PROJECT REPORT

IBM HACK CHALLENGE 2020

Project Title: Warehouse Management System

Team Name - CompBrats

Aashish Prasad
Darshita Kumar

1. INTRODUCTION

1.1 Overview

1. Our project is a responsive cross-browser web application to predict weekly sales for perishable goods.
2. Data entered by the user and will be stored for future references and for training the model to make accurate predictions.
3. Considering fields (features) in the historical data entered by the warehouse manager, we can predict the quantity of products to be ordered in the future for the warehouse.
4. The web app will also present the statistics obtained after analysing the data in the form of charts for better visualisation. This will enable the warehouse manager to better understand the demand of products and form business strategies accordingly.
5. The web-app is also responsive and hence can be used by the warehouse manager from anywhere using his/her cell phone to check predictions.

1.2 Purpose

1. Our project once deployed, can prevent the loss of warehouse owners.
2. The owners can use the app to make predictions for the upcoming week based on current and previous weeks' data.

2. LITERATURE SURVEY

2.1 Existing problem

1. Perishable goods often get damaged, the implications of which are faced by warehouse owners in the form of losses. Farmer's also may face a reduction in income due to complete rejection of products by consumers.
2. Food waste in the global food supply chain is reviewed in relation to the prospects for feeding a population of nine billion by 2050.
3. In the current pandemic situation, damage of perishable goods becomes inevitable and consumption becomes unpredictable. Given below is a summary of how the market of perishable goods has taken a hit during the 2020 COVID-19 pandemic.
4. The sales of vegetables and fruits, both perishable goods have taken a hit as shown below: [1]

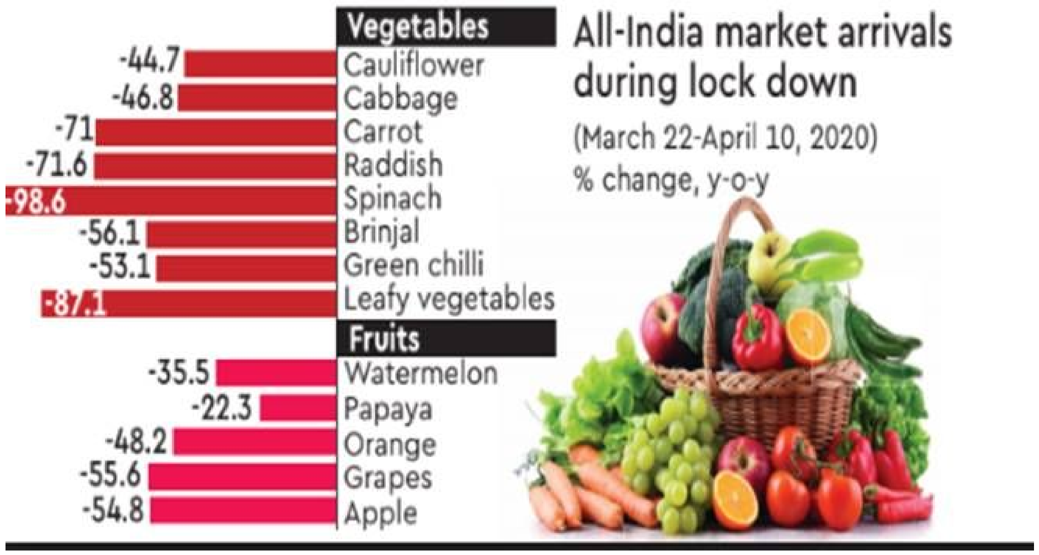


Figure 2.1: All India market arrivals during lockdown

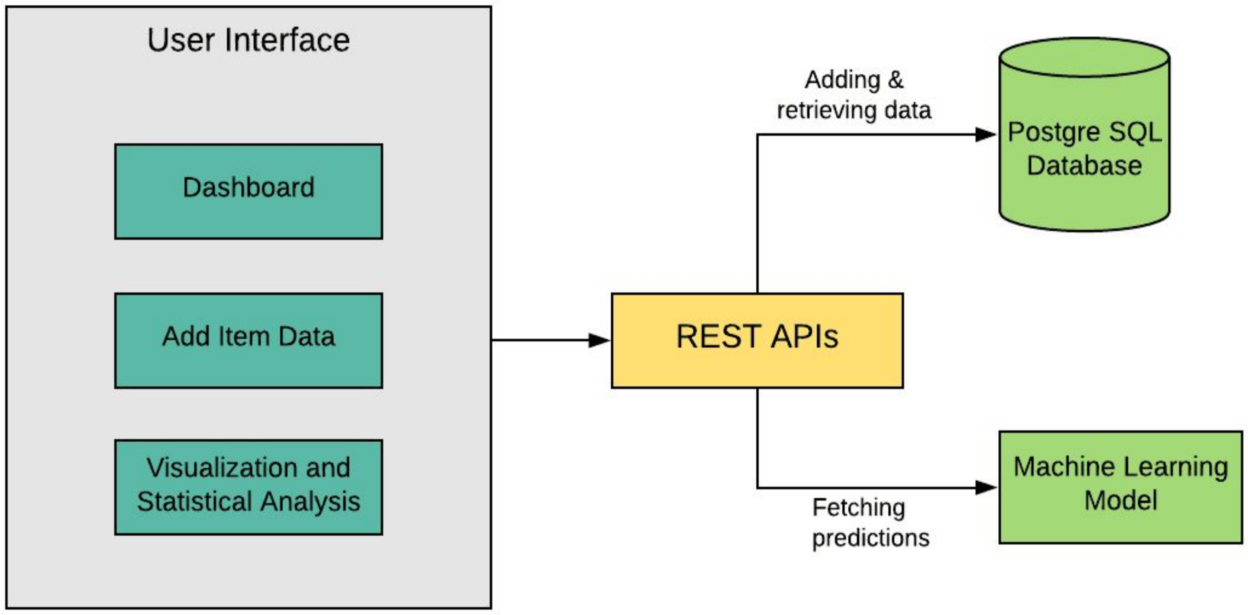
5. According to reports[2], the COVID-19 pandemic struck India when it was the peak harvesting season. As a result, the crops were harvested and stocked in abundance, with lack of consumers coming forward to buy them. Summer vegetables and fruits were picked and stacked in warehouses where a lot of them were damaged.
6. Even during the non-pandemic situations, a survey shows that overstocking and mismanagement of warehouses and transport facilities of perishable goods leads to a loss of $14 billion each year and 40% crops go to waste each year. [3]

2.2 Problem Solution

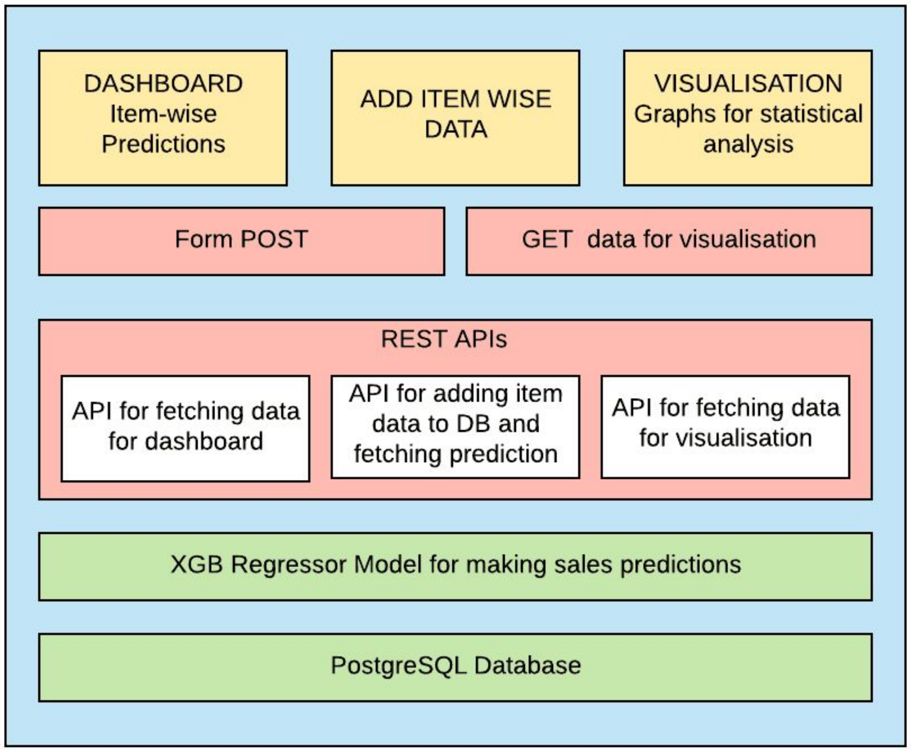
1. An intelligent solution to the above problem is controlling the amount of perishable goods being stocked in the warehouses per week.
2. This puts less pressure on the infrastructure of the warehouses and does not lead to avoidable losses that occur due to overstocking of perishable goods.
3. Our web-app can solve the problem of overstocking of inventory by predicting the sales of each item for the upcoming week based on the current and previous weeks' data.

