



## **BEACONHOUSE NATIONAL UNIVERSITY**

**Wasail**

**PRJ-F23/333**

### **DESIGN DOCUMENT**

**EXTERNAL SUPERVISOR**

**Hamza Zafar**

**INTERNAL SUPERVISOR**

**Huda Sarfraz**

### **GROUP MEMBERS**

|                           |                  |
|---------------------------|------------------|
| <b>Fatima Ali Tirmizi</b> | <b>F2020-718</b> |
| <b>Fizza Adeel</b>        | <b>F2020-336</b> |
| <b>Irtaza Ahmed Khan</b>  | <b>F2020-153</b> |
| <b>Malaika Sultan</b>     | <b>F2020-661</b> |

**SCHOOL OF COMPUTER & IT**

## Table of Contents

|   |           |
|---|-----------|
| <b>Introduction.....</b>                            | <b>3</b>  |
| <b>Development Tools.....</b>                       | <b>3</b>  |
| Programming Languages.....                          | 3         |
| Machine Learning.....                               | 3         |
| Cloud Services.....                                 | 3         |
| Web Development.....                                | 3         |
| Data Storage.....                                   | 4         |
| Object Relational Mapper.....                       | 4         |
| Mobile Development.....                             | 4         |
| Version Control.....                                | 4         |
| Project Management.....                             | 4         |
| Collaboration.....                                  | 4         |
| <b>System Architecture.....</b>                     | <b>4</b>  |
| System Context Diagram.....                         | 4         |
| Container Diagram.....                              | 5         |
| Component Diagram.....                              | 6         |
| Class Diagram.....                                  | 7         |
| <b>Data Design.....</b>                             | <b>9</b>  |
| <b>Sequence Diagrams.....</b>                       | <b>10</b> |
| Order Recommendation.....                           | 10        |
| Order Tracking.....                                 | 11        |
| All Products Search.....                            | 12        |
| OTP Code Generation and Delivery.....               | 13        |
| Add Vendor to Vendor List.....                      | 14        |
| <b>API Design.....</b>                              | <b>15</b> |
| <b>ML Design.....</b>                               | <b>16</b> |
| Data Collection.....                                | 16        |
| Local Pharmacy Dataset.....                         | 16        |
| Corporación Favorita Grocery Sales Forecasting..... | 17        |
| Instacart Market Basket Analysis.....               | 18        |
| Feature Engineering.....                            | 19        |
| Local Pharmacy Dataset.....                         | 19        |
| Corporación Favorita Grocery Sales Forecasting..... | 21        |
| Model Shortlisting.....                             | 25        |
| Project Part I.....                                 | 25        |
| Random Forest.....                                  | 25        |
| XGBoost.....  | 25        |
| Prophet (Facebook).....                             | 25        |
| RNN (LSTM/GRU).....                                 | 26        |
| Project Part II.....                                | 26        |
| LightGBM.....                                       | 26        |
| N-BEATS.....  | 26        |
| DeepAR (Amazon).....                                | 26        |

|   |           |
|---|-----------|
| Temporal Fusion Transformer (Google)..... | 26        |
| Training and Testing.....                 | 26        |
| Deployment.....                           | 26        |
| <b>Conclusion.....</b>                    | <b>27</b> |
| <b>References.....</b>                    | <b>27</b> |

## **Introduction**

The design document presented here outlines a comprehensive approach to developing a demand forecasting system for grocery stores and creating an environment to aid the communication between them and the vendors. Leveraging a diverse set of technologies, the system integrates machine learning models with a robust backend infrastructure and a user-friendly front end. The document delves into various components such as development tools, programming languages, data storage, machine learning frameworks, and cloud services. It also provides detailed insights into the system architecture, data design, and the integration of machine learning for demand forecasting. With a focus on feature engineering and model selection, the document showcases the meticulous process involved in preparing datasets, cleaning, and transforming them for accurate predictions.

## **Development Tools**

### **Programming Languages**

- Python
- JavaScript
- Dart

### **Machine Learning**

- TensorFlow
- PyTorch
- Scikit-Learn
- Prophet (Facebook)
- LightGBM
- N-BEATS
- DeepAR (Amazon)
- Temporal Fusion Transformer (Google)

### **Cloud Services**

- Digital Ocean

### **Web Development**

- Front-End
  - HTML
  - CSS
  - Bootstrap
  - React
- Back-End
  - Node.js
  - Express.js
  - Flask

