**INTRODUCTION**

**1.1 Project Overview**

Crude oil is among the main assets in this day and age, it is the central fuel and its expense straightforwardly affects the worldwide environment, our economy and oil investigation, abuse and different exercises. Expectation of oil costs has turned into a need of great importance, it is a shelter to numerous huge and little ventures, people, and the public authority.

As a vital info figure in modern creation, the value unpredictability of raw petroleum frequently achieves monetary unpredictability, so estimating raw petroleum cost has forever been an urgent issue in financial matters. In our review, we built a LSTM (short for Long Momentary Memory brain organisation) model to lead this determining in view of information

**1.2 Purpose**

The major goal of this project is to employ neural networks to forecast the price of crude oil. This choice enables us to purchase crude oil at the appropriate time. The best solution for this type of prediction is time series analysis because we are using past data on crude oil prices to forecast future prices. Therefore, to complete the objective, by designing an RNN (Recurrent Neural Network) with an LSTM (Long Short Term Memory).

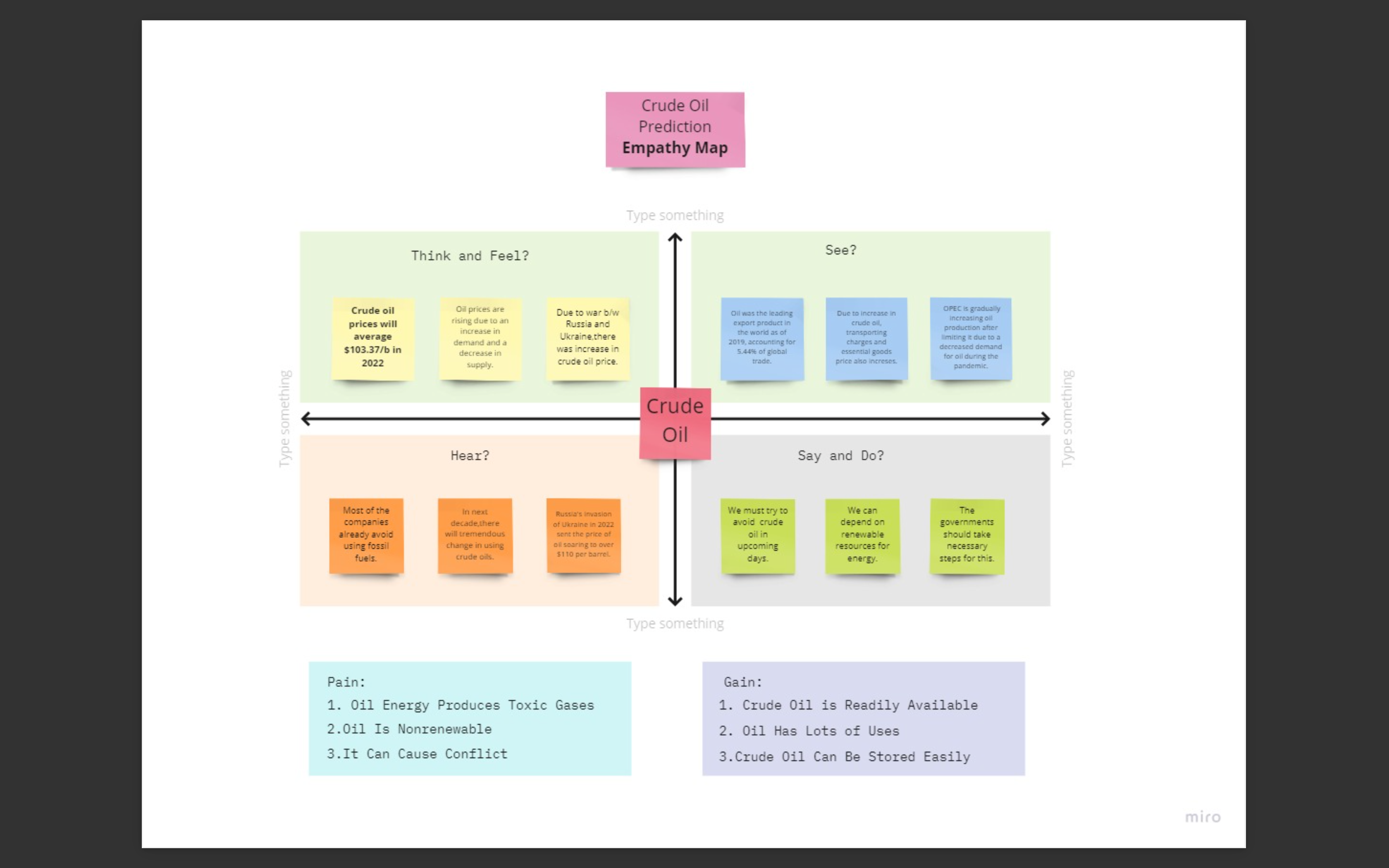
**LITERATURE SURVEY**

| **S.no** | **Author** | **Title** | **Objective** |
| --- | --- | --- | --- |
| **1** | Nidhi Moitra et al. (2020) | Crude Oil Price Prediction Using Lstm [1] | In this paper, Recurrent neural networks that are LSTM-based are used to predict the price of crude oil. The most effective and powerful models for processing time-series-based sequential data are recurrent neural networks (RNNs). In addition to prediction, LSTM variations can be utilised for tasks including polyphonic modelling, speech recognition, and handwriting recognition. |
| 2 | Varun Gupta et al. (2018) | Crude Oil Price Prediction Using LSTM Networks [2] | In this study, For the objective of predicting the price of crude oil, LSTM-based recurrent neural networks have been utilised. One of the most effective RNN architectures is LSTM. The hidden layer of the network's LSTM introduces the memory cell, which makes it well-suited to grasp the changing structure of data with a high capacity for prediction. |
| 3 | Zhenda Hu et al. (2021) | Crude oil price prediction using CEEMDAN and LSTMattention with news sentiment index | This paper combines Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN), Long Short-Term Memory (LSTM) with attention mechanism and addition, following the well known “decomposition and ensemble” framework to study the crude oil prices |
| 4 | Kexian Zhang et al. (2022) | Forecasting crude oil price using LSTM neural networks | An ANN (short for Artificial Neural Network) model and a typical ARIMA (short for Autoregressive Integrated Moving Average) model are taken as the comparable models. The results show that the LSTM model has strong generalisation ability, with stable applicability in forecasting crude oil prices with different timescales. |
