Milestone 4: Interpretation of data & Communication of Insights of data

Step 1: Data Cleaning: Changing date format, Checking missing data and indexing

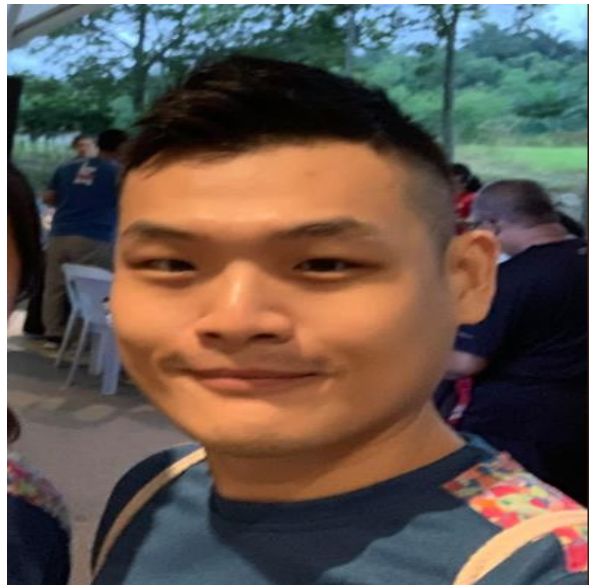
In [ ]:

```
# import library
import pandas as pd
df3.rename(columns = {'close':'Closing_Index'}, inplace = True)
df3['Trading_Date'] = pd.to_datetime(df3['date'], format="%d/%m/%Y", errors='ignore')
df3.sort_index(ascending=False, inplace=True)
df3.info()
df3.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1226 entries, 1225 to 0
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   date            1226 non-null   object        
 1   Closing_Index   1226 non-null   float64       
 2   Trading_Date    1226 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 38.3+ KB
```

Out[8]:

```
date             0
Closing_Index    0
Trading_Date     0
dtype: int64
```



Milestone 4: Interpretation of data & Communication of Insights of data

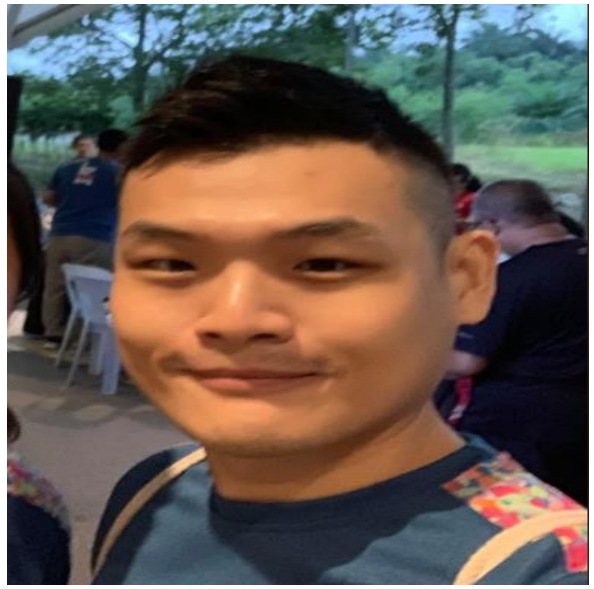
Continued from Step 1: Data Cleaning: Dropping the unnecessary attribute

In [9]:
```
df3.drop(columns=['date'])
df3.set_index('Trading_Date')
df3
```

Out[9]:

Trading_Date	Closing_Index
2015-03-10	1789.729980
2015-03-11	1778.160034
2015-03-12	1786.869995
2015-03-13	1781.750000
2015-03-16	1780.540039
...	...
2020-03-03	1478.640015
2020-03-04	1489.949951
2020-03-05	1491.030029
2020-03-06	1483.099976
2020-03-09	1424.160034