3. THEORETICAL ANALYSIS

3.1 Block diagram



3.2 Software Design


DASHBOARD Item-wise Predictions
ADD ITEM WISE DATA
VISUALISATION Graphs for statistical analysis
Form POST
GET data for visualisation
REST APIs
API for fetching data for dashboard
API for adding item data to DB and fetching prediction
API for fetching data for visualisation
XGB Regressor Model for making sales predictions
PostgreSQL Database

4. EXPERIMENTAL INVESTIGATIONS

➔ During the course of development of this project, we performed certain experimental investigations in order to finalize the technology stack and a few other things.

➔ During the development of the machine learning model, we were faced with the issue of improving the accuracy of our model. We tried out a few different regressors. We experimented with Linear Regression, Random Forest Regressor and XGBoost Regressor.

➔ Of the models we experimented with, XGBoost gave the best results. Hence we selected XGBoost Regressor for our model.

➔ We obtained an average accuracy of 76% with our model. The accuracy was tested by splitting the dataset into training and test sets per item and comparing the results of predictions for test sets with actual sales values.

➔ Some plots obtained during the testing of the model are as follows:

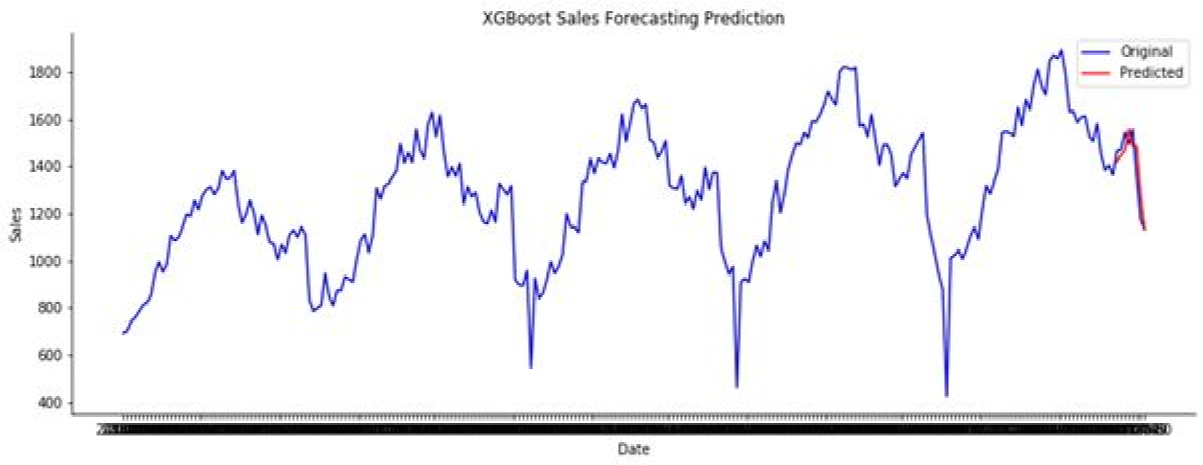


Figure 4.1: Prediction plot 1

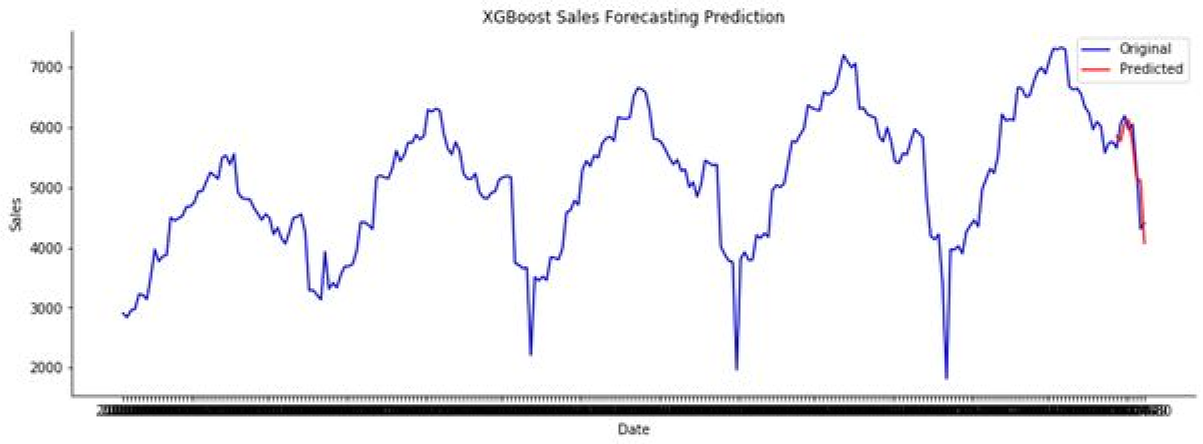
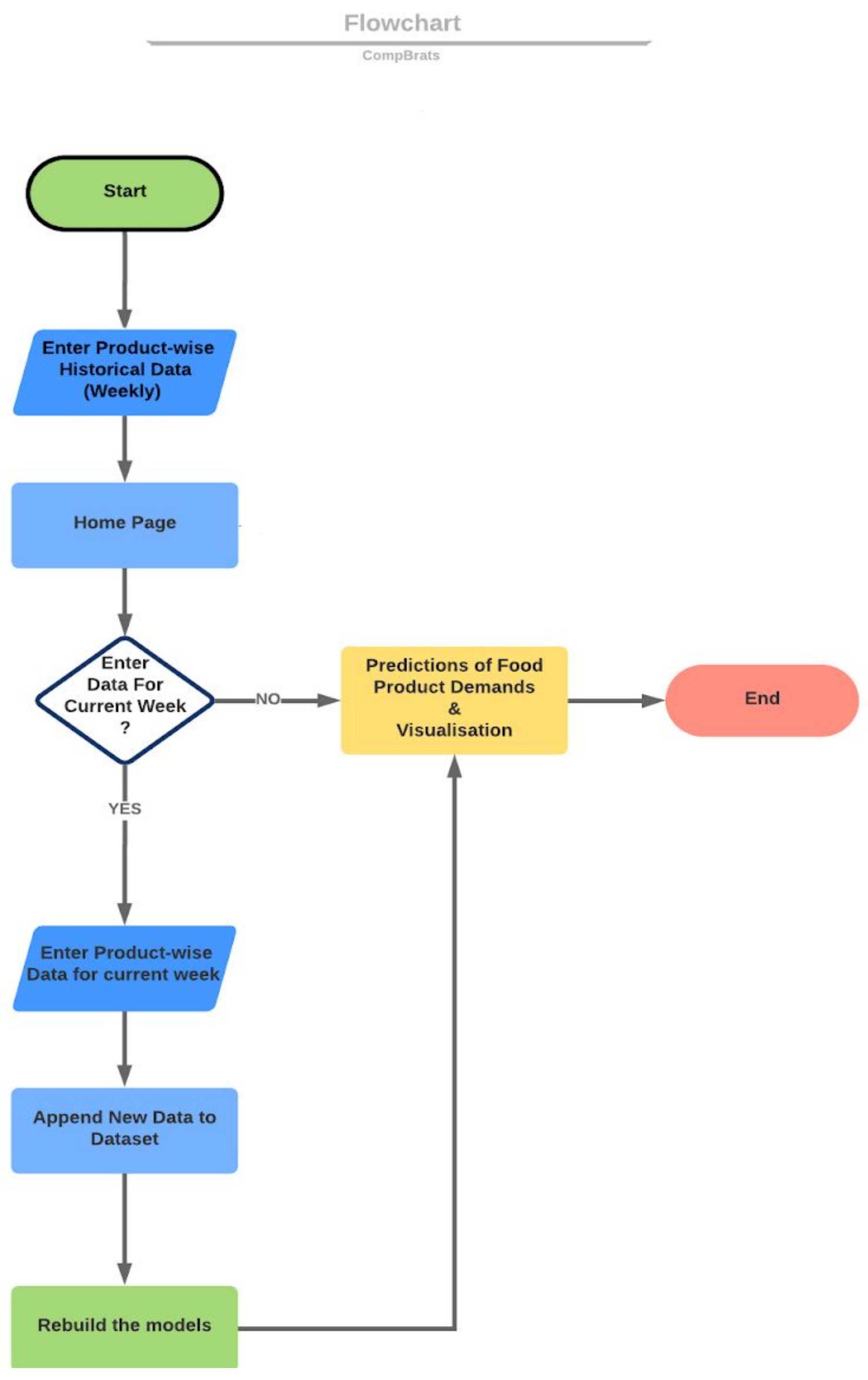