## Data Storage

- MySQL

## Object Relational Mapper

- Sequelize

## Mobile Development

- Flutter

## Version Control

- GitHub

## Project Management

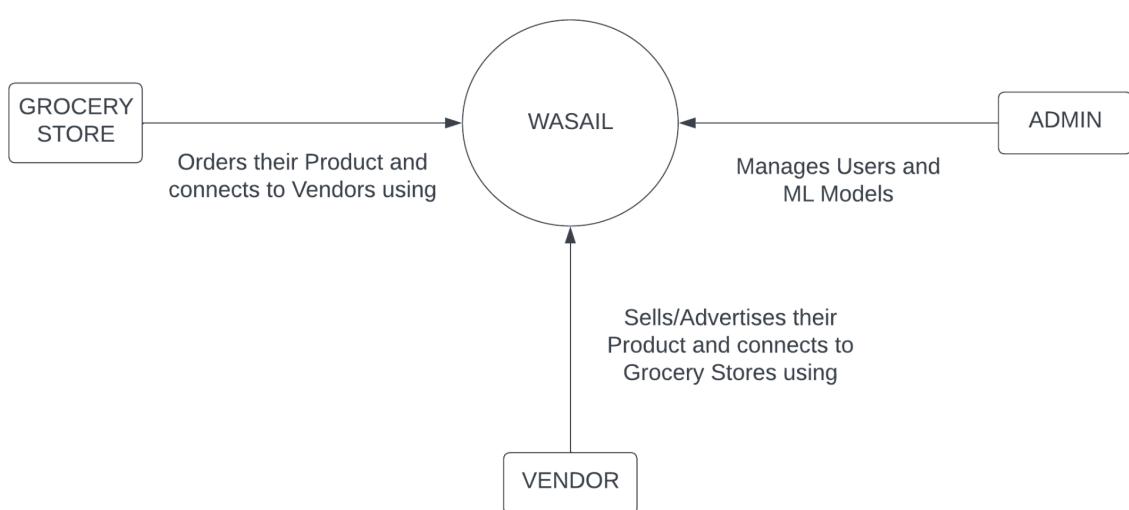
- Jira

## Collaboration

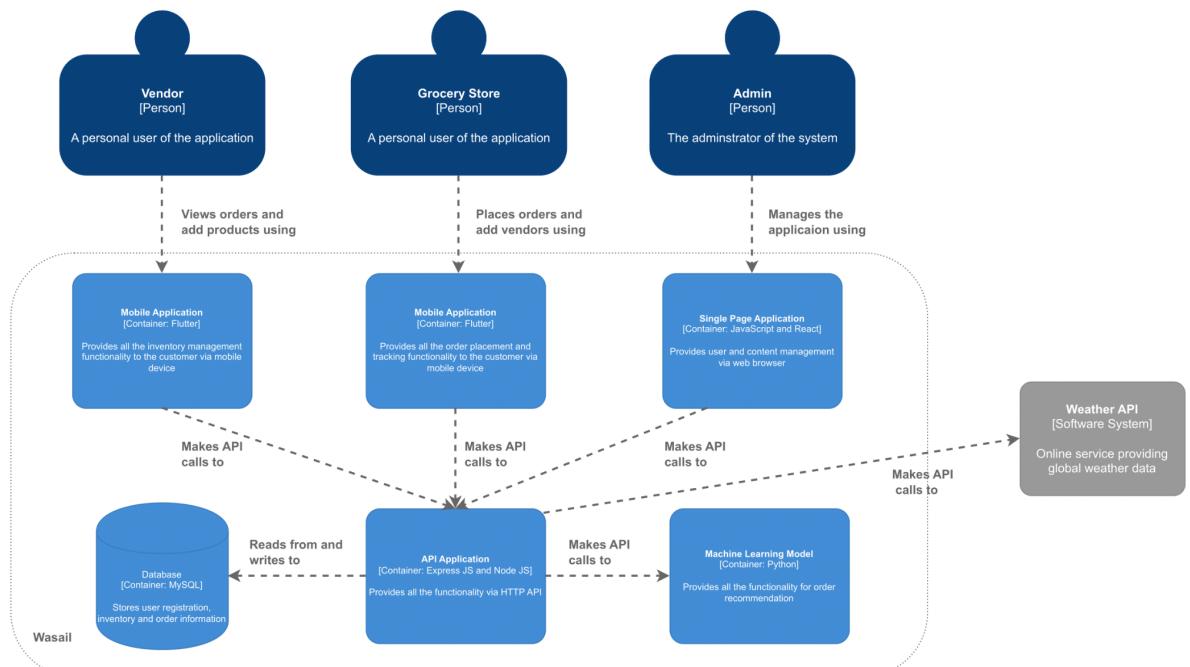
- Discord