| 5 | Shaolong Sun et al. (2021) | Analysis and forecasting of crude oil price based on the variable selection-LSTM integrated model | This paper assesses and selects core influence factors with the elastic-net regularised generalised linear Model (GLMNET), spike-slab lasso method, and Bayesian model average (BMA) and the new machine learning method long short-term Memory Network (LSTM) is developed for crude oil price forecasting. |
| 6 | Norshakirah Aziz et al. (2020) | Predictive analytics for crude oil price using rnn-lstm neural network | This study demonstrated the use of RNN-LSTM networks for predicting the crude oil price based on historical data alongside other technical analysis indicators. This study aims to certify the capability of a prediction model built based on the RNN-LSTM network to predict the future price of crude oil. |
| 7 | Rayan H. Assaad et al. (2021) | Predicting the Price of Crude Oil and its Fluctuations Using LSTM, and Convolutional Neural Networks | Deep neural networks, long short term memory (LSTM) neural networks, and a combination of convolutional and LSTM neural networks are being used here. The findings suggest that LSTM networks are the best architectures to predict the crude oil price. The outcomes of this paper could potentially help in making the oil price prediction mechanism more traceable. |
| 8 | Kaijian He et al. (2017) | Forecasting Crude Oil Prices: a Deep Learning based Model | In this paper, we use the deep learning model to capture the unknown complex nonlinear characteristics of the crude oil price movement. We further propose a new hybrid crude oil price forecasting model based on the deep learning model |
| 9 | Rajesh Prasad et al. (2020) | CPPCNDL: Crude oil price prediction using complex network and deep learning algorithms | This paper proposed a hybrid model for crude oil price prediction that uses the complex network analysis and long short-term memory (LSTM) of the deep learning algorithms. The complex network analysis tool called the visibility graph is used to map the dataset on a network and K-core centrality was employed to extract the non-linearity features of crude oil and reconstruct the dataset |
| 10 | Lin Yao et al. (2021) | Prediction of Oil Price Using LSTM | In this paper, we selected the LSTM algorithm to do the oil price’s prediction, to reach good results. RMSE and MAE are selected to represent the prediction’s precision. In this paper, we use a two-layer LSTM network, and the Dense layer is used for the output layer |

**IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**

An empathy map is a widely-used visualisation tool within the field of UX and HCI practice. In relation to empathetic design, the primary purpose of an empathy map is to bridge the understanding of the end user.



**3.2 Ideation and Brainstorming**

Ideation is the creative process of generating, developing, and communicating new ideas, where an idea is understood as a basic element of thought that can be either visual, concrete, or abstract.

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members

**3.3 Proposed Solution**

Prediction of crude oil price in future by applying neural networks. Time series analysis is done so that the previous history is analysed to predict the future crude oil price.

RNN model along with LSTM would be used to predict the future crude oil prices.

There is no much better system with good accuracy for crude oil price prediction.