Figure 4.2: Prediction plot 2

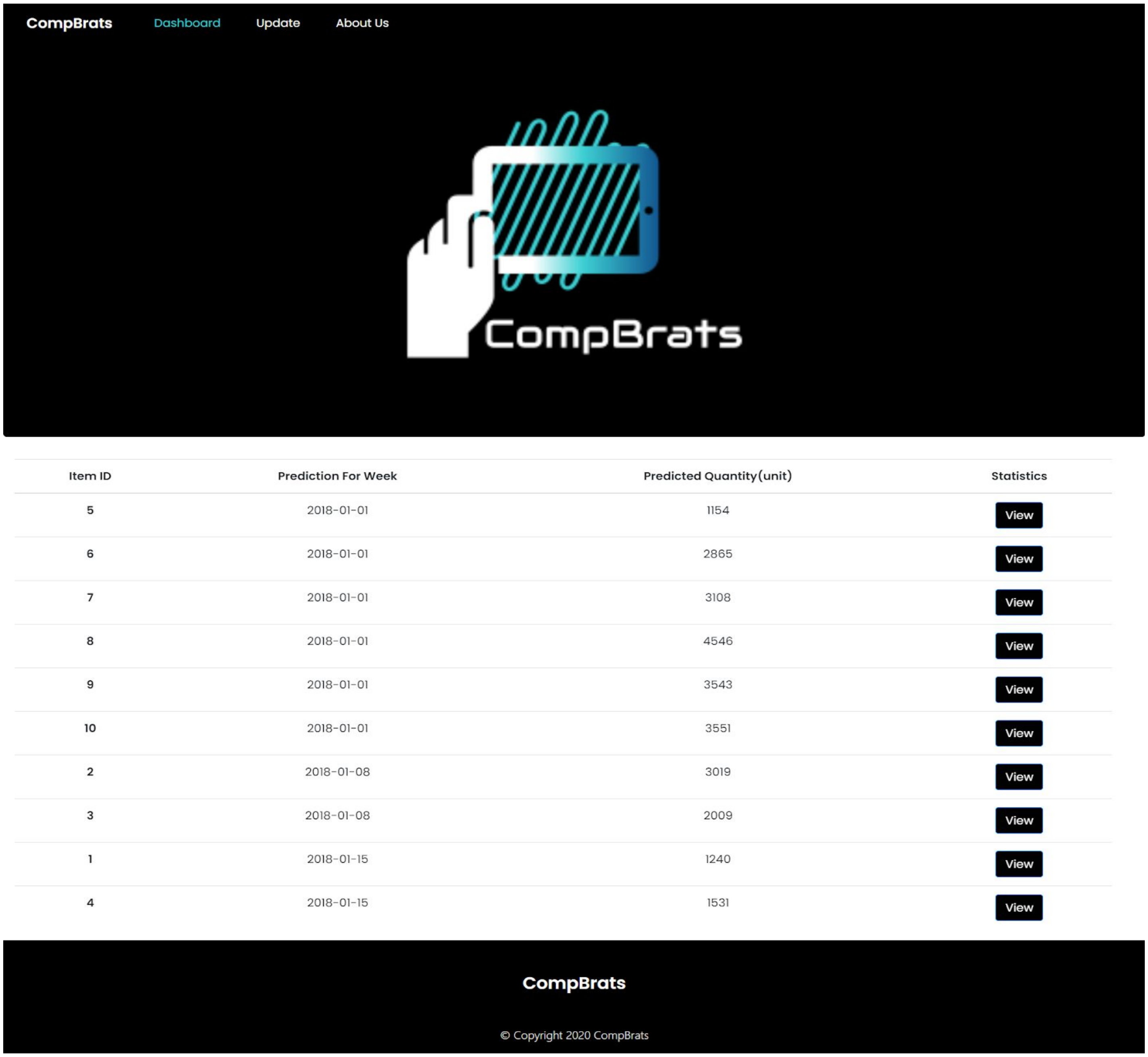
- ➔ While developing the project we realized that it is much easier to operate with an SQL database as compared to connecting to a no-SQL database. Hence we used the most popular and IBM cloud supported database, PostgreSQL.
- ➔ According to the future scope of our project, we found out that Django would be a better option as compared to Flask, as Django would be useful for expanding the scope of our project.
- ➔ We used Bootstrap for the front-end CSS framework which provides better and easy usage.
- ➔ We used Canvas.js for visualization and graphical representations.

5. FLOWCHART



Flowchart
CompBrats
Start
Enter Product-wise Historical Data (Weekly)
Home Page
Enter Data For Current Week ?
NO
Predictions of Food Product Demands & Visualisation
End
YES
Enter Product-wise Data for current week
Append New Data to Dataset
Rebuild the models