## System Architecture

### System Context Diagram



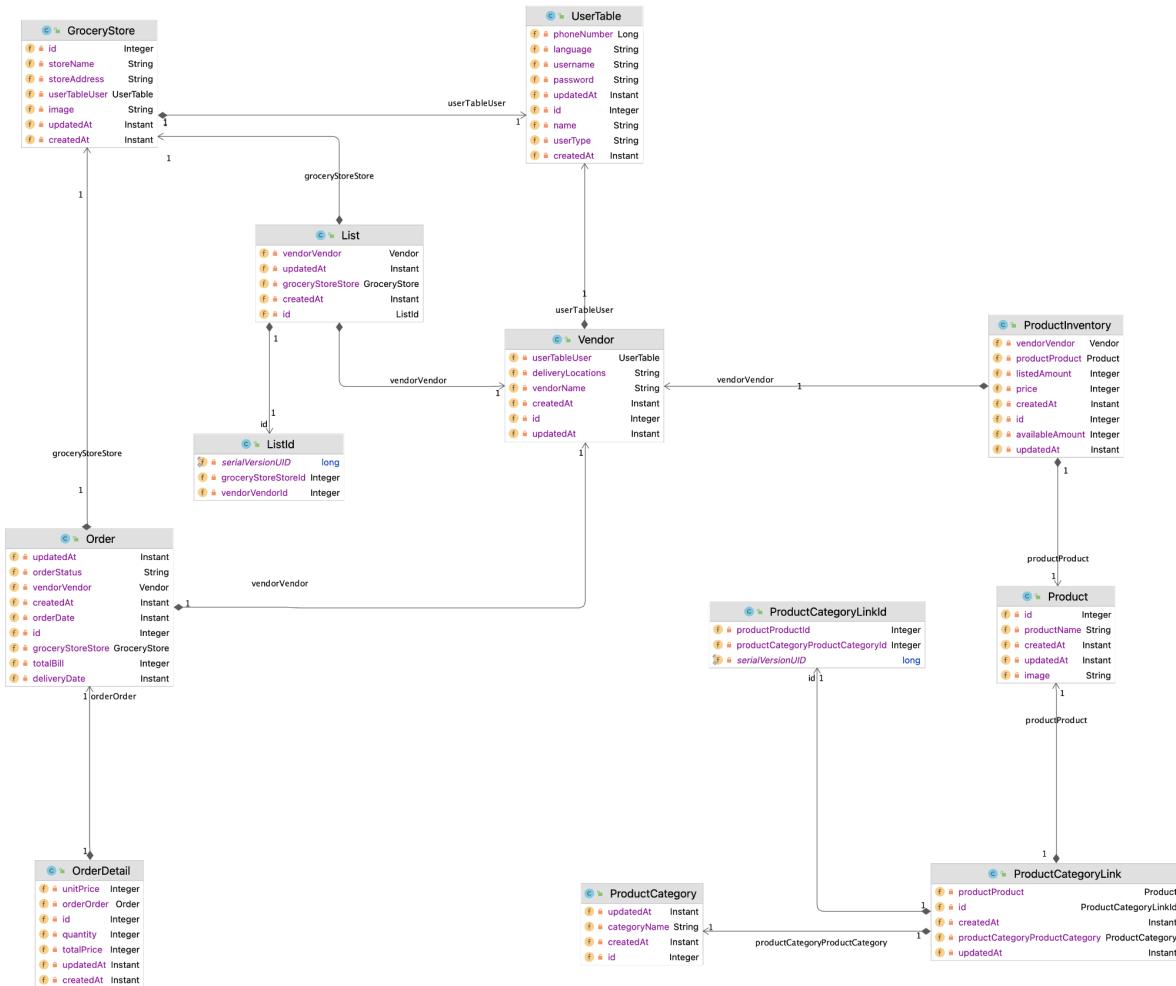
## Container Diagram



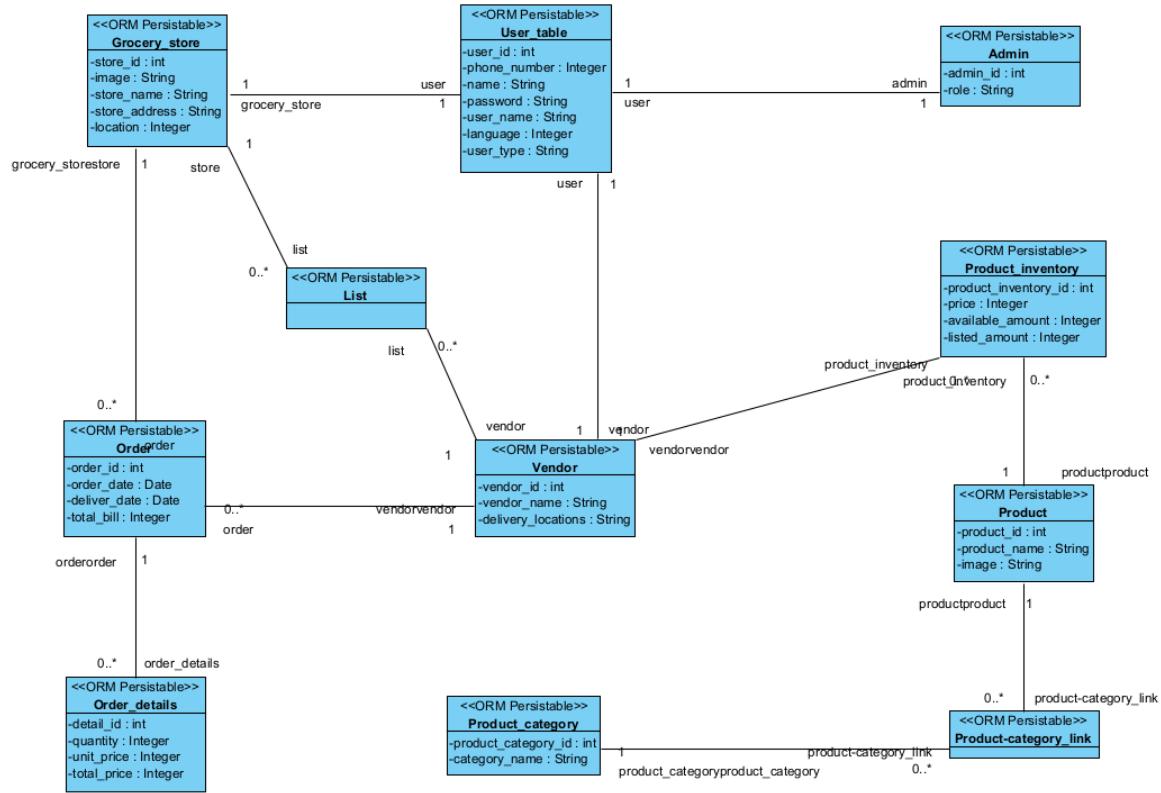
## **Component Diagram**

## Class Diagram

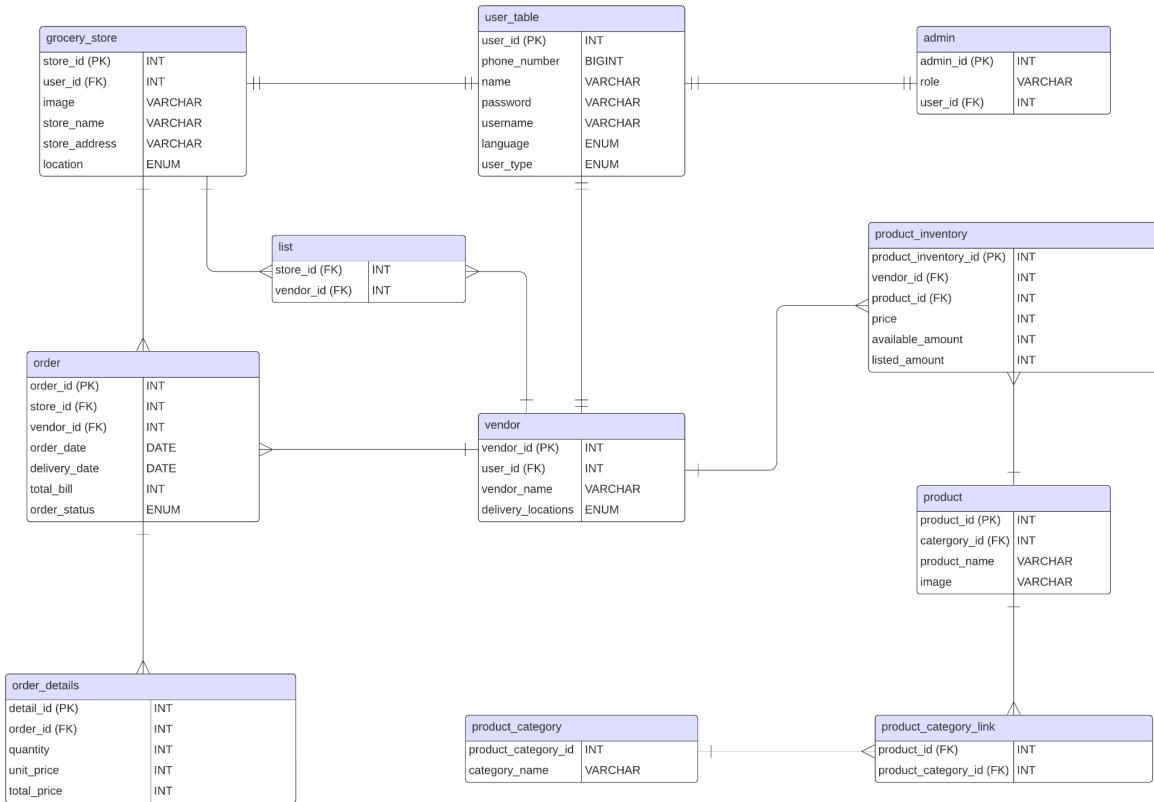
(Automatically generated using JetBrains)



(Automatically generated using VisualParadigm)