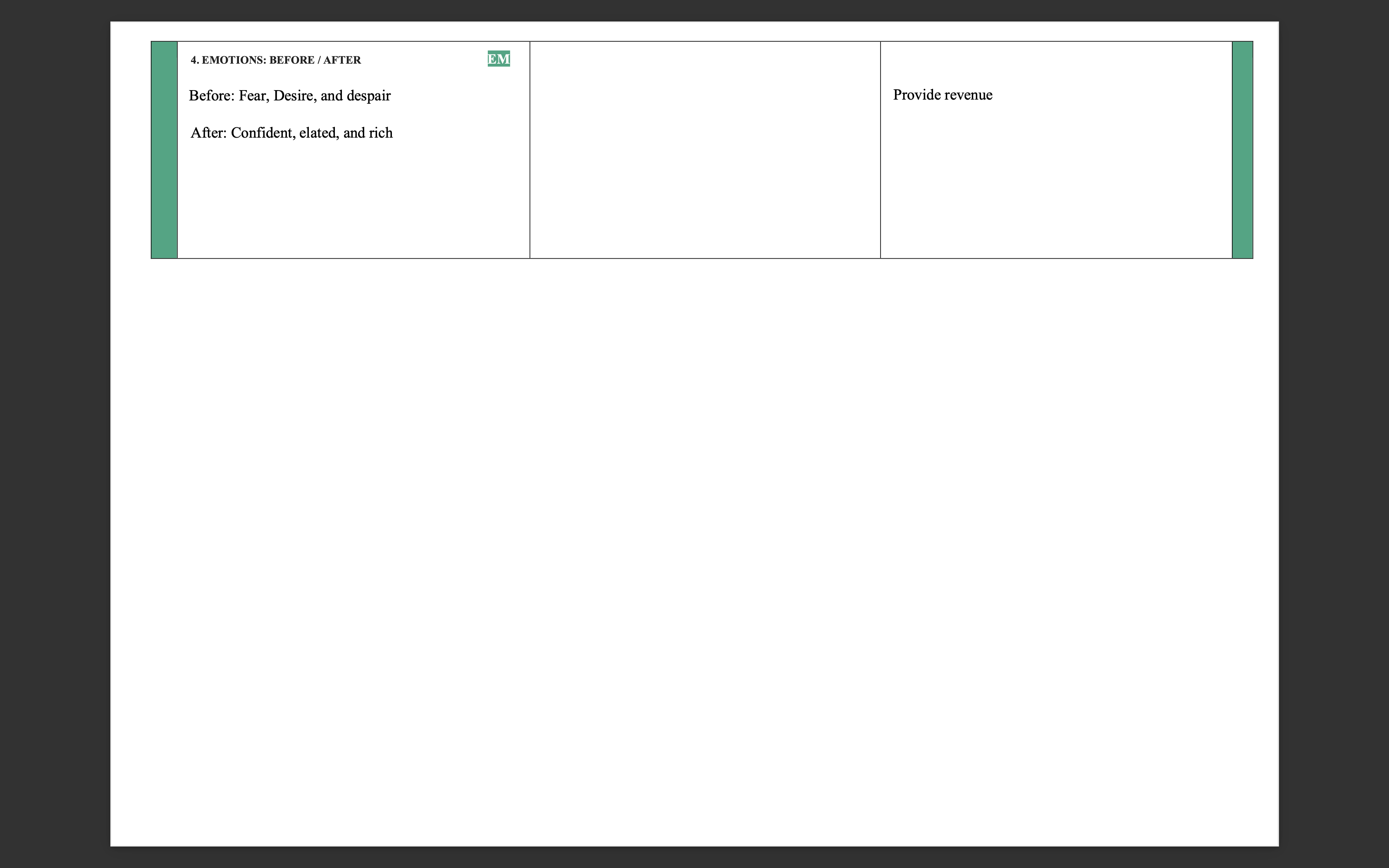
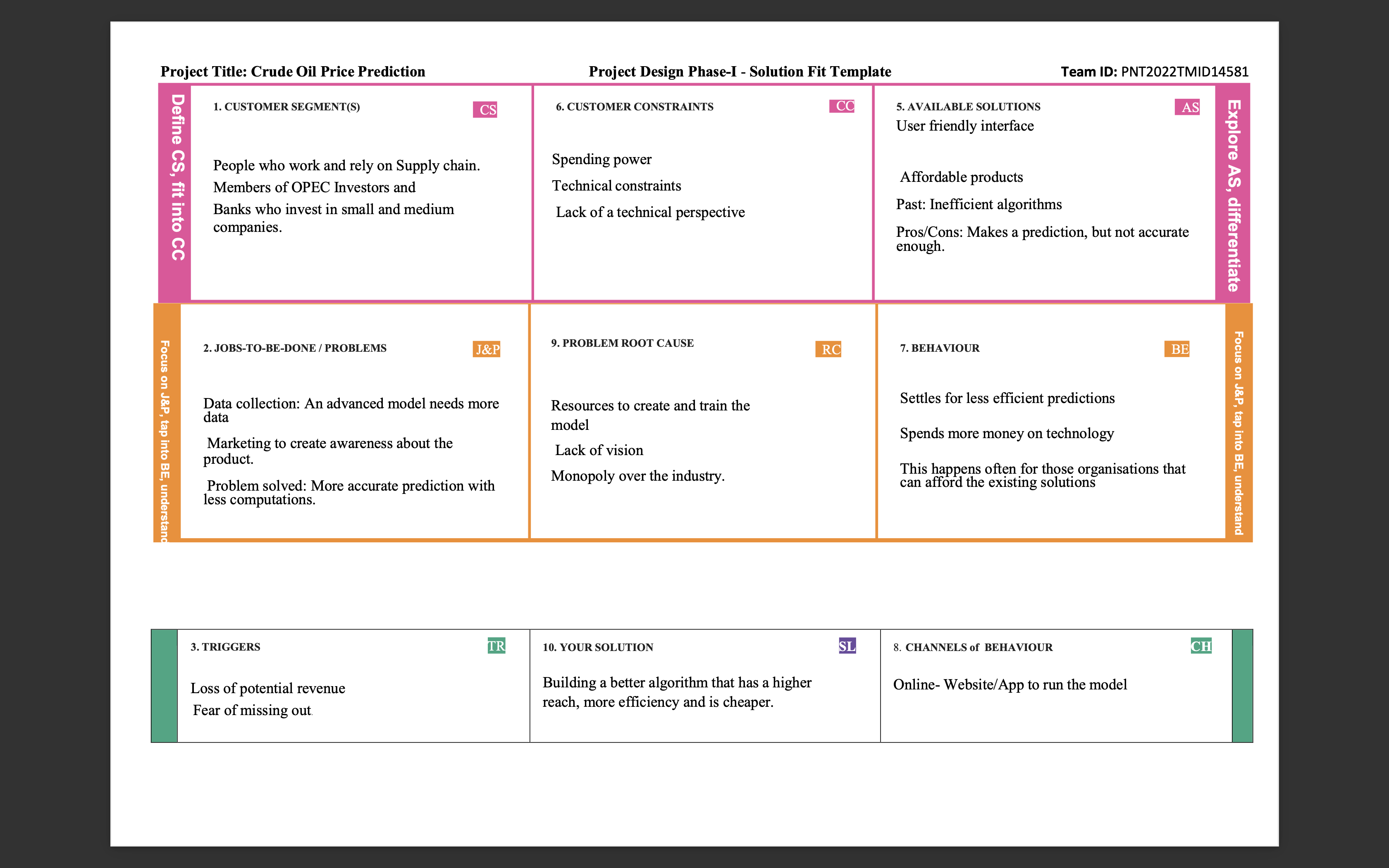
This system would give a high performance with low computation cost.

It helps oil importers in saving their penny as they could now forecast the oil price in future and buy accordingly when the oil price drops. This will have a greater impact on economic/political power as it stops many oil importers from quitting their business.

As oil is the largest traded commodity, this would definitely be a never ending problem and the solution would be used for life-time.

This solution is scalable to like in future more features can be added and it could even be useful for oil exporters.