1226 rows × 1 columns

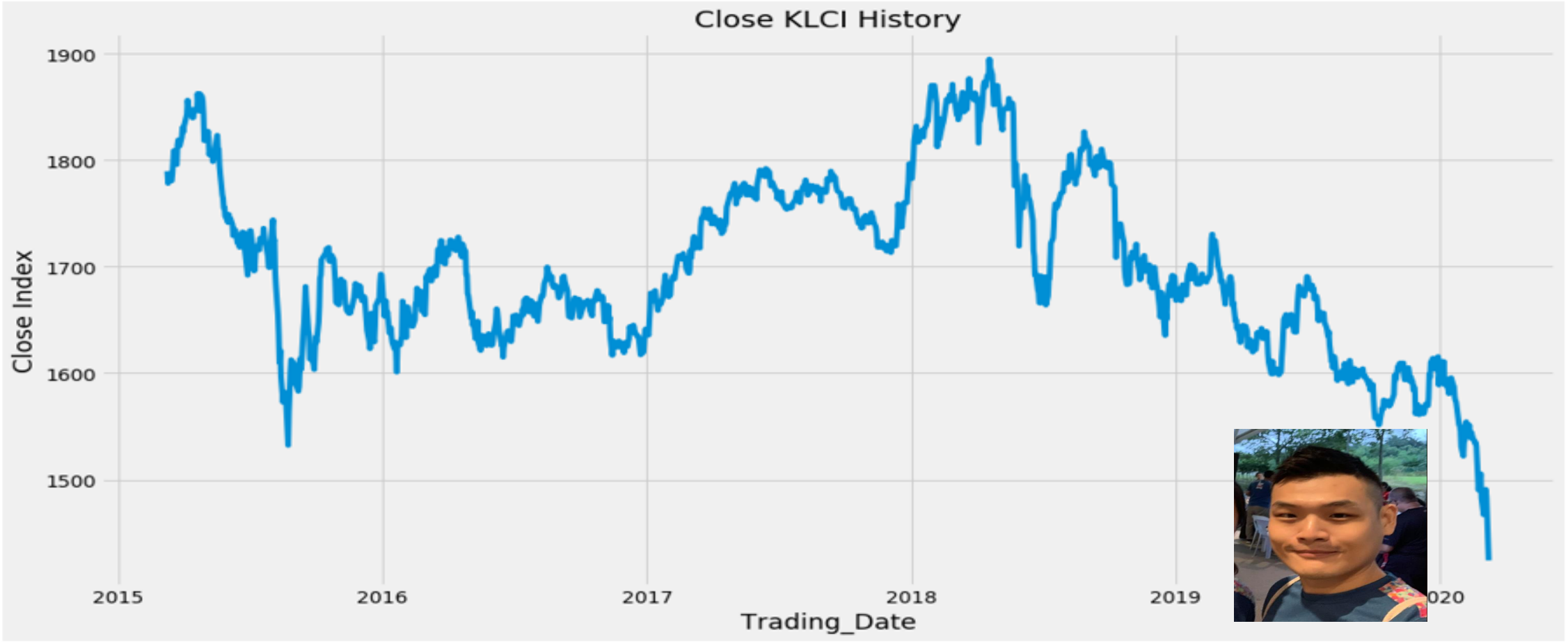


Milestone 4: Interpretation of data & Communication of Insights of data

Step 2: Data interpretation

In [9]:

```
plt.figure(figsize =(16,8))
plt.title("Close KLCI History")
plt.plot(df3['Closing_Index'])
plt.xlabel('Trading_Date', fontsize =18)
plt.ylabel('Close Index', fontsize =18)
plt.show()
```



Milestone 4: Interpretation of data & Communication of Insights of data

Step 3: Machine Leaning: Prediction Using RMSE model

Creating the train and valid data set

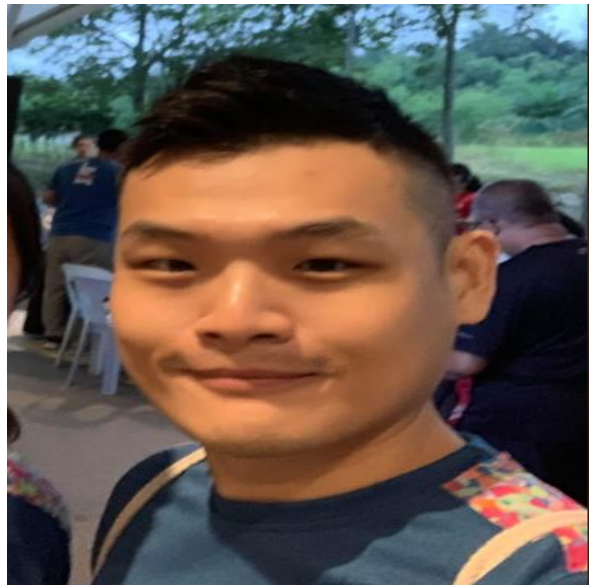
In [244]:

```
# Machine Learning
## create Train and Valid Data
train = df3[:981]
valid = df3[981:]
print(train)
print(valid)
```

```
                Closing_Index
Trading_Date
2015-03-10        1789.729980
2015-03-11        1778.160034
2015-03-12        1786.869995
2015-03-13        1781.750000
2015-03-16        1780.540039
...                       ...
2019-03-06        1686.819946
2019-03-07        1686.949951
2019-03-08        1679.900024
2019-03-11        1664.630005
2019-03-12        1671.280029

[981 rows x 1 columns]
                Closing_Index
Trading_Date
2019-03-13        1678.239990
2019-03-14        1674.520020
2019-03-15        1680.540039
2019-03-18        1690.939941
2019-03-19        1687.680054
...                       ...
2020-03-03        1478.640015
2020-03-04        1489.949951
2020-03-05        1491.030029
2020-03-06        1483.099976
2020-03-09        1424.160034

[245 rows x 1 columns]
```