6. RESULT

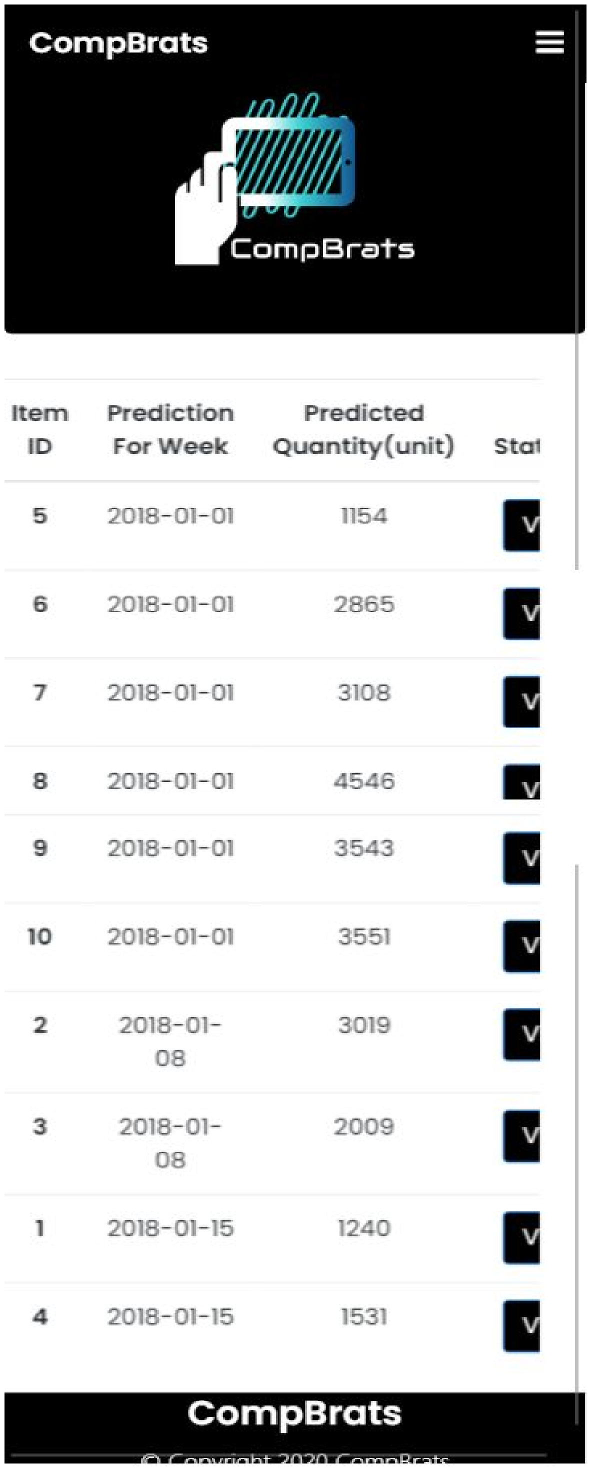


Item ID	Prediction For Week	Predicted Quantity(unit)	Statistics
5	2018-01-01	1154	View
6	2018-01-01	2865	View
7	2018-01-01	3108	View
8	2018-01-01	4546	View
9	2018-01-01	3543	View
10	2018-01-01	3551	View
2	2018-01-08	3019	View
3	2018-01-08	2009	View
1	2018-01-15	1240	View
4	2018-01-15	1531	View

CompBrats

© Copyright 2020 CompBrats

Fig. 1.1 Dashboard Desktop View.

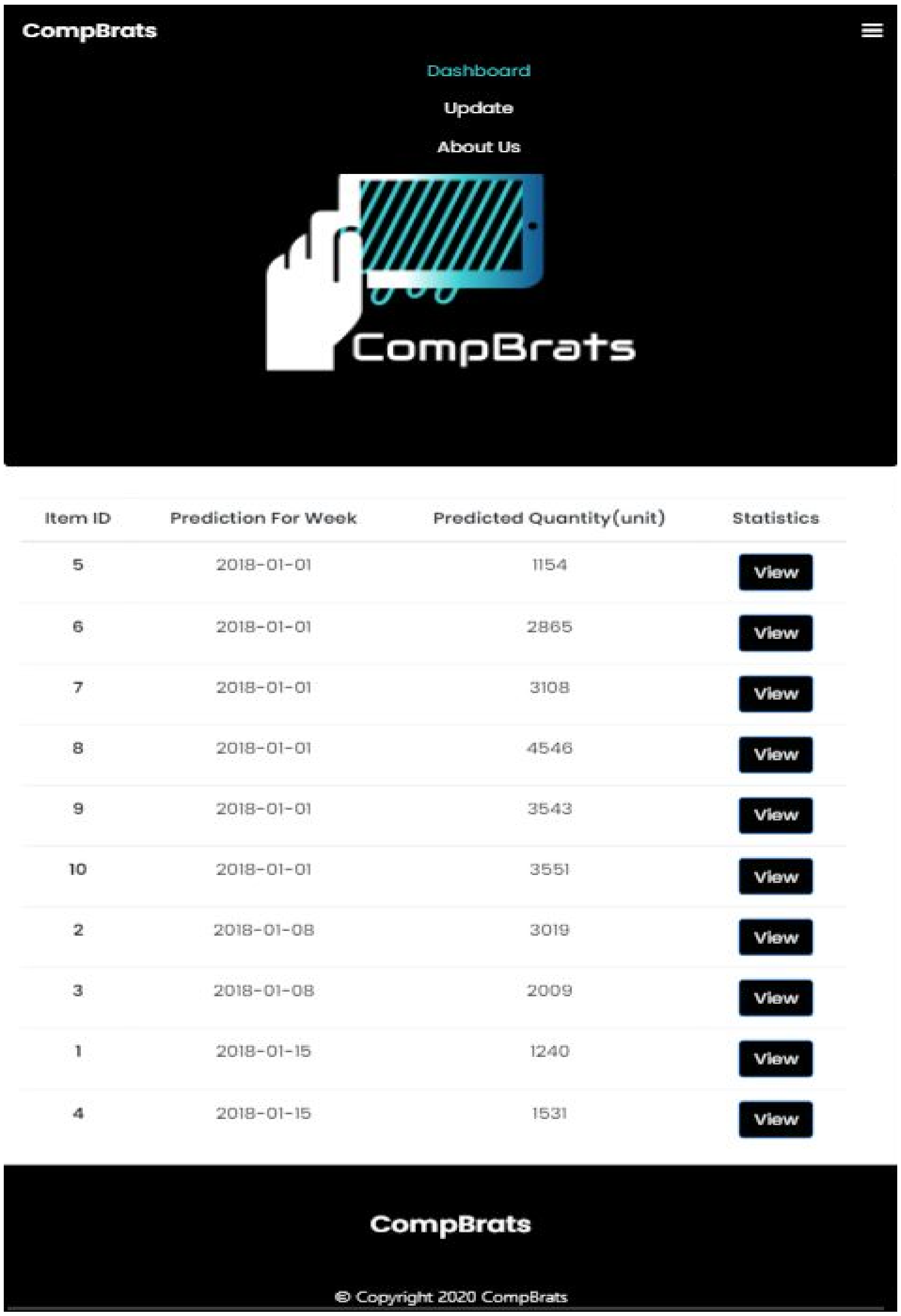


Item ID	Prediction For Week	Predicted Quantity(unit)	Stat
5	2018-01-01	1154	V
6	2018-01-01	2865	V
7	2018-01-01	3108	V
8	2018-01-01	4546	V
9	2018-01-01	3543	V
10	2018-01-01	3551	V
2	2018-01-08	3019	V
3	2018-01-08	2009	V
1	2018-01-15	1240	V
4	2018-01-15	1531	V