**3.4 Problem Solution Fit**



**REQUIREMENT ANALYSIS**

**4.1 Functional requirements**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirements (Epic)** | **Sub Requirement (Story/Sub- Task)** |
| FR-1 | User Application | User Direct Open with Google Play Store App  User Can Download the Crude Oil Price |
| FR-2 | User Products Available | User Using the Application There Are So Many Products in Crude Oil Price App  User Update the Energy and Oil Price Instant the Application |
| FR-3 | User Additional Features | User Can Read Latest News and View Oil Price Charts  User View Major Energy Quotes User Can Using a Multiple Color Theme |
| FR-4 | User Exception | User Can Exchange Rates and Currency Converter |

**4.2 Non Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **FR NO.** | **Non Functional Requirement** | **Description** |
| **FR-1** | Usability | Used to improve to the Accuracy of crude oil price prediction |
| **FR-2** | Security | In the rising oil price can even shift economically/political power from oil importers to oil exporters communications will be secured |

|  |  |  |
| --- | --- | --- |
| **FR-4** | Performance | Performance of the this project is to improve to the accuracy of crude oil price prediction |
| **FR-5** | Availability | The Availability Solution is More Benefit for and the exporters in the crude oil price prediction |
| **FR-6** | Scalability | The scalability are 90%-95% |

**PROJECT DESIGN**

**5.1 Data Flow Diagrams**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

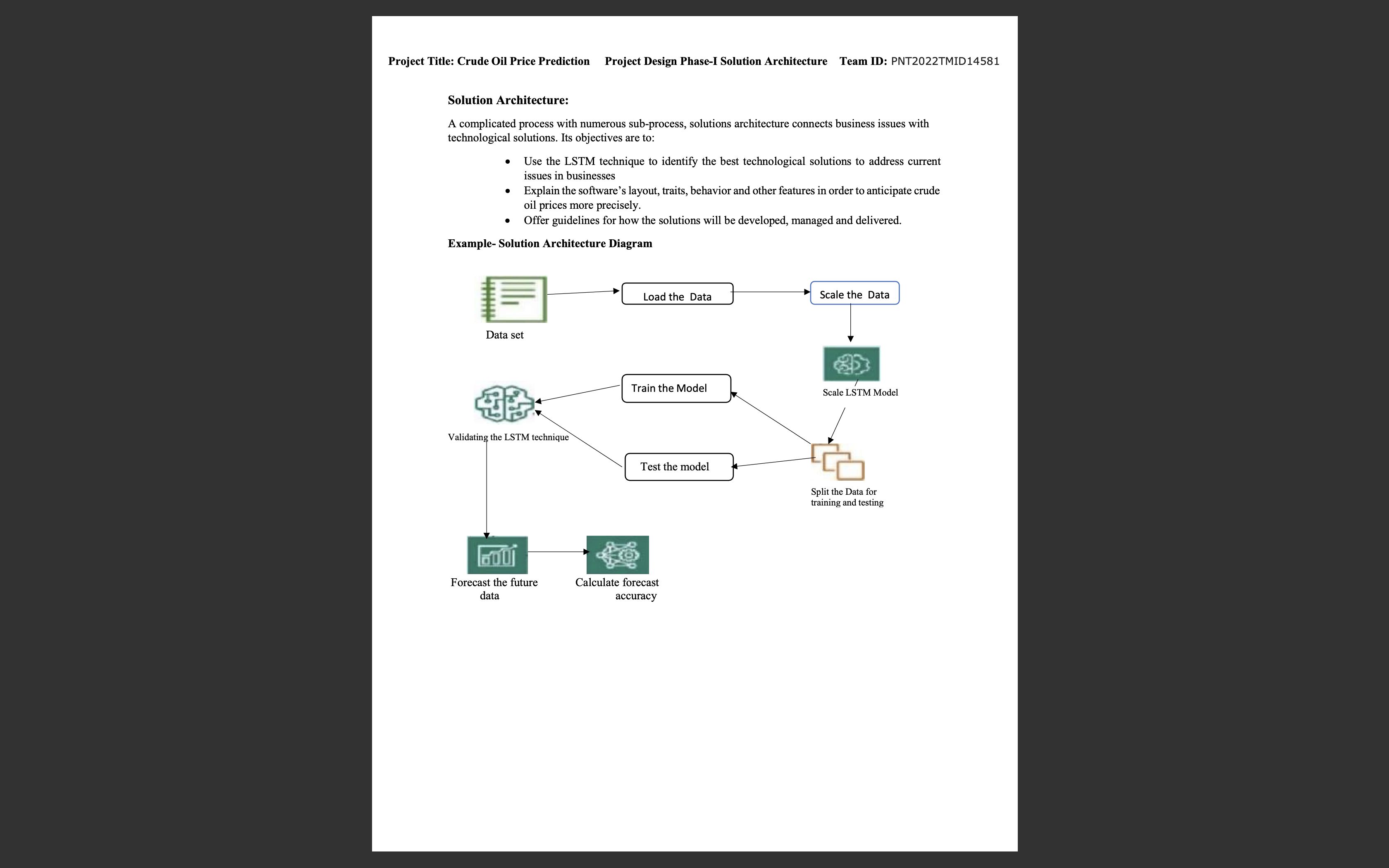
**5.2 Solution & Technical Architecture**

