6.60230592e-02,
6.78726360e-02, 6.62487075e-02, 6.78820536e-02, 6.80118650e-02,
6.92554563e-02, 6.81443587e-02, 6.41637221e-02, 6.19778484e-02,
5.90904839e-02, 5.69816232e-02, 5.93645573e-02, 6.11468814e-02,
5.93548045e-02, 5.85860498e-02, 6.06326982e-02, 6.20834455e-02,
6.20642751e-02, 6.24725297e-02, 6.35775700e-02, 6.51964620e-02,
6.70931265e-02, 6.71519712e-02, 6.82711378e-02, 6.80212826e-02,
6.85499236e-02, 6.53935224e-02, 6.26167953e-02, 6.35822788e-02,
6.26695976e-02, 5.85238375e-02, 6.04309253e-02, 5.82067184e-02,
6.56144693e-02, 6.95204511e-02, 6.87661543e-02, 6.62103668e-02,
6.89965114e-02, 7.05770701e-02, 7.03850463e-02, 7.26142973e-02,
7.24027753e-02, 6.88283667e-02, 6.74785078e-02, 6.93523064e-02,
6.69259876e-02, 6.43893704e-02, 6.42646104e-02, 6.23380132e-02,
6.80791214e-02, 7.17974529e-02, 7.41178393e-02, 7.25662112e-02,
7.22201690e-02, 7.11010024e-02, 6.99912533e-02, 7.17301965e-02,
7.03080371e-02, 6.87133595e-02, 6.61286488e-02, 6.46584034e-02,
6.50814474e-02, 6.54224455e-02, 6.26793429e-02, 6.11324236e-02,
6.45480976e-02, 6.48554638e-02, 6.36784583e-02, 6.17807843e-02,
5.66261634e-02, 5.94893210e-02, 6.24486543e-02, 6.33182898e-02,
6.70796707e-02, 6.57345206e-02, 6.51292056e-02, 6.37938008e-02,
6.41852468e-02, 6.11418374e-02, 5.99937513e-02, 6.01235591e-02,
5.88214509e-02, 5.79951927e-02, 5.46276048e-02, 5.51559143e-02,
5.59391305e-02, 5.60255535e-02, 5.56458868e-02, 5.51273301e-02,
5.58765791e-02, 5.54178841e-02, 5.44594638e-02, 5.36907092e-02,
5.39980754e-02, 5.35706542e-02, 5.35370260e-02, 5.41998483e-02,
5.36668338e-02, 5.32811098e-02, 5.38830645e-02, 5.25379144e-02,
5.34697659e-02, 5.21679968e-02, 5.25089949e-02, 5.16682751e-02,
4.91461158e-02, 5.18939234e-02, 5.09956963e-02, 5.08971661e-02,
4.79835682e-02, 5.02077751e-02, 4.70227934e-02, 4.78392988e-02,
4.40298319e-02, 4.45534326e-02, 4.71690781e-02, 4.76472788e-02,
5.17906807e-02, 5.24511524e-02, 5.15145883e-02, 5.27346432e-02,
5.20190224e-02, 5.25664985e-02, 4.99387458e-02, 4.82862256e-02,
4.57976982e-02, 4.42601889e-02, 4.63068374e-02, 4.75511029e-02,
4.76566963e-02, 4.80363630e-02, 4.73782495e-02, 4.83628996e-02,
4.88481633e-02, 4.83484417e-02, 5.24370261e-02, 5.08705974e-02,
4.48991358e-02, 4.38566431e-02, 4.30882238e-02, 4.15604711e-02,
4.09406908e-02, 3.86105515e-02, 3.85960937e-02, 3.84760387e-02,
3.35278995e-02, 3.53919417e-02, 3.59827988e-02, 3.77217457e-02,
3.74769270e-02, 3.95444259e-02, 4.00999710e-02, 3.84666212e-02,
3.79043482e-02, 3.59925516e-02, 3.57329361e-02, 3.19570974e-02,
3.25334966e-02, 2.94826925e-02, 2.84452438e-02, 2.79552732e-02,
2.62112841e-02, 2.60478482e-02, 2.59758830e-02, 2.35209800e-02,
2.39820313e-02, 2.08115093e-02, 1.99274085e-02, 1.64542273e-02,
1.61656924e-02, 1.57429799e-02, 1.50993243e-02, 1.63533408e-02,
1.49311805e-02, 1.77512895e-02, 1.95958298e-02, 1.97111759e-02,
2.12436393e-02, 2.07153317e-02, 1.86350551e-02, 1.93365514e-02,
2.06289049e-02, 1.98601522e-02, 2.42897347e-02, 2.54136082e-02,
2.46166065e-02, 2.53897328e-02, 2.60048024e-02, 2.55003702e-02,
2.54616980e-02, 2.32519507e-02, 2.36649122e-02, 2.50342768e-02,
2.65331119e-02, 2.42271852e-02, 2.45443042e-02, 2.41041034e-02,
2.18059123e-02, 2.06769947e-02, 1.76453590e-02, 1.97209287e-02,
2.04654690e-02, 2.10788585e-02, 2.06326041e-02, 2.32687648e-02,
2.43906211e-02, 2.36554965e-02, 2.39315890e-02, 1.91109013e-02,
1.82987675e-02, 1.90386008e-02, 2.05038060e-02, 2.27330588e-02,
2.85605900e-02, 2.77918372e-02, 2.88968775e-02, 2.98431925e-02,
3.12266797e-02, 2.94588152e-02, 3.13342921e-02, 3.02756578e-02,
```