CompBrats

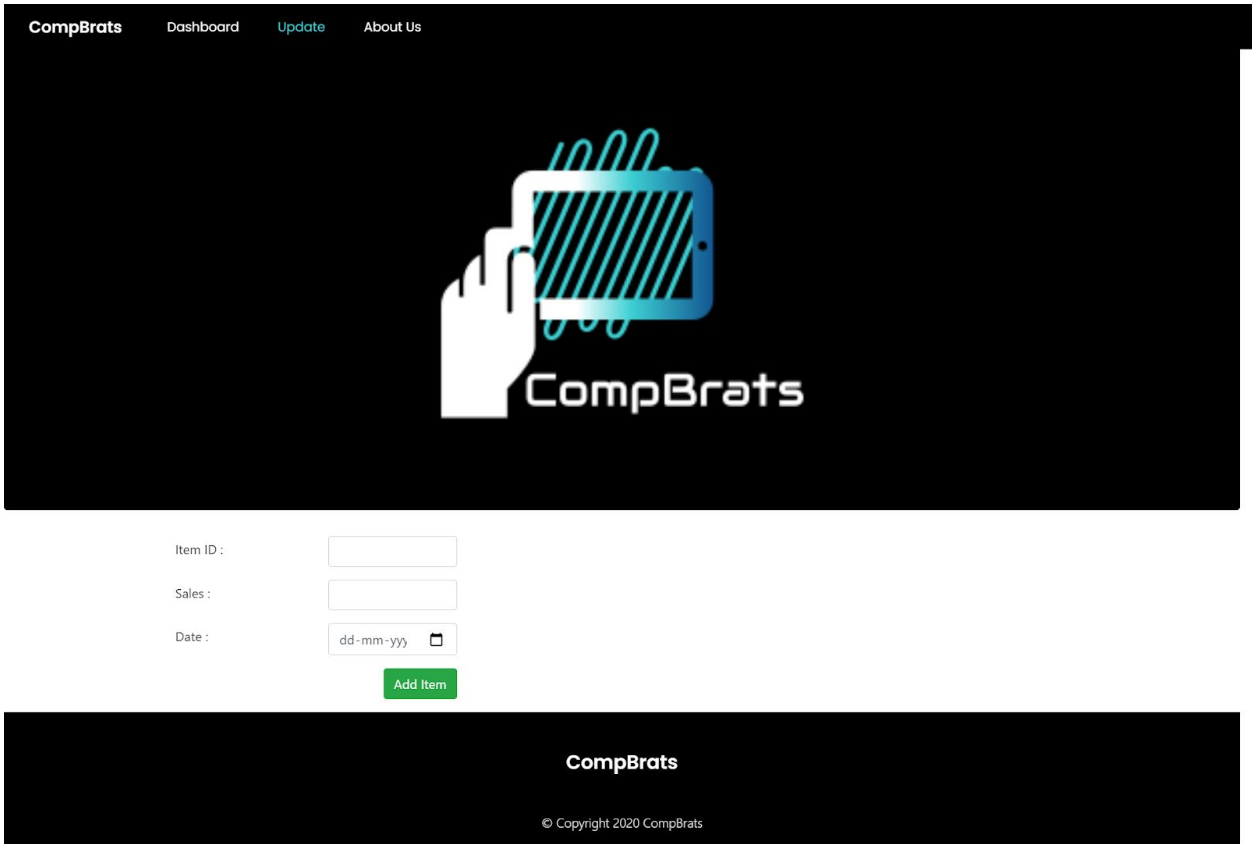
© Copyright 2020 CompBrats

Fig. 1.2 Dashboard Mobile View (scroll left to see view button).



Item ID	Prediction For Week	Predicted Quantity(unit)	Statistics
5	2018-01-01	1154	View
6	2018-01-01	2865	View
7	2018-01-01	3108	View
8	2018-01-01	4546	View
9	2018-01-01	3543	View
10	2018-01-01	3551	View
2	2018-01-08	3019	View
3	2018-01-08	2009	View
1	2018-01-15	1240	View
4	2018-01-15	1531	View

Fig. 1.3 Dashboard Tabled View (Dropdown navigation).



Item ID :

Sales :

Date : dd-mm-yyyy

Add Item

CompBrats

© Copyright 2020 CompBrats

Fig. 2.1 Update Sales Data Desktop View.

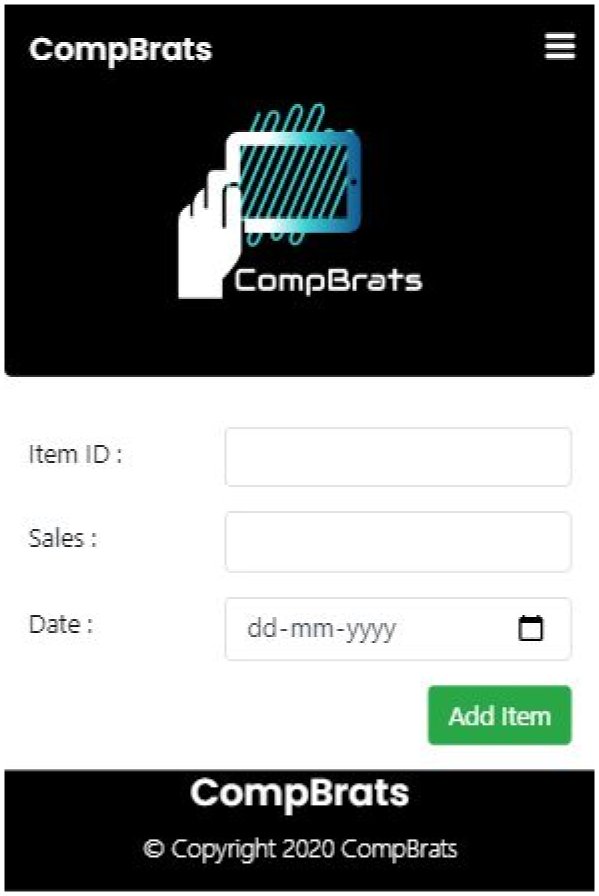


Fig. 2.1 Update Sales Data Mobile View.

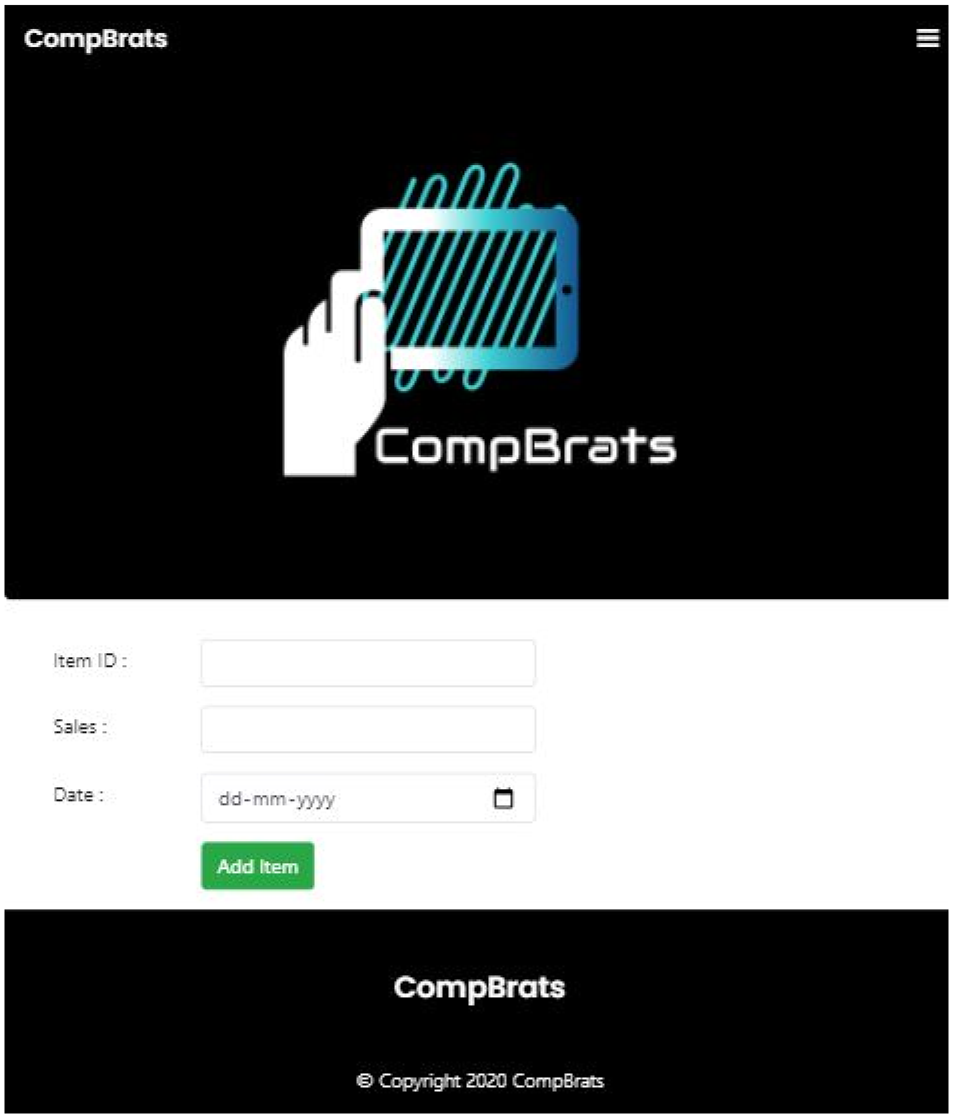


Fig. 2.3 Update Sales Data Tablet View.