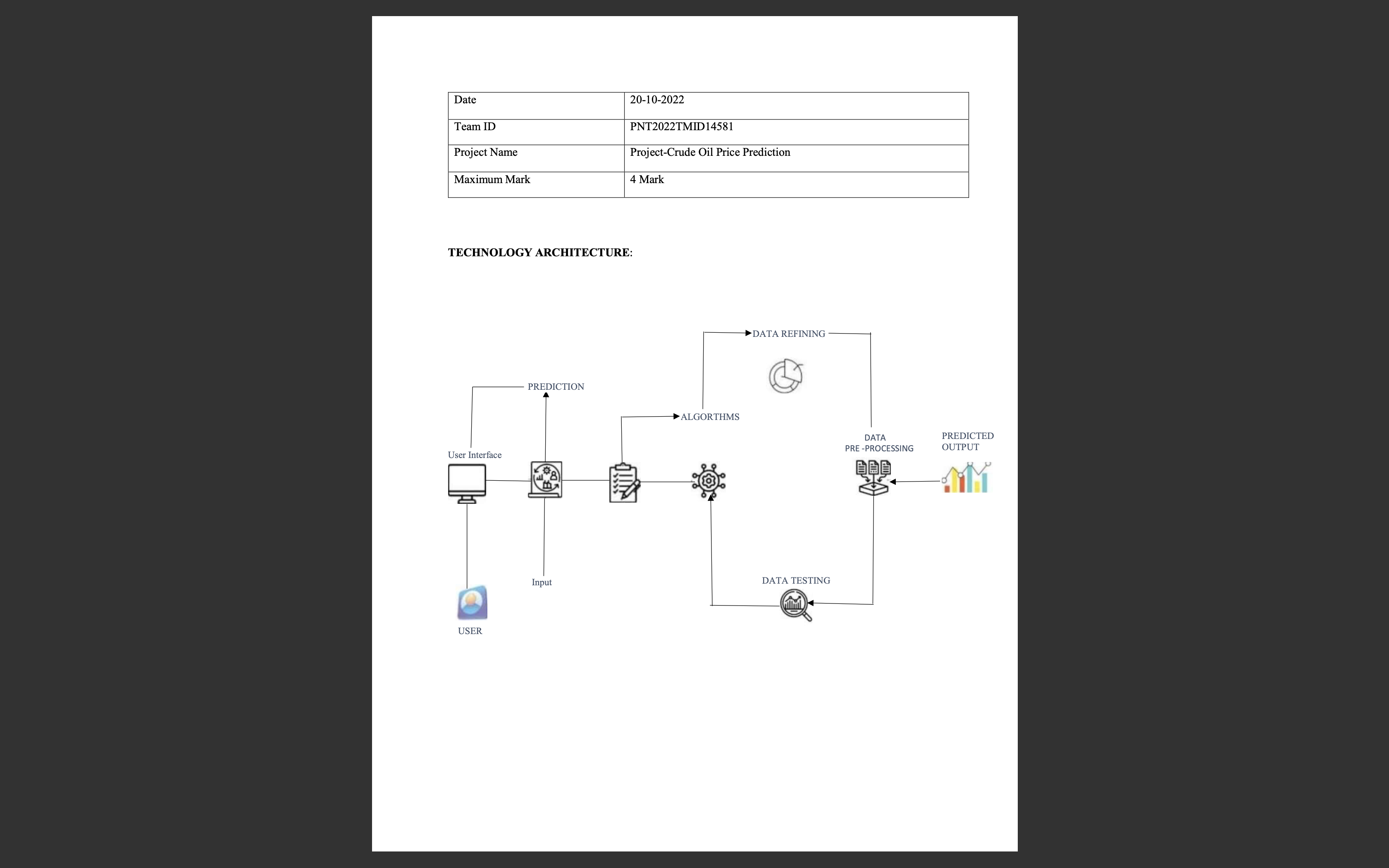
A complicated process with numerous sub-process, solutions architecture connects business issues with technological solutions. Its objectives are to:

• Use the LSTM technique to identify the best technological solutions to address current issues in businesses

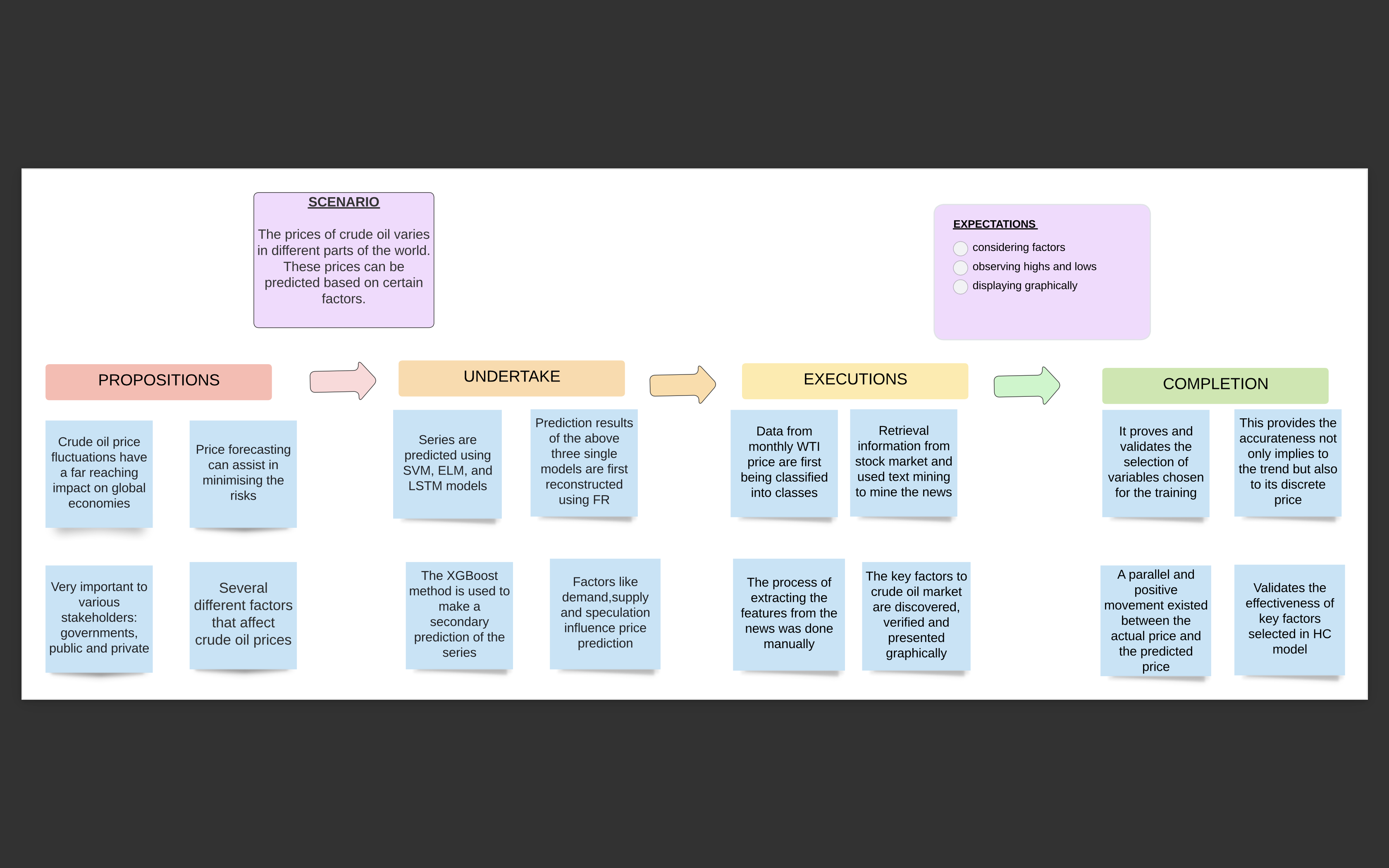
• Explain the software’s layout, traits, behavior and other features in order to anticipate crude oil prices more precisely.