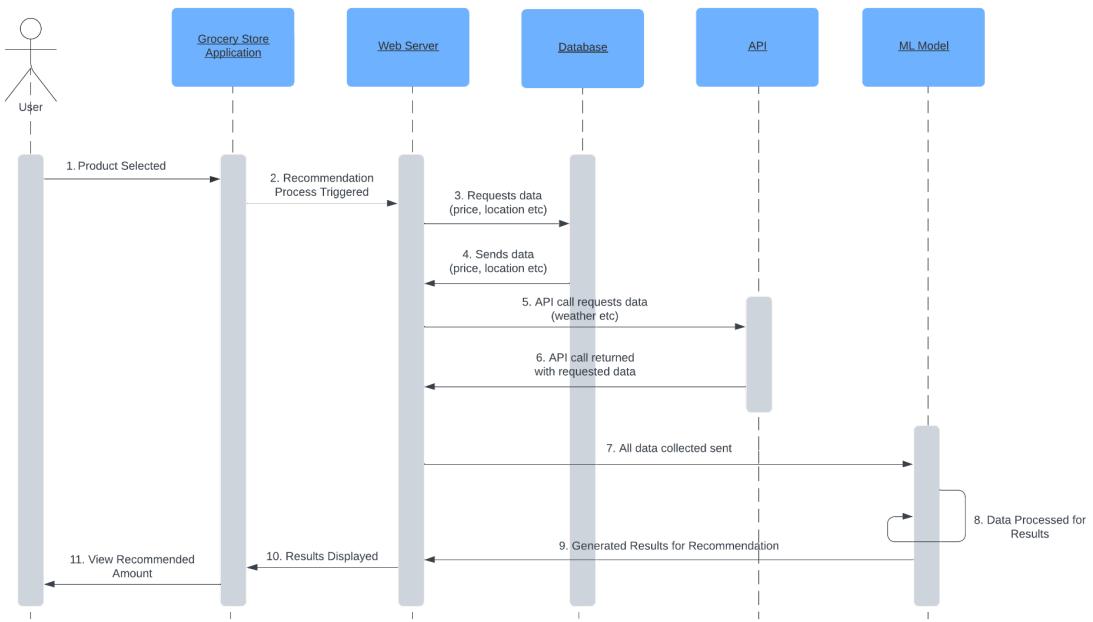
## Data Design



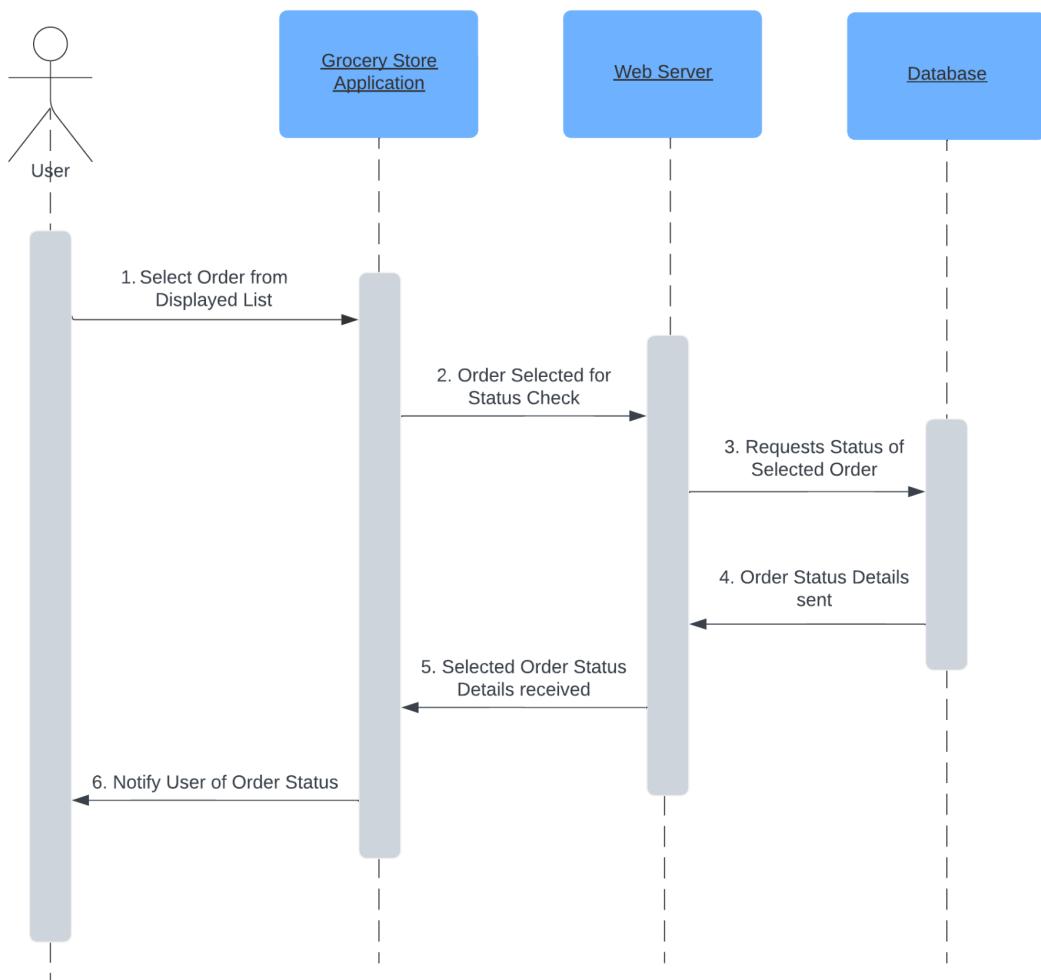
When designing the database diagram, it was mainly divided into three sections; user management, inventory management, and order management. Under user management the tables that were created were user table, vendor, grocery store and admin. All the common attributes including phone number, name, password, user name, language and user type were in the user table which had a one to one relation with grocery store, vendor and admin. Since a grocery store and vendor had a many to many relationship (one grocery store can have many vendors and vice versa), it required a pivot table (called lists), since a many to many relation can not be made in the diagram. Next, under the inventory management section, the tables that were created were product (a table from which the vendor will add products to their own inventory), product inventory (the vendors' own inventory), product category (the table that will contain the categories of the product) and product category link. Similar to the vendor and grocery store table, the product and product category tables also have a many to many relation and hence requires a pivot table. Lastly, under order management, order and order details were the two tables that were created.