```
 2.80561596e-02, 2.44242493e-02, 2.18250807e-02, 2.14551650e-02,
 2.00138353e-02, 1.65019818e-02, 1.94471907e-02, 2.02206522e-02,
 2.26345267e-02, 2.60720607e-02, 2.64705624e-02, 2.49717273e-02,
 2.30740551e-02, 2.13781539e-02, 1.69388186e-02, 1.74725074e-02,
 1.82123408e-02, 1.60792675e-02, 1.38886878e-02, 1.89713426e-02,
 2.09026430e-02, 2.17961613e-02, 2.16041394e-02, 2.37853024e-02,
 2.55770441e-02, 2.29012016e-02, 2.16858573e-02, 1.29760029e-02,
 1.79675240e-02, 2.26442777e-02, 2.39389874e-02, 2.76236925e-02,
 2.50968263e-02, 2.87236907e-02, 2.53416430e-02, 2.55531687e-02,
 2.91369874e-02, 2.97759343e-02, 2.76811980e-02, 2.41982639e-02,
 1.71698481e-02, 1.83613170e-02, 1.85197089e-02, 1.92501247e-02,
 1.76984929e-02, 1.61320642e-02, 1.60648059e-02, 1.58246979e-02,
 1.49409324e-02, 1.52530074e-02, 1.47438683e-02, 1.42394360e-02,
 1.30288005e-02, 1.25627052e-02, 1.29686045e-02, 1.13090239e-02,
 1.05930669e-02, 8.55146348e-03, 9.85794235e-03, 9.37267859e-03,
 7.63844233e-03, 6.43755822e-03, 7.59035302e-03, 7.33578298e-03,
 7.49451108e-03, 7.20126787e-03, 8.52725096e-03, 7.99894240e-03,
 7.68182334e-03, 6.80747489e-03, 4.91451053e-03, 4.85229725e-03,
 8.93516815e-04, 2.30693462e-04, 0.00000000e+00, 6.72575698e-05],
dtype=float32))
```

- reshaped x_train and x_test to have three dimensions, where the last dimension is 1, making them suitable for LSTM and other neural network models that expect 3D input.
- The reshaping format is (samples, sequence_length, features), with 1 feature per timestep in each sequence.
- x_train and x_test are ready in 3D format, compatible with models like LSTM.

In [28]:
```
x_train = x_train.reshape((x_train.shape[0], x_train.shape[1], 1))
x_test = x_test.reshape((x_test.shape[0], x_test.shape[1], 1))

x_train, x_test
```

Out[28]:
```
(array([[[0.8189426 ],
         [0.81948066],
         [0.88381255],
         ...,
         [0.79277945],
         [0.779866  ],
         [0.772165  ]],

        [[0.81948066],
         [0.88381255],
         [0.8684442 ],
         ...,
         [0.779866  ],
         [0.772165  ],
         [0.78800416]],

        [[0.88381255],
         [0.8684442 ],
         [0.90240926],
         ...,
         [0.772165  ],
         [0.78800416],
         [0.79839545]],

        ...,

        [[0.16979644],
         [0.17125224],
         [0.17376934],
         ...,
         [0.15440892],
         [0.15580216],
         [0.16157655]],

        [[0.17125224],
         [0.17376934],
         [0.16724065],
         ...,
         [0.15580216],
         [0.16157655],
         [0.16123556]],

        [[0.17376934],
         [0.16724065],
         [0.16931118],
         ...,
         [0.16157655],
         [0.16123556],
         [0.16026032]]], dtype=float32),
 array([[[0.15224692],
         [0.15063779],
         [0.14774571],
         ...,
         [0.08688232],
         [0.08603218],
         [0.08327933]],

        [[0.15063779],
         [0.14774571],
         [0.14613624],
```

```
         ...,
         [0.08603218],
         [0.08327933],
         [0.07895568]],