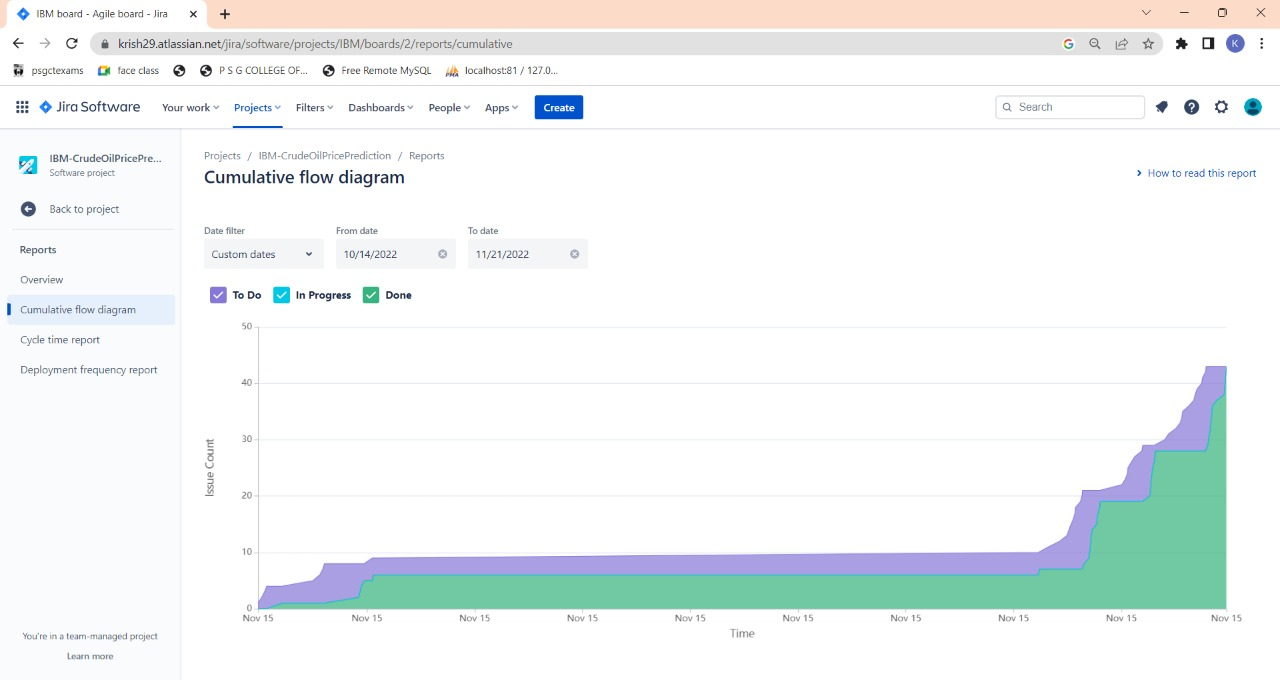
• Offer guidelines for how the solutions will be developed, managed and delivered.





**5.3 CUSTOMER JOURNEY:**



**6.3 Reports from JIRA** It is a product for software developers, project managers and other software development teams. ****

**CODING & SOLUTIONING**

**7.1 Feature 1**

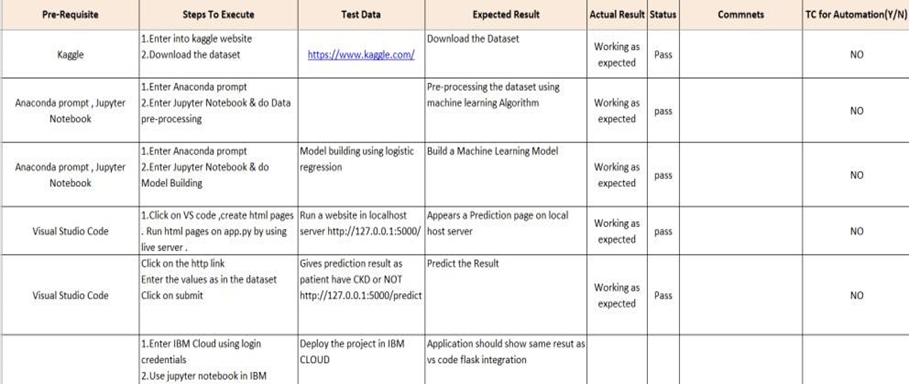
A LSTM Price forecasting machine learning model (Add in GITHUB)

**7.2 Feature 2**

A User Interface for forecasting and predicting crude oil price based on the past 10 days price (Add in Github)