Milestone 4: Interpretation of data & Communication of Insights of data

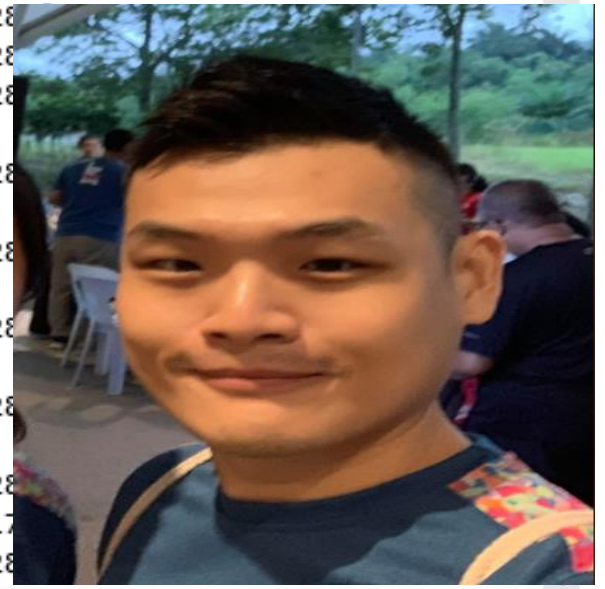
Continued from Step 3: Machine Leaning: Prediction Using RMSE model

Making prediction for validation set by using the RMSE model on the train value

In [290]:

```
# Making Prediction for the validation set and check the RMSE (Root-mean-square error (RMSE)) using the train Value.
preds = []
for i in range(0,valid.shape[0]):
    a = train['Closing_Index'][len(train)-245+i:].sum() + sum(preds)
    b = a/245
    preds.append(b)
    print(preds)
len(preds)
```

```
[1753.1492242448978]
[1753.1492242448978, 1752.7962823234484]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734, 1750.3172360692224]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734, 1750.3172360692224, 1749.918776049097]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734, 1750.3172360692224, 1749.918776049097, 1749.484199563583]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734, 1750.3172360692224, 1749.918776049097, 1749.484199563583, 1749.009440994455]
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
077758209734, 1750.3172360692224, 1749.918776049097, 1749.484199563583, 1749.009440994455, 17
[1753.1492242448978, 1752.7962823234484, 1752.42434860232, 1751.980284837431, 1751.5229389428
```



Milestone 4: Interpretation of data & Communication of Insights of data

Continued from Step 3: Machine Leaning: Prediction Using RMSE model

Joining the prediction data into valid dataset

```
import numpy as np
valid1 = valid
valid1['Prediction'] = np.array(preds)
valid1
```

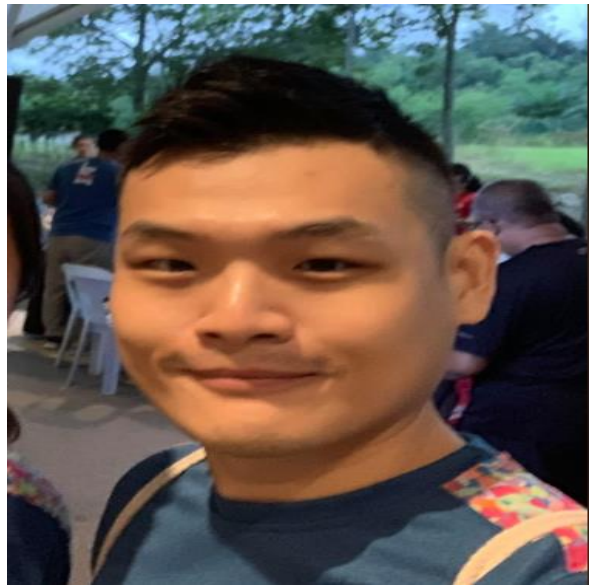
```
C:\Users\L-ven Lew\anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
w-versus-a-copy
```