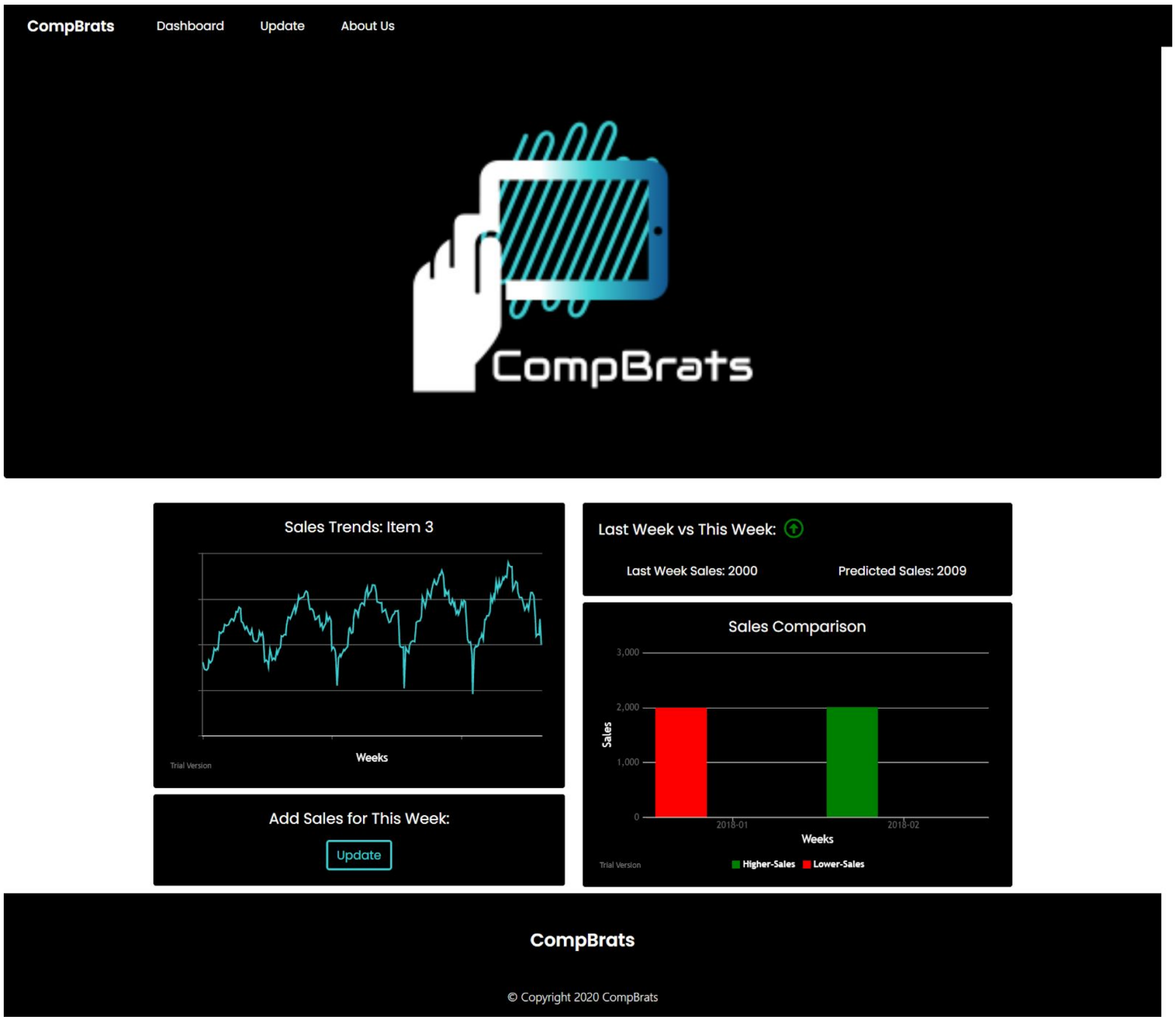


Fig. 2.3 Analytics Tablet View.

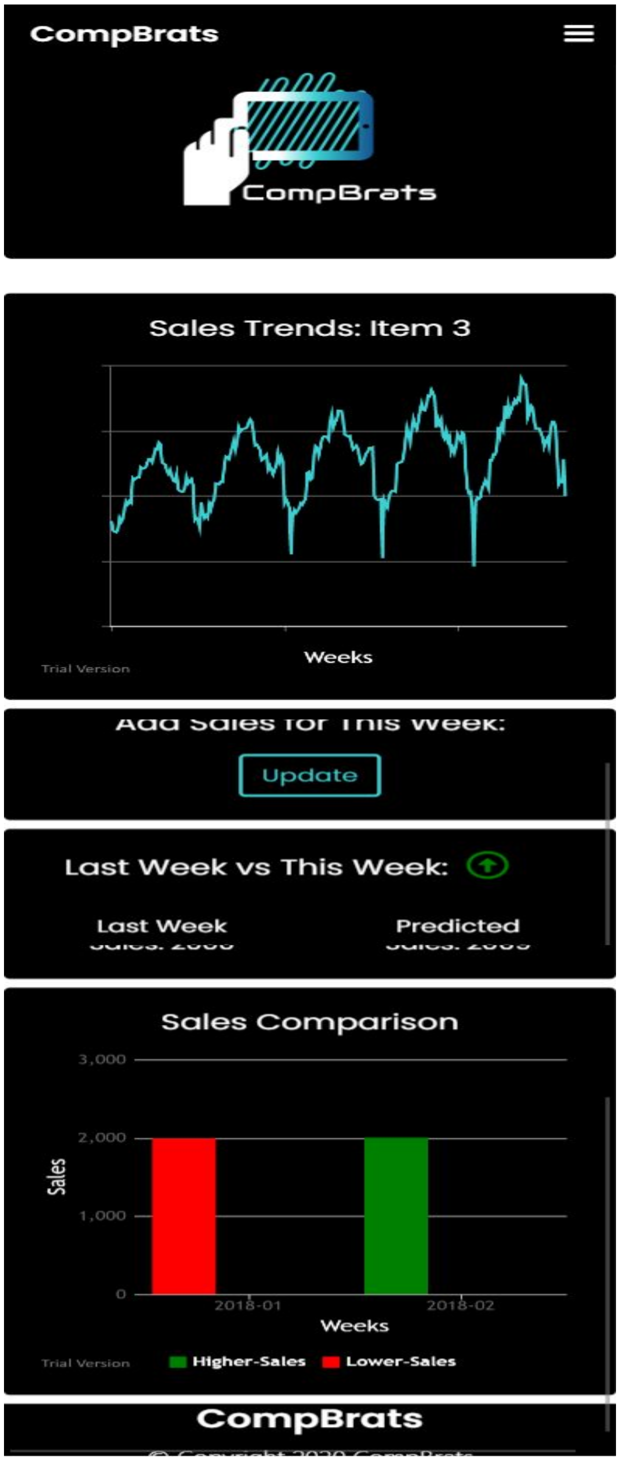


Fig. 2.3 Analytics Mobile View.