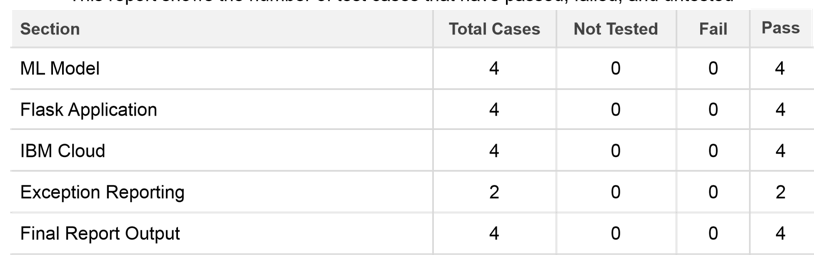
**TESTING**

**8.1 Test Cases**

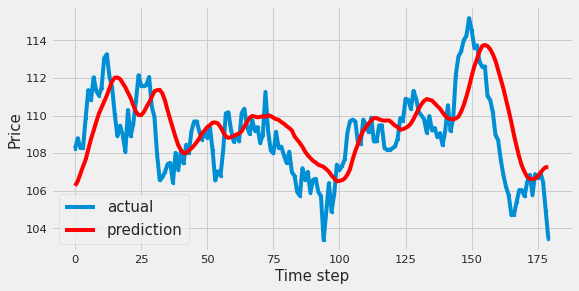
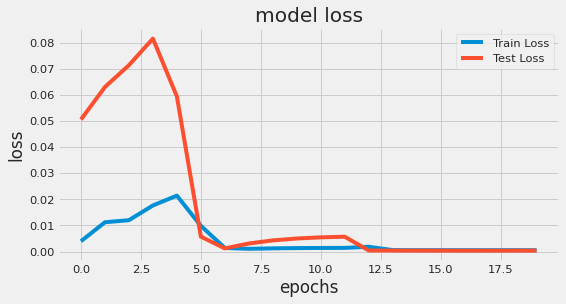
****

**8.2 User Acceptance Testing**

**1.** **Test Case Analysis**

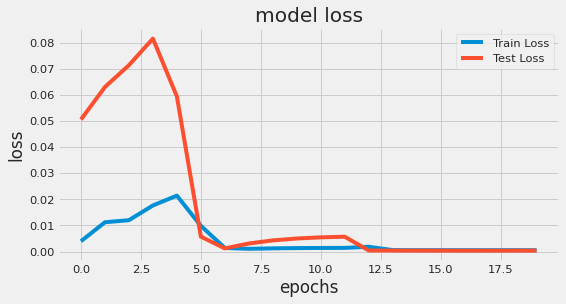
****

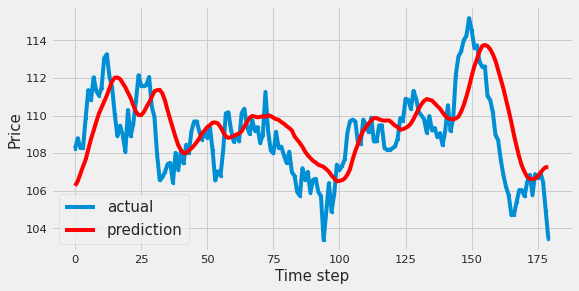
**8.3 Performance Testing**



**RESULTS**

**9.1 Performance Metrics**





**ADVANTAGES**

* continuously captures the unstable pattern of the crude oil price
* We’ll gain valuable insight
* forecasting can assist in minimising the risks associated with volatility in oil prices.

**DISADVANTAGES**

* The system developed is never 100% accurate.
* It can be time-consuming and resource-intensive

**CONCLUSION**

Therefore the DL model was deployed as a web app and the user interface is handy for stakeholders who do not have much knowledge in programming. The predicted value is displayed in the user interface.

**FUTURE SCOPE**

To have a better grasp of the pricing value, transform the machine learning model from univariate to multivariate. This can significantly improve prediction accuracy.

**APPENDIX**

**Source Code**

**Importing The Libraries**

import numpy as np

import pandas as pd

import datetime

from pylab import rcParams

import matplotlib.pyplot as plt

import warnings

import itertools

import statsmodels.api as sm

import seaborn as sns

sns.set\_context("paper", font\_scale=1.3)

sns.set\_style('white')

import math

from sklearn.preprocessing import MinMaxScaler

plt.style.use('fivethirtyeight')

**Importing The Dataset**

parser=lambda x: pd.datetime.strptime(x, '%b %d, %Y')

df=pd.read\_csv('CrudeOilPrices.csv',parse\_dates=['Date'], date\_parser=parser)

df=df.sort\_values('Date')

df=df.groupby('Date')['Price'].sum().reset\_index()

df.set\_index('Date', inplace=True)

df=df.loc[datetime.date(year=2000,month=1,day=1):]

df.head(10)

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: FutureWarning: The pandas.datetime class is deprecated and will be removed from pandas in a future version. Import from datetime module instead.

"""Entry point for launching an IPython kernel.

Price

Date

2000-01-04 23.95

2000-01-05 23.72

2000-01-06 23.55

2000-01-07 23.35

2000-01-10 22.77

2000-01-11 23.93

2000-01-12 24.62

2000-01-13 24.90

2000-01-14 25.50

2000-01-17 25.99

**Handling Missing Data**

df.isnull().sum()

Price 0

dtype: int64

#There is no null value

**Data Visualization**

y = df['Price'].resample('MS').mean()

y.plot(figsize=(15, 6))

plt.show()

rcParams['figure.figsize'] = 18, 8

decomposition =sm.tsa.seasonal\_decompose(y, model='additive')

fig = decomposition.plot()

plt.show()

**Feature Scaling**

sc = MinMaxScaler(feature\_range = (0, 1))

df=sc.fit\_transform(df)

**Splitting Data Into Train And Test**

train\_set\_length=int(len(df) \* 0.70)

test\_set\_length=len(df)-train\_set\_length

train,test=df[0:train\_set\_length,:],df[train\_set\_length:len(df),:]