## Sequence Diagrams

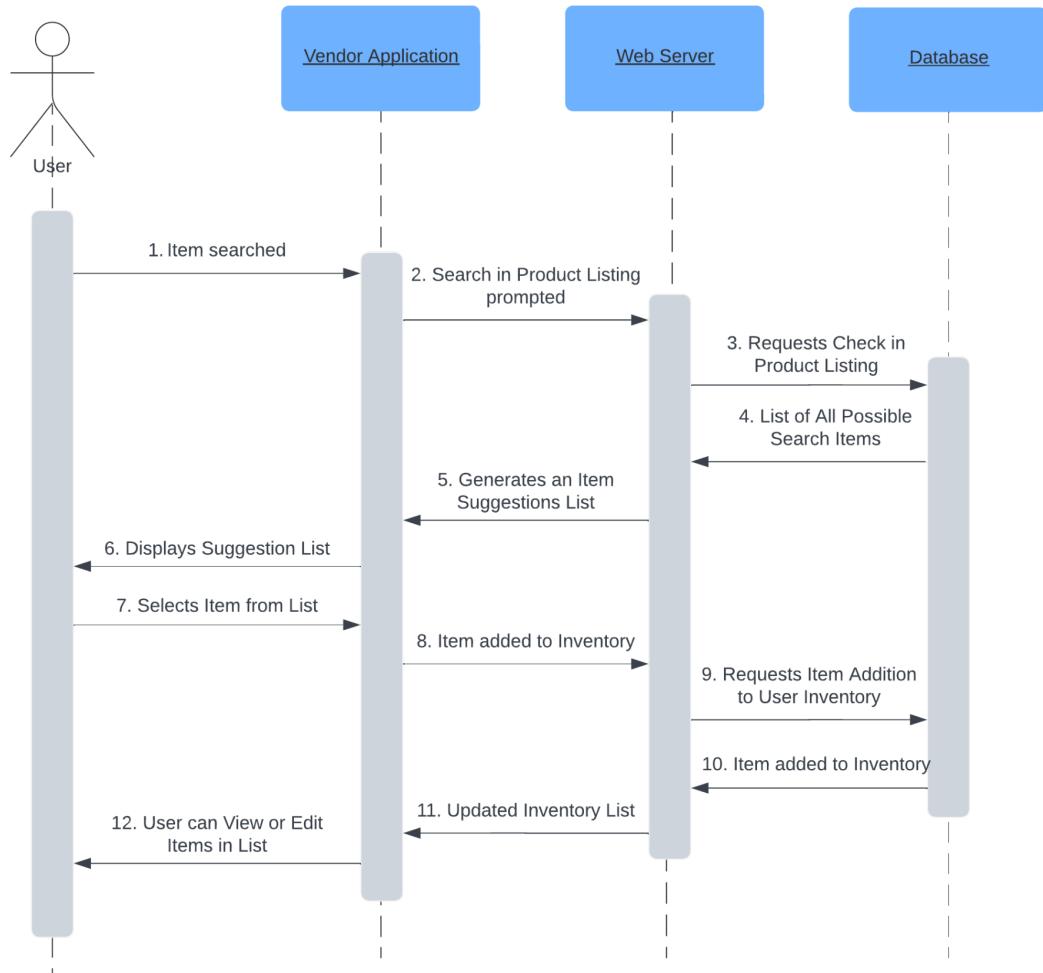
### Order Recommendation



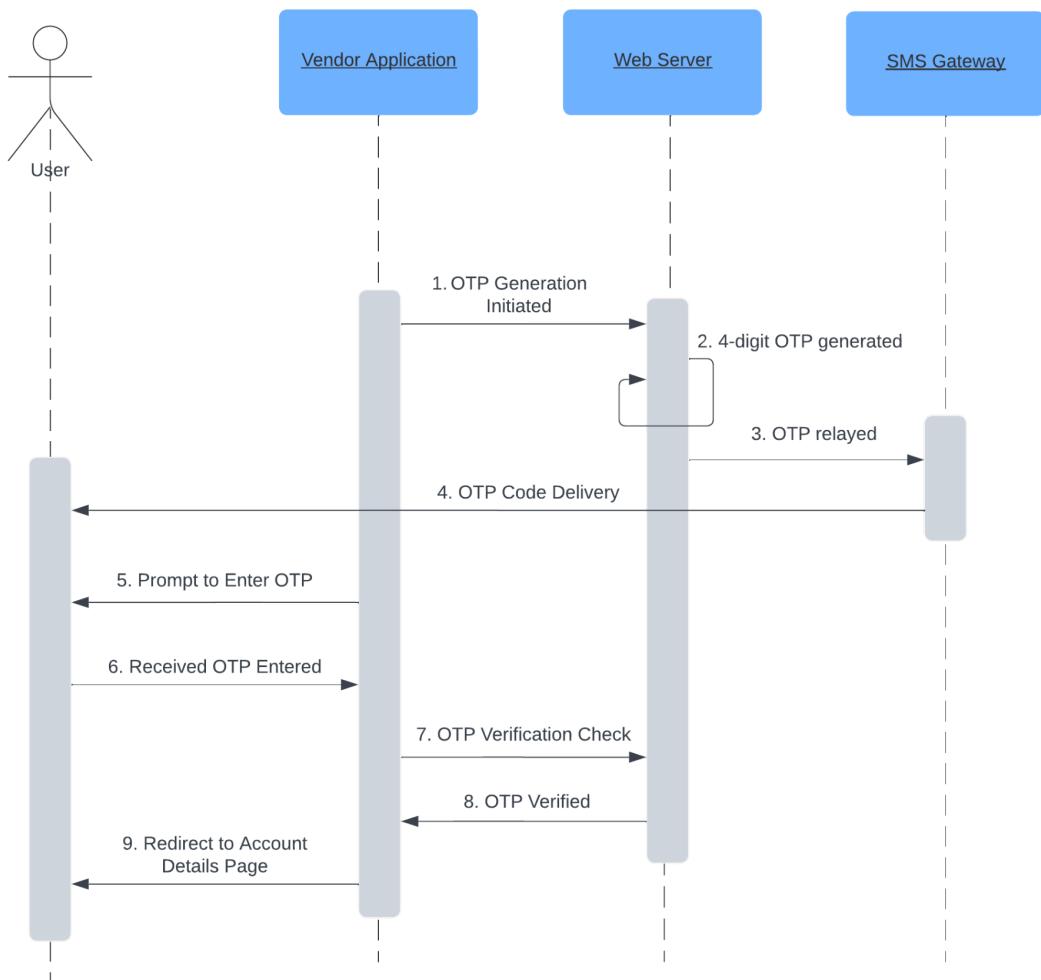
## Order Tracking



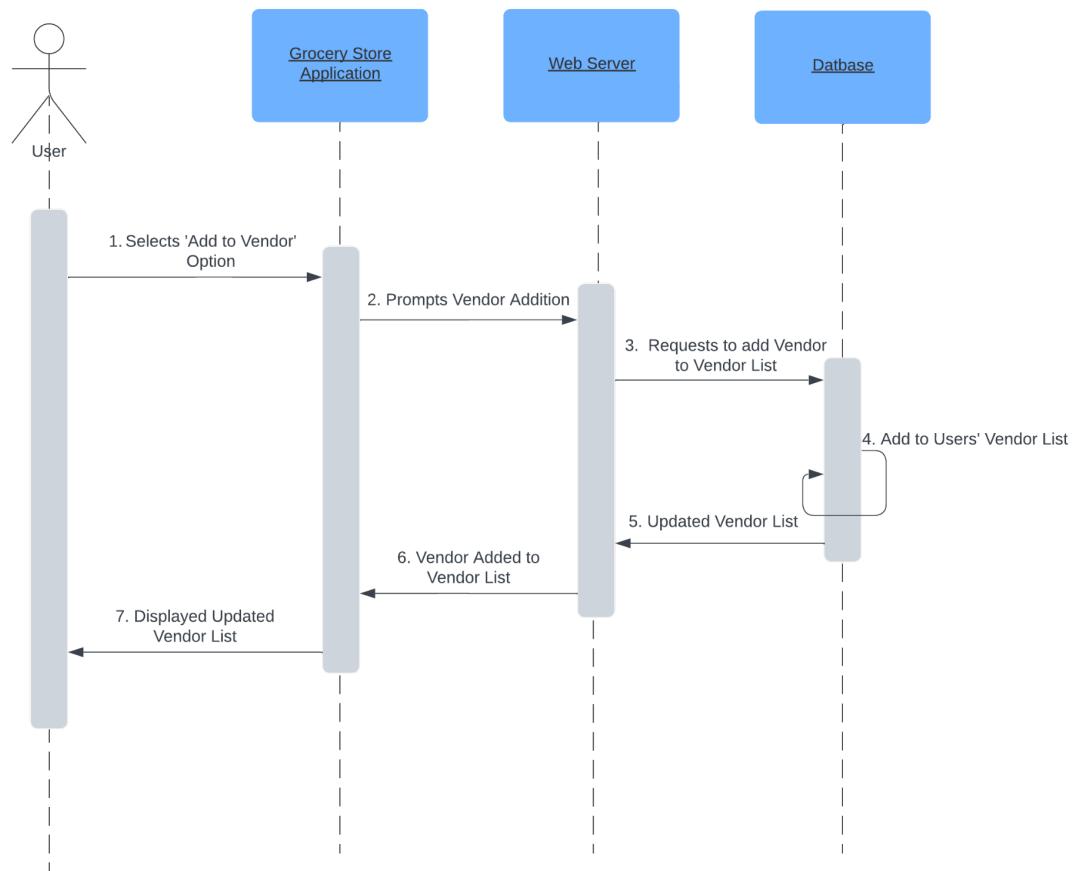
## All Products Search



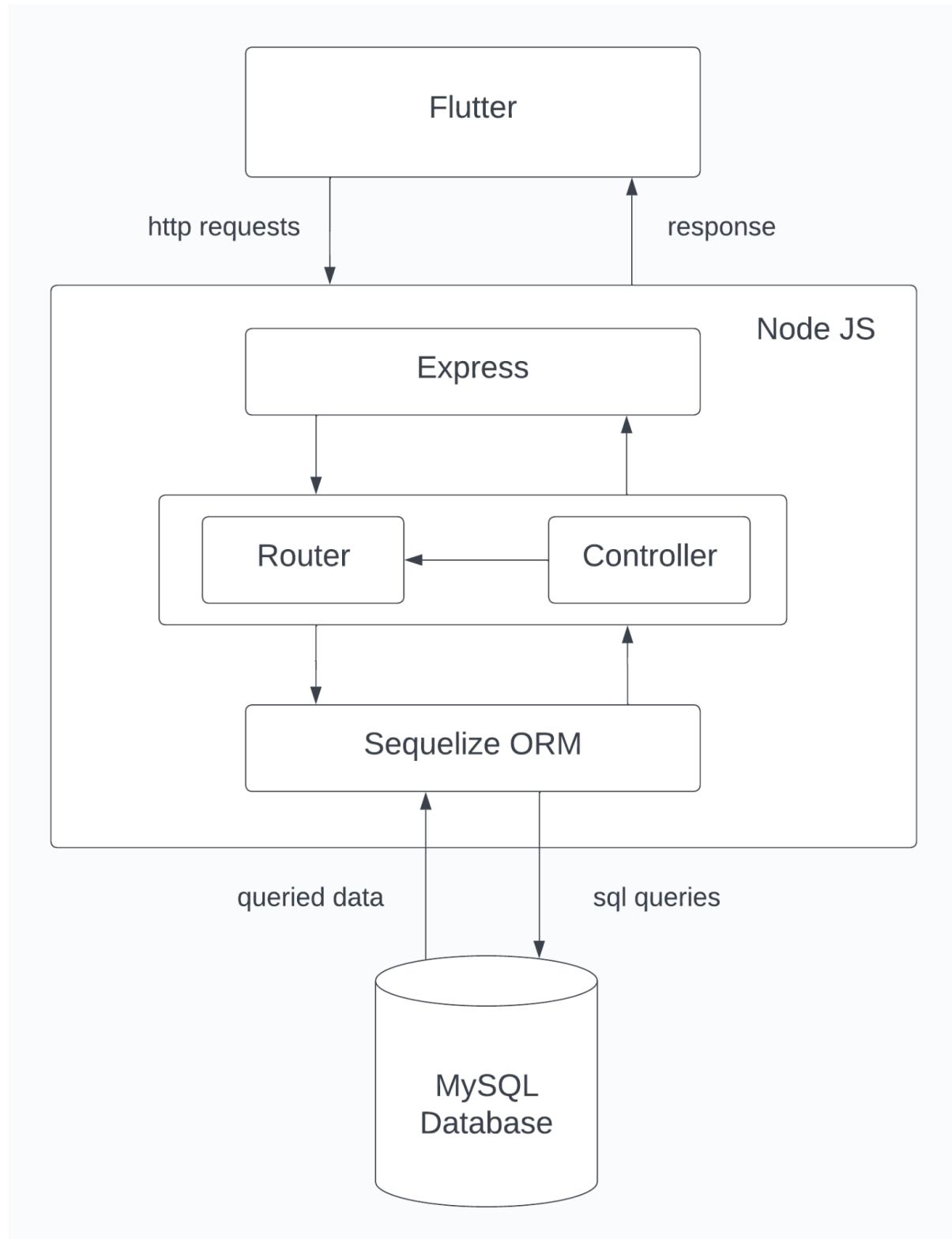
## OTP Code Generation and Delivery



## Add Vendor to Vendor List



## API Design



## ML Design

### Data Collection

#### Local Pharmacy Dataset

##### Attributes:

1. saleinvcode
2. customerref
3. date
4. invdiscperc
5. flatdisc
6. misccharges
7. invsalestax
8. remarks
9. looseqty
10. packqty
11. price
12. itemdiscperc
13. packunits
14. batch
15. expiry
16. salestax
17. itemname
18. datestring
19. customer
20. rowid