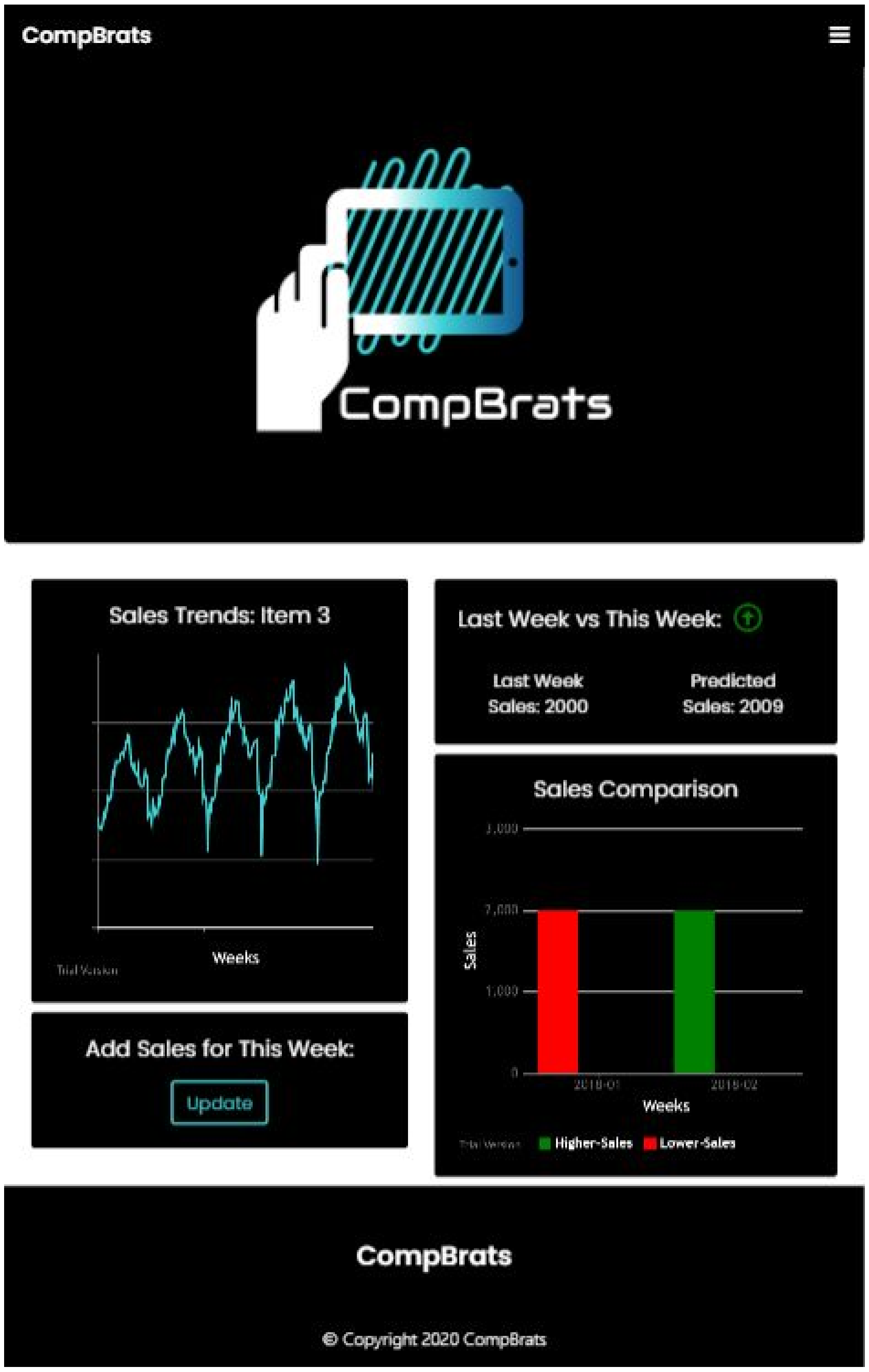
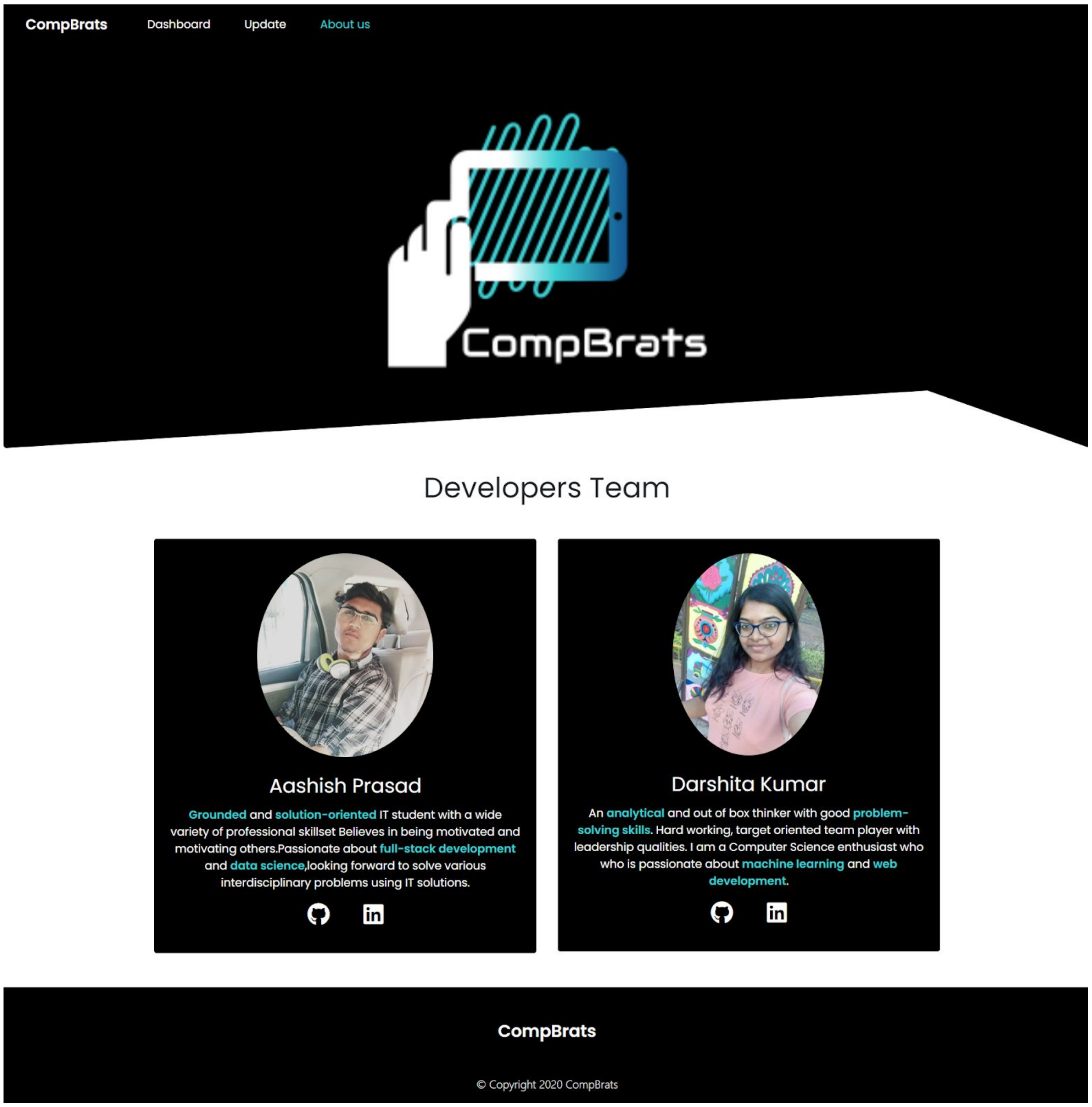


Fig. 2.3 Analytics Tablet View.



CompBrats Dashboard Update About us

Developers Team

Aashish Prasad

Grounded and solution-oriented IT student with a wide variety of professional skillset Believes in being motivated and motivating others.Passionate about full-stack development and data science,looking forward to solve various interdisciplinary problems using IT solutions.

Darshita Kumar

An analytical and out of box thinker with good problem-solving skills. Hard working, target oriented team player with leadership qualities. I am a Computer Science enthusiast who who is passionate about machine learning and web development.

CompBrats

© Copyright 2020 CompBrats

Fig. 2.3 About our team.

7. ADVANTAGES AND DISADVANTAGES

7.1 Advantages

1. It is an open-platform for all food delivery service providers.
   - The web app can be used by local food product delivery services to manage their warehouses.
   - Any provider can have an account on the web app and use its features hence making it an open-platform and providing local services an opportunity to boost their business,in competition to giants like Amazon grocery,Big Basket,etc.
2. To detect the increase in demand of a particular category of product in order to promote similar products of the same category that are not selling well.
   Example:
   - During the period of quarantine, there has been a trend of baking cakes among the youngsters. Thus, there has been a spike in the demand of Pillsbury baking products.
   - If this spike in demand is noticed, and the demand for the upcoming weeks is predicted, the food service provider can stock baking products of different companies like Weikfield along with Pillsbury.
   - It can also boost the sales of say Cookwell baking products, which were not in demand, through promotions.