**Creating a dataset with sliding window**

def create\_data\_set(\_data\_set,\_look\_back=1):

data\_x, data\_y = [], []

for i in range(len(\_data\_set)-\_look\_back - 1):

a = \_data\_set[i:(i + \_look\_back), 0]

data\_x.append(a)

data\_y.append(\_data\_set[i + \_look\_back, 0])

return np.array(data\_x), np.array(data\_y)

look\_back=10

X\_train,Y\_train,X\_test,Ytest = [],[],[],[]

X\_train,Y\_train=create\_data\_set(train,look\_back)

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

X\_test,Y\_test=create\_data\_set(test,look\_back)

X\_test = np.reshape(X\_test,(X\_test.shape[0], X\_test.shape[1], 1))

**Importing Model Libraries**

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

from sklearn.metrics import mean\_squared\_error

from keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

**Initializing The Model, Adding LSTM Layers, Adding Output Layers,Configuring The Learning Process**

regressor = Sequential()

regressor.add(LSTM(units = 50, return\_sequences = True, input\_shape = (X\_train.shape[1], 1)))

regressor.add(LSTM(units = 50, return\_sequences = True))

regressor.add(LSTM(units = 50))

regressor.add(Dense(units = 1))

regressor.compile(optimizer='adam',loss = 'mean\_squared\_error')

**Train The Model**

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss',patience=5)

history =regressor.fit(X\_train, Y\_train, epochs = 20, batch\_size = 15,validation\_data=(X\_test, Y\_test), callbacks=[reduce\_lr],shuffle=False)

Epoch 1/20

234/234 [==============================] - 14s 28ms/step - loss: 0.0039 - val\_loss: 0.0506 - lr: 0.0010

Epoch 2/20

234/234 [==============================] - 6s 28ms/step - loss: 0.0112 - val\_loss: 0.0631 - lr: 0.0010

Epoch 3/20

234/234 [==============================] - 7s 31ms/step - loss: 0.0120 - val\_loss: 0.0715 - lr: 0.0010

Epoch 4/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0177 - val\_loss: 0.0816 - lr: 0.0010

Epoch 5/20

234/234 [==============================] - 6s 25ms/step - loss: 0.0214 - val\_loss: 0.0595 - lr: 0.0010

Epoch 6/20

234/234 [==============================] - 5s 22ms/step - loss: 0.0098 - val\_loss: 0.0057 - lr: 0.0010

Epoch 7/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0014 - val\_loss: 0.0012 - lr: 0.0010

Epoch 8/20

234/234 [==============================] - 5s 22ms/step - loss: 0.0010 - val\_loss: 0.0031 - lr: 0.0010

Epoch 9/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0012 - val\_loss: 0.0043 - lr: 0.0010

Epoch 10/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0013 - val\_loss: 0.0050 - lr: 0.0010

Epoch 11/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0014 - val\_loss: 0.0054 - lr: 0.0010

Epoch 12/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0014 - val\_loss: 0.0057 - lr: 0.0010

Epoch 13/20

234/234 [==============================] - 5s 21ms/step - loss: 0.0018 - val\_loss: 3.9126e-04 - lr: 1.0000e-04

Epoch 14/20

234/234 [==============================] - 5s 22ms/step - loss: 5.3776e-04 - val\_loss: 3.6034e-04 - lr: 1.0000e-04

Epoch 15/20

234/234 [==============================] - 5s 21ms/step - loss: 5.1708e-04 - val\_loss: 3.3980e-04 - lr: 1.0000e-04

Epoch 16/20

234/234 [==============================] - 5s 21ms/step - loss: 5.0532e-04 - val\_loss: 3.2689e-04 - lr: 1.0000e-04

Epoch 17/20

234/234 [==============================] - 5s 21ms/step - loss: 4.9852e-04 - val\_loss: 3.1859e-04 - lr: 1.0000e-04

Epoch 18/20

234/234 [==============================] - 6s 25ms/step - loss: 4.9422e-04 - val\_loss: 3.1326e-04 - lr: 1.0000e-04

Epoch 19/20

234/234 [==============================] - 5s 22ms/step - loss: 5.1797e-04 - val\_loss: 3.3054e-04 - lr: 1.0000e-05

Epoch 20/20

234/234 [==============================] - 5s 21ms/step - loss: 4.8829e-04 - val\_loss: 3.4008e-04 - lr: 1.0000e-05

Testing The Model

test\_predict=regressor.predict(X\_test)

47/47 [==============================] - 2s 7ms/step

**Model Evaluation**

Y\_train = sc.inverse\_transform([Y\_train])

test\_predict = sc.inverse\_transform(test\_predict)

Y\_test = sc.inverse\_transform([Y\_test])

print('Mean Absolute Error:', mean\_absolute\_error(Y\_test[0], test\_predict[:,0]))

print('Mean Squared Error:',np.sqrt(mean\_squared\_error(Y\_test[0], test\_predict[:,0])))

plt.figure(figsize=(8,4))

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Test Loss')

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epochs')

plt.legend(loc='upper right')

plt.show();

Mean Absolute Error: 2.1092967999827272

Mean Squared Error: 2.654637673385558

aa=[x for x in range(180)]

plt.figure(figsize=(8,4))

plt.plot(aa, Y\_test[0][:180], marker='.', label="actual")

plt.plot(aa, test\_predict[:,0][:180], 'r', label="prediction")

plt.tight\_layout()

sns.despine(top=True)

plt.subplots\_adjust(left=0.07)

plt.ylabel('Price', size=15)

plt.xlabel('Time step', size=15)

plt.legend(fontsize=15)

plt.show();

**Save The Model**

regressor\_json = regressor.to\_json()

with open("regressor.json", "w") as json\_file:

json\_file.write(regressor\_json)

regressor.save("regressor.h5")

import joblib

scaler\_filename="minmaxscaler.save"

joblib.dump(sc,scaler\_filename)

['minmaxscaler.save']

***GitHub & Project Demo Link***

***https://github.com/IBM-EPBL/IBM-Project-13772-1659529660***

***https://drive.google.com/file/d/1uqbd6qf\_JBGjpQd\_inEKqqf403V9TxXZ/view?usp=drivesdk***