        [[0.14774571],
         [0.14613624],
         [0.14700587],
         ...,
         [0.08327933],
         [0.07895568],
         [0.08035833]],

        ...,

        [[0.02137815],
         [0.01693882],
         [0.01747251],
         ...,
         [0.00491451],
         [0.0048523 ],
         [0.00089352]],

        [[0.01693882],
         [0.01747251],
         [0.01821234],
         ...,
         [0.0048523 ],
         [0.00089352],
         [0.00023069]],

        [[0.01747251],
         [0.01821234],
         [0.01607927],
         ...,
         [0.00089352],
         [0.00023069],
         [0.        ]]], dtype=float32))
```

- created a Sequential model with a SimpleRNN layer of 50 units and ReLU activation, suitable for time-series data processing.
- The input shape for the SimpleRNN layer is set to (sequence_length, 1), matching the shape of x_train.
- added a Dense layer with 1 unit as the output layer to predict a single value for each sequence.

In [29]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

model = Sequential([
    SimpleRNN(50, activation='relu', input_shape=(x_train.shape[1], 1)),
    Dense(1)
])
```

```
c:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src
\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` arg
ument to a layer. When using Sequential models, prefer using an `Input(shape)` ob
ject as the first layer in the model instead.
  super().__init__(**kwargs)
```

- compiled the model using the Adam optimizer, which adapts learning rates during training for efficient convergence.
- set the loss function to Mean Squared Error, ideal for regression tasks like time-series forecasting.

In [30]:
```
model.compile(optimizer='adam', loss='mean_squared_error')
```

- trained the model on x_train and y_train for 50 epochs, with a batch size of 32 to control updates per training batch.
- included validation_data as (x_test, y_test) to monitor the model's performance on unseen data after each epoch.
- This setup tracks training progress and helps prevent overfitting by using both training and validation data.

In [31]:
```
history = model.fit(x_train, y_train, epochs=50, batch_size=32, validation_data=
```

```
Epoch 1/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.0228 - val_loss: 9.7903e-04
Epoch 2/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 1.7303e-04 - val_loss: 2.0708e-05
Epoch 3/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.0496e-05 - val_loss: 1.2874e-05
Epoch 4/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.1938e-05 - val_loss: 1.3887e-05
Epoch 5/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.0532e-05 - val_loss: 1.6519e-05
Epoch 6/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.5846e-05 - val_loss: 1.2287e-05
Epoch 7/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.3494e-05 - val_loss: 1.3019e-05
Epoch 8/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.7416e-05 - val_loss: 1.2226e-05
Epoch 9/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.4403e-05 - val_loss: 1.3889e-05
Epoch 10/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.6141e-05 - val_loss: 3.0730e-05
Epoch 11/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.7487e-05 - val_loss: 1.2913e-05
Epoch 12/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.1956e-05 - val_loss: 1.3215e-05
Epoch 13/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2626e-05 - val_loss: 1.3142e-05
Epoch 14/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.3668e-05 - val_loss: 2.2998e-05
Epoch 15/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 1.4629e-04 - val_loss: 1.6313e-05
Epoch 16/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.9700e-05 - val_loss: 1.5791e-05
Epoch 17/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.3285e-05 - val_loss: 1.7922e-05
Epoch 18/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 1.1074e-04 - val_loss: 1.7821e-05
Epoch 19/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.0629e-05 - val_loss: 5.1676e-05
Epoch 20/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.7448e-05 - val_loss: 3.0262e-05
Epoch 21/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 1.0741e-04 - val_loss: 3.0158e-05
Epoch 22/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.2536e-05 - val_loss: 7.8522e-05
Epoch 23/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.9803e-05 - val_loss: 9.2827e-05
Epoch 24/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 1.0139e-04 - val_loss: 4.7734e-05
Epoch 25/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.2084e-05 - val_loss: 5.6436e-05
Epoch 26/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.5303e-05 - val_loss: 7.0249e-05
Epoch 27/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.2960e-05 - val_loss: 4.1804e-05
Epoch 28/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.2981e-05 - val_loss: 6.2996e-05
Epoch 29/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.8299e-05 - val_loss: 1.3634e-04
Epoch 30/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.6190e-05 - val_loss: 4.4956e-05
```

```
Epoch 31/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.4526e-05 - val_loss: 9.1757e-05
Epoch 32/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.0900e-05 - val_loss: 4.4539e-05
Epoch 33/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.9766e-05 - val_loss: 4.4009e-05
Epoch 34/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 9.4020e-05 - val_loss: 9.8293e-05
Epoch 35/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.8226e-05 - val_loss: 4.5018e-05
Epoch 36/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.3395e-05 - val_loss: 7.2891e-05
Epoch 37/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 7.8704e-05 - val_loss: 3.9896e-05
Epoch 38/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.2200e-05 - val_loss: 8.5654e-05
Epoch 39/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.1440e-05 - val_loss: 9.4426e-05
Epoch 40/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.0268e-05 - val_loss: 6.6794e-05
Epoch 41/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 8.0337e-05 - val_loss: 9.9476e-05
Epoch 42/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.2974e-05 - val_loss: 7.0504e-05
Epoch 43/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.5203e-05 - val_loss: 4.9368e-05
Epoch 44/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 8.7591e-05 - val_loss: 6.1487e-05
Epoch 45/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 7.8586e-05 - val_loss: 7.7440e-05
Epoch 46/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.4162e-05 - val_loss: 4.4111e-05
Epoch 47/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 6.8256e-05 - val_loss: 5.5714e-05
Epoch 48/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.7018e-05 - val_loss: 3.3207e-05
Epoch 49/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.3396e-05 - val_loss: 1.7134e-05
Epoch 50/50
62/62 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 6.6380e-05 - val_loss: 4.5663e-05
```