3. Visualization of sales: The responsive web-app enables the warehouse owner to visualize the sales of an item over time and perform a statistical analysis of the same.

7.2 Disadvantages

1. Inability of the model to predict sudden spikes in sale of goods
   Example:

- Consider the current pandemic situation of COVID-19, when the fear of virus became evident among the population people started hoarding food products, considering the possible lockdown.
- This hoarding started at a slow rate but eventually led to shortage of food products.Our web app cannot predict this hoarding situation, at least for the first few weeks.
- But once the model gets trained with a lot of out-of-sample data, the model will be able to predict the sales of items even with a sudden spike in sales.

8. APPLICATIONS

1. The web-app is to be used for prediction of sales of perishable goods for the upcoming week so that the inventory can be stocked accordingly.
2. The sales trends can be visualized item wise and a statistical comparative analysis can be obtained between the current week

sales and the sales of the upcoming week (predicted).

3. The sales prediction can be obtained from anywhere as our web-app is a responsive, cross-browser solution.

9. CONCLUSION

The problem of overstocking and understocking of items by warehouse managers leads to losses and adversely affects farmers too. The web-app developed by us solves these problems as the sales for the upcoming week is predicted and a statistical analysis of past sales is graphically represented.

10. FUTURE SCOPE

1. The web app can be integrated with the ERPs and CRMs to make it a complete business planning platform.
2. The warehouse manager may enter historical data in the form of a CSV.
3. The warehouse manager may be able to add new items and obtain store wise sales predictions for each item.

11. BIBLIOGRAPHY

References:

Links:

[1] https://royalsocietypublishing.org/doi/10.1098/rstb.2010.0126

[2]

http://www.fao.org/in-action/food-for-cities-programme/news/detail/en/c/1272232/

[3] https://numadic.com/blog/poor-cold-chain-logistics-waste-40-of-crops-worth-over-14-billion-each-year/

APPENDIX: Source Code

ML Model code:

```
import xgboost as xgb

import pandas as pd

import numpy as np

import datetime as dt

from sklearn.preprocessing import MinMaxScaler

from csv import writer

from datetime import datetime




def callingFunction(item_no, date1, sales1, date2):

   def getWeekYear(data):

       data.date = data.date.apply(lambda x:
str(str(x)[0:4])+"-"+str(dt.date(int(str(x)[0:4]), int(str(x)[5:7]),
int(str(x)[8:10])).isocalendar()[1]).zfill(2))
```

```
    data = data.groupby(['item', 'date'])['sales'].sum().reset_index()

    return data


weekly_s_collection = {}

model_df = {}

weekly_collection = {}


def generate_model_df():

    for i in range(1, 11):

        weekly_collection[i] = weekly_data[weekly_data.item == i]


    def get_diff(data):

        data['sales_diff'] = data.sales.diff()

        data = data.dropna()

        return data


    for i in range(1, 11):

        weekly_s_collection[i] = get_diff(weekly_collection[i])


    def generate_supervised(data, item_no):

        supervised_df = data.copy()


        # create column for each lag

        for i in range(1, 53):
            col = 'lag_' + str(i)
            supervised_df[col] = supervised_df['sales_diff'].shift(i)

        # drop null values.
        supervised_df = supervised_df.dropna().reset_index(drop=True)
        return supervised_df

    for i in range(1, 11):
        model_df[i] = generate_supervised(weekly_s_collection[i], i)

# Below function is only for appending week
def appendWeek(item_no, date, sales):
    weekly_data.loc[len(weekly_data)] = [item_no, date, sales]
    generate_model_df()

def makeModel():
    # train test split
    def tts(data):
        data = data.drop(['item', 'sales', 'date'], axis=1)
        train, test = data[:-1].values, data[-1:].values

        return train, test

    # scale data
```

```
       def scale_data(train_set, test_set):

           scaler = MinMaxScaler(feature_range=(-1, 1))

           scaler = scaler.fit(train_set)



           train_set = train_set.reshape(

               train_set.shape[0], train_set.shape[1])

           train_set_scaled = scaler.transform(train_set)



           test_set = test_set.reshape(test_set.shape[0],
test_set.shape[1])

           test_set_scaled = scaler.transform(test_set)



           X_train, y_train = train_set_scaled[:,1:], train_set_scaled[:,
0:1].ravel()

           X_test, y_test = test_set_scaled[:, 1:], test_set_scaled[:,
0:1].ravel()



           return X_train, y_train, X_test, y_test, scaler



       # unscaling predictions, X_test, scaler_object

       def undo_scaling(y_pred, x_test, scaler_obj, lstm=False):

           y_pred = y_pred.reshape(y_pred.shape[0], 1, 1)



           if not lstm:

               x_test = x_test.reshape(x_test.shape[0], 1,
x_test.shape[1])


           pred_test_set = []

           for index in range(0, len(y_pred)):

               pred_test_set.append(np.concatenate(

                   [y_pred[index], x_test[index]], axis=1))


           pred_test_set = np.array(pred_test_set)

           pred_test_set = pred_test_set.reshape(

               pred_test_set.shape[0], pred_test_set.shape[2])


           pred_test_set_inverted = scaler_obj.inverse_transform(

               pred_test_set)


           return pred_test_set_inverted


       # prediction function

       def predict_df(unscaled_predictions, original_df):

           result_list = []

           sales_dates = list(original_df[-2:].date)

           act_sales = list(original_df[-2:].sales)


           for index in range(0, len(unscaled_predictions)):

               result_dict = {}
               result_dict['pred_value'] = int(
                   unscaled_predictions[index][0] + act_sales[index])
               result_dict['date'] = sales_dates[index+1]
               result_list.append(result_dict)

           df_result = pd.DataFrame(result_list)
           return df_result

       def regressive_model(train_data, test_data, model, model_name):

           X_train, y_train, X_test, y_test, scaler_object = scale_data(
               train_data, test_data)
           mod = model
           mod.fit(X_train, y_train)
           predictions = mod.predict(X_test)

           # Undo scaling to compare predictions against original data
           original_df = weekly_s_collection[item_no]
           unscaled = undo_scaling(predictions, X_test, scaler_object)
           unscaled_df = predict_df(unscaled, original_df)
           return unscaled_df, original_df, mod

       train, test = tts(model_df[item_no])
       unscaled_df, original_df, model = regressive_model(train, test,
xgb.XGBRegressor(

           n_estimators=100, learning_rate=0.2,
objective='reg:squarederror'), 'XGBoost')

       return unscaled_df


   # write the new actual sales to csv

           csv_writer.writerow(

               [datetime.strptime(date1, '%Y-%m-%d').date(), 0, item_no,
sales1])


   update_csv()

   ans_date = date2

   df = pd.read_csv("D:\IBM_Hack_2020/Warehouse_train_copy.csv")

   date1 =
str(str(date1)[0:4])+"-"+str(dt.date(int(str(date1)[0:4]),int(str(date1)[5
:7]), int(str(date1)[8:10])).isocalendar()[1]).zfill(2)

   date2 =
str(str(date2)[0:4])+"-"+str(dt.date(int(str(date2)[0:4]),int(str(date2)[5
:7]), int(str(date2)[8:10])).isocalendar()[1]).zfill(2)


   weekly_data = getWeekYear(df)

   generate_model_df()

   appendWeek(item_no, date2, 0)


   unscaled_df = makeModel()
```

```
dict1 = {

    "item_no": item_no,

    "date": ans_date,

    "sales": int(unscaled_df.iloc[0]['pred_value'])

}



return dict1
```