**Instances:** 300,000

**Format:** XLS

**Time Frame:** July 2022 to June 2023

**Description:** The data needs to be cleaned. saleinvcode, customerref, invdiscperc, flatdisc, misccharges, invsalestax, packqty, itemdiscperc, batch, salestax, datestring, customer, and rowid need to be removed as they are either empty, zero, same for every entry, or irrelevant to demand forecasting. The remarks attribute is used to enter the names of customers, therefore, it should be removed to ensure privacy. The date attribute contains the date and the time, it should be split into two parts, date and time. The sales are only recorded in terms of looseqty. The dataset also contains expiry dates which will enable us to analyse the perishable aspect of products. This dataset is uploaded on our GitHub repository.

## Corporación Favorita Grocery Sales Forecasting

### Datasets:

#### Training Data

1. Date
2. On Promotion
3. Unit Sales
4. Store Number
5. Item Number

#### Stores

1. City
2. State
3. Type
4. Cluster

#### Items

1. Class
2. Perishable
3. Family

#### Transactions

1. Date
2. Transactions

#### Oil

1. Date
2. Oil Price

#### Holidays

1. Type
2. Locale
3. Locale Name
4. Description
5. Transferred

**Instances:** 126 Million

**Size:** 4.7 GB (Training Data)

**Format:** CSV

**Time Frame:** January 2013 to August 2017

**Description:** This is a dataset from Corporación Favorita Grocery Sales Forecasting competition hosted on Kaggle 6 years ago. The dataset contains millions of instances, spanning over 5 years, including key variables such as holidays, oil prices, and location. The dataset is considered credible based on the fact it's shared by a large grocery store chain, hosted as a competition on Kaggle with prizes of \$30,000 and 1500+ participants, and most importantly is used for demand forecasting in several research papers including [1] and [2].

## Instacart Market Basket Analysis

### Datasets:

#### Aisles

1. aisle\_id
2. aisle

#### Departments

1. department\_id
2. department

#### Prior Product Orders

1. order\_id
2. product\_id
3. add\_to\_cart\_order
4. reordered

#### Orders

1. order\_id
2. user\_id
3. eval\_set
4. order\_number
5. order\_dow
6. order\_hour\_of\_day
7. days\_since\_prior\_order

#### Products

1. product\_id
2. product\_name
3. aisle\_id
4. department\_id

**Instances:** 3.4 Million

**Format:** CSV

**Description:** This is a dataset from the Instacart Market Basket Analysis competition hosted on Kaggle 6 years ago. The dataset contains millions of instances, including key variables such as aisle, department, day of week, and hour of day. The dataset is considered credible based on the fact it's shared by a large grocery delivery company, hosted as a competition on Kaggle with prizes of \$25,000 and 2500+ participants.

### Other Datasets for Further Exploration

- [Grupo Bimbo Inventory Demand](#)
- [Store Item Demand Forecasting Challenge](#)

## Feature Engineering

### Local Pharmacy Dataset

We got one year's data from the pharmacy. There was a separate csv for each month. So, the first step was to combine the csv files into one dataset (Figure 1).

```
import pandas as pd
import glob

csv_files_path = '../Data/Pharmacy/'

csv_files = sorted(glob.glob(csv_files_path + '*.csv'))

combined_data = pd.DataFrame()

for csv_file in csv_files:
    month_data = pd.read_csv(csv_file)
    combined_data = pd.concat([combined_data, month_data], ignore_index=True)
```

**Figure 1** Combining monthly csv files into one dataset

Then, the dataset is inspected (Figure 2). It is noticed that the data needs to be cleaned. saleinvcode, customerref, invdiscperc, flatdisc, misccharges, invsalestax, packqty, itemdiscperc, batch, salestax, datestring customer, and rowid need to be removed as they are either empty, zero, same for every entry, or irrelevant to demand forecasting. The remarks attribute is used to enter the names of customers, therefore, it should be removed to ensure privacy (Figure 3).

| In [3]: df.head() |             |             |             |          |             |             |         |          |         |       |              |           |       |              |          |                             |            |                         |        |
|-------------------|-------------|-------------|-------------|----------|-------------|-------------|---------|----------|---------|-------|--------------|-----------|-------|--------------|----------|-----------------------------|------------|-------------------------|--------|
| saleinvcode       | CustomerRef | date        | invdiscperc | flatdisc | misccharges | invsalestax | remarks | looseqty | packqty | price | itemdiscperc | packunits | batch | expiry       | salestax | itemname                    | datestring | customer                | rowid  |
| 115439            | NaN         | 7/1/22 0:02 | 0.0         | 0.0      | 0           | 0           | NaN     | 1        | 0       | 18.00 | 0.0          | 1         | .     | 7/5/24 0:00  | 0        | PEDITRAL (LEMON) ORS SACHET | 1/7/2022   | ****CASH SALES CUSTOMER | 226819 |
| 115440            | NaN         | 7/1/22 0:03 | 0.0         | 3.0      | 0           | 0           | NaN     | 3        | 0       | 6.31  | 0.0          | 20        | .     | 2/8/24 0:00  | 0        | SPIROMIDE 20MG TAB          | 1/7/2022   | ****CASH SALES CUSTOMER | 226820 |
| 115440            | NaN         | 7/1/22 0:03 | 0.0         | 3.0      | 0           | 0           | NaN     | 2        | 0       | 7.83  | 0.0          | 30        | .     | 8/1/25 0:00  | 0        | CARLOV 6.25MG TAB(30'S)     | 1/7/2022   | ****CASH SALES CUSTOMER | 226821 |
| 115440            | NaN         | 7/1/22 0:03 | 0.0         | 3.0      | 0           | 0           | NaN     | 2        | 0       | 34.29 | 0.0          | 14        | .     | 10/1/24 0:00 | 0        | NEXUM 40MG CAP              | 1/7/2022   | ****CASH SALES CUSTOMER | 226822 |
| 115441            | NaN         | 7/1/22 0:04 | 0.0         | 0.0      | 0           | 0           | NaN     | 8        | 0       | 6.31  | 0.0          | 20        | .     | 2/8/24 0:00  | 0        | SPIROMIDE 20MG TAB          | 1/7/2022   | ****CASH SALES CUSTOMER | 226823 |