- generated predictions on the x_test data using the trained model.
- used scaler.inverse_transform to convert the predictions back to their original scale, making them comparable to the actual prices.
- Similarly, transformed y_test back to the original scale, providing the true values for comparison with the predictions.

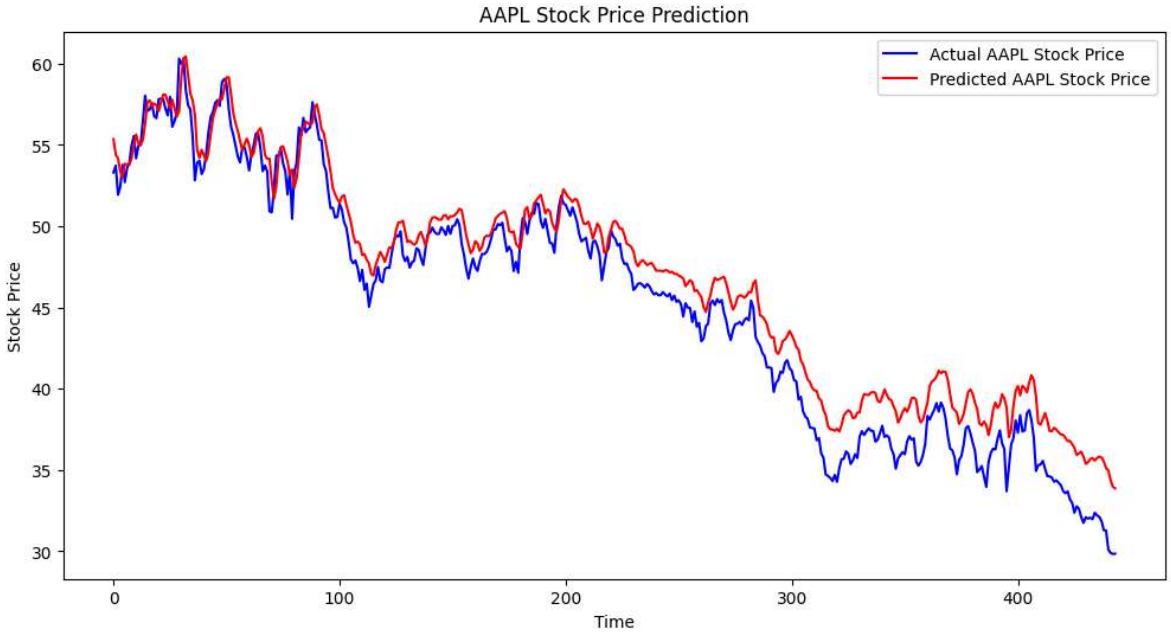
In [32]:
```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
actual_prices = scaler.inverse_transform(y_test.reshape(-1, 1))
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step
```

In [33]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(actual_prices, color='blue', label='Actual AAPL Stock Price')
plt.plot(predictions, color='red', label='Predicted AAPL Stock Price')
```

```
plt.title('AAPL Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```



- evaluated the model's performance by calculating Mean Absolute Error (MAE), which measures the average absolute difference between predicted and actual prices.
- calculated Root Mean Squared Error (RMSE) by taking the square root of Mean Squared Error, providing insight into the model's prediction accuracy.

In [34]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

mae = mean_absolute_error(actual_prices, predictions)
rmse = np.sqrt(mean_squared_error(actual_prices, predictions))

print(f"MAE: {mae}")
print(f"RMSE: {rmse}")
```

```
MAE: 1.7192814350128174
RMSE: 2.009431838989258
```