292]:

Trading_Date	Closing_Index	Prediction
2019-03-13	1678.239990	1753.149224
2019-03-14	1674.520020	1752.796282
2019-03-15	1680.540039	1752.424349
2019-03-18	1690.939941	1751.980285
2019-03-19	1687.680054	1751.522939
...	...	...
2020-03-03	1478.640015	1725.393636
2020-03-04	1489.949951	1725.551079
2020-03-05	1491.030029	1725.708635
2020-03-06	1483.099976	1725.895609
2020-03-09	1424.160034	1726.145672

245 rows × 2 columns



Milestone 4: Interpretation of data & Communication of Insights of data

Continued from Step 3: Machine Leaning: Moving Average Model

Crediting 50-day moving average and 100-day moving average

In [298]:
```
# 100-day moving average
df3['pandas_SMA_100'] = df.iloc[:,1].rolling(window=101).mean()
```

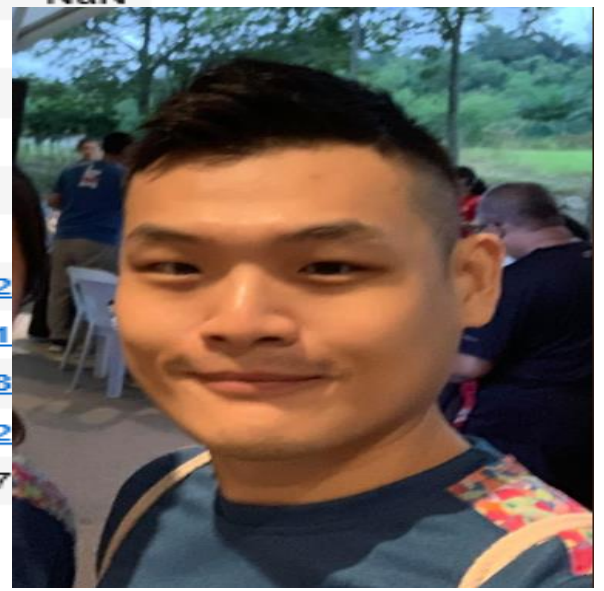
In [299]:
```
# 50-day moving average
df3['pandas_SMA_50'] = df.iloc[:,1].rolling(window=51).mean()
```

In [301]:
```
df3.set_index('Trading_Date')
df3
```

Out[301]:

Trading_Date	date	Closing_Index	pandas_SMA_100	pandas_SMA_50
2015-03-10	10/03/2015	1789.729980	NaN	NaN
2015-03-11	11/03/2015	1778.160034	NaN	
2015-03-12	12/03/2015	1786.869995	NaN	
2015-03-13	13/03/2015	1781.750000	NaN	
2015-03-16	16/03/2015	1780.540039	NaN	
...	...	...	...	
2020-03-03	03/03/2020	1478.640015	1570.624058	1560.2
2020-03-04	04/03/2020	1489.949951	1570.017326	1558.1
2020-03-05	05/03/2020	1491.030029	1569.414950	1555.8
2020-03-06	06/03/2020	1483.099976	1568.684851	1553.2
2020-03-09	09/03/2020	1424.160034	1567.264752	1549.7

1226 rows × 4 columns

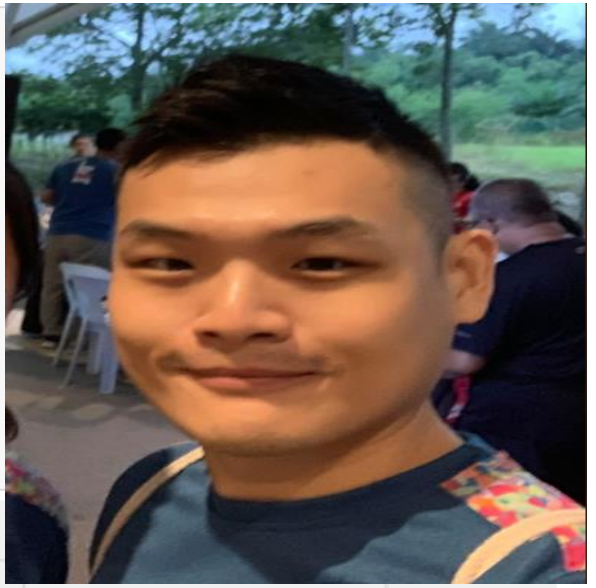


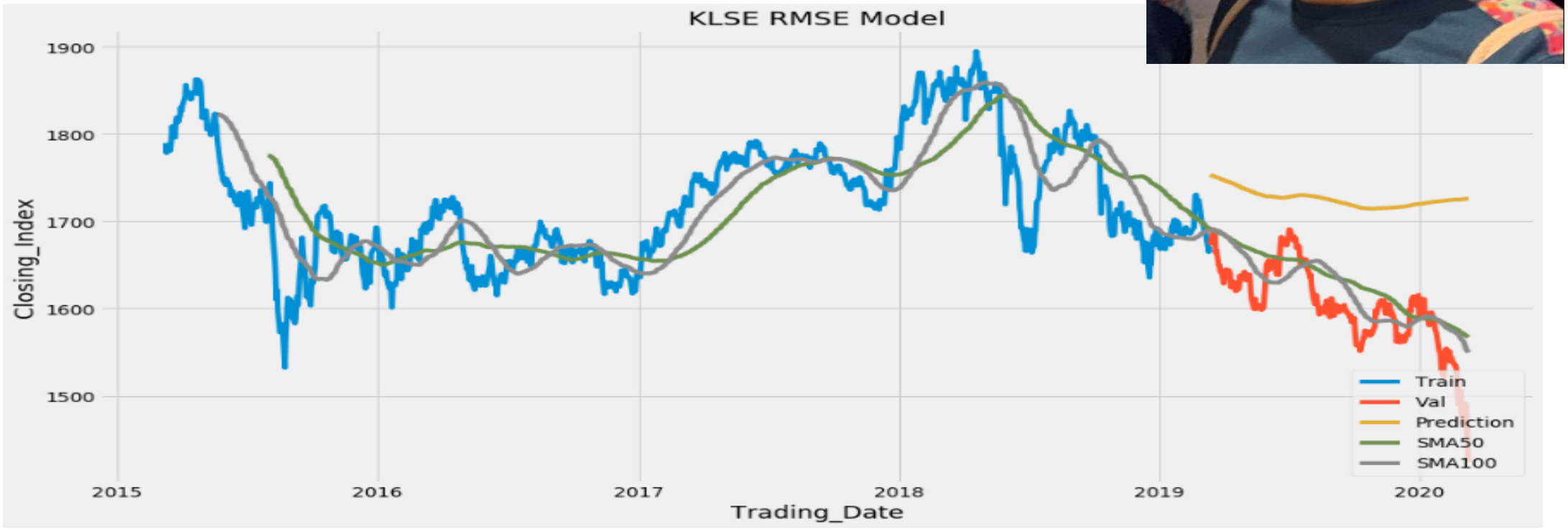
Milestone 4: Interpretation of data & Communication of Insights of data

Step 4: Plotting the predictive model and 50-day and 100-day moving average

In [302]:

```
import math
import pandas_datareader as web
import numpy as np
from pandas.util.testing import assert_frame_equal
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use ('fivethirtyeight')
plt.figure(figsize =(16,8))
plt.title('KLSE RMSE Model')
plt.xlabel('Trading_Date', fontsize =18)
plt.ylabel('Closing_Index', fontsize =18)
plt.plot(train['Closing_Index'])
plt.plot(valid[['Closing_Index','Prediction']])
plt.plot(df3[['pandas_SMA_100','pandas_SMA_50']])
plt.legend(['Train', 'Val','Prediction','SMA50','SMA100'], loc = 'lower right')
plt.show()
```





Milestone 4: Interpretation of data & Communication of Insights of data

Observation: The RMSE predictive model does not seem to be accurate enough in predicting the future performance of KLCI performance. However, this could be largely due to political instability happened in Malaysia coupled will the emergence of covid-19 since late 2019 . Thus, the predictive model is only accurate when there is no unexpected events to cause the sudden change in the performance.