**Figure 2** Inspecting the dataset

| df.head() |             |      |          |       |              |                             |          |            |
|-----------|-------------|------|----------|-------|--------------|-----------------------------|----------|------------|
|           |             | date | looseqty | price | packunits    | expiry                      | itemname | datestring |
| 0         | 7/1/22 0:02 | 1    | 18.00    | 1     | 7/5/24 0:00  | PEDITRAL (LEMON) ORS SACHET | 1/7/2022 |            |
| 1         | 7/1/22 0:03 | 3    | 6.31     | 20    | 2/8/24 0:00  | SPIROMIDE 20MG TAB          | 1/7/2022 |            |
| 2         | 7/1/22 0:03 | 2    | 7.83     | 30    | 8/1/25 0:00  | CARLOV 6.25MG TAB(30'S)     | 1/7/2022 |            |
| 3         | 7/1/22 0:03 | 2    | 34.29    | 14    | 10/1/24 0:00 | NEXUM 40MG CAP              | 1/7/2022 |            |
| 4         | 7/1/22 0:04 | 8    | 6.31     | 20    | 2/8/24 0:00  | SPIROMIDE 20MG TAB          | 1/7/2022 |            |

**Figure 3** Dataset after removing the aforementioned columns

The date attribute contains the date and the time, it should be split into two parts, date and time (Figure 4).

```
df[['date', 'time']] = df['date'].str.split(' ', 1, expand=True)
```

```
df.head()
```

|   | date     | time | itemname                    | packunits | expiry  | price | looseqty |
|---|----------|------|-----------------------------|-----------|---------|-------|----------|
| 0 | 1/7/2022 | 0:02 | PEDITRAL (LEMON) ORS SACHET | 1         | 7/5/24  | 18.00 | 1        |
| 1 | 1/7/2022 | 0:03 | SPIROMIDE 20MG TAB          | 20        | 2/8/24  | 6.31  | 3        |
| 2 | 1/7/2022 | 0:03 | CARLOV 6.25MG TAB(30'S)     | 30        | 8/1/25  | 7.83  | 2        |
| 3 | 1/7/2022 | 0:03 | NEXUM 40MG CAP              | 14        | 10/1/24 | 34.29 | 2        |
| 4 | 1/7/2022 | 0:04 | SPIROMIDE 20MG TAB          | 20        | 2/8/24  | 6.31  | 8        |

**Figure 4** Date is split in date and time

There were multiple instances of sales for each item for each day. So, daily sales data was aggregated based on date and itemname (Figure 5).

```
agg_functions = {
    'packunits': 'first',
    'expiry': 'first',
    'price': 'first',
    'looseqty': 'sum'
}
```

```
combined_df = df.groupby(['date', 'itemname'], as_index=False).agg(agg_functions)
```

```
combined_df.head()
```

|   | date       | itemname                       | packunits | expiry   | price  | looseqty |
|---|------------|--------------------------------|-----------|----------|--------|----------|
| 0 | 2022-07-01 | 10CC SHIFA D/SYRINGE(UNJT)(BM) | 100       | 12/12/24 | 30.00  | 6        |
| 1 | 2022-07-01 | 1CC BD SYRINGE                 | 100       | 12/12/24 | 30.00  | 1        |
| 2 | 2022-07-01 | 3CC SYRINGE INJEKT             | 100       | 3/1/24   | 15.00  | 3        |
| 3 | 2022-07-01 | ACCU CHECK LANCET (CHINA)      | 200       | 12/12/24 | 3.00   | 50       |
| 4 | 2022-07-01 | ACDERMIN GEL                   | 1         | 5/1/23   | 278.44 | 1        |

**Figure 5** Aggregating daily sales data for each item

Lastly, we filtered the data for 'PANADOL TAB' and saved the filtered dataset (Figure 6) .

```
filtered_df = df[df['itemname'] == 'PANADOL TAB']
```

```
filtered_df.head(10)
```

|      | date       | itemname    | packunits | expiry  | price | looseqty |
|------|------------|-------------|-----------|---------|-------|----------|
| 376  | 01/07/2022 | PANADOL TAB | 200       | 4/25/24 | 1.70  | 60       |
| 952  | 02/07/2022 | PANADOL TAB | 200       | 4/25/24 | 1.70  | 70       |
| 1490 | 03/07/2022 | PANADOL TAB | 200       | 4/25/24 | 1.70  | 55       |
| 2671 | 05/07/2022 | PANADOL TAB | 200       | 4/25/24 | 1.45  | 20       |
| 4407 | 08/07/2022 | PANADOL TAB | 200       | 4/25/24 | 1.70  | 70       |

**Figure 6** Filtering Panadol Tab rows

## Corporación Favorita Grocery Sales Forecasting

The initial step involved loading the various datasets required for the analysis including the training set, test set, holiday events (Figure 1.1), oil prices (Figure 1.2), store information (Figure 1.3), and transaction data. After loading the datasets, we examined their content and shape to gain insights into the structure and size of each dataset as part of data exploration.

```
In [3]: holiday_events.head()
```

```
Out[3]:
```

|   | date       | type    | locale   | locale_name | description                   | transferred |
|---|------------|---------|----------|-------------|-------------------------------|-------------|
| 0 | 2012-03-02 | Holiday | Local    | Manta       | Fundacion de Manta            | False       |
| 1 | 2012-04-01 | Holiday | Regional | Cotopaxi    | Provincializacion de Cotopaxi | False       |
| 2 | 2012-04-12 | Holiday | Local    | Cuenca      | Fundacion de Cuenca           | False       |
| 3 | 2012-04-14 | Holiday | Local    | Libertad    | Cantonizacion de Libertad     | False       |
| 4 | 2012-04-21 | Holiday | Local    | Riobamba    | Cantonizacion de Riobamba     | False       |

**Figure 1.1 Holiday Events**

```
In [4]: oil.head()
```

```
Out[4]:
```

|   | date       | dcoilwtico |
|---|------------|------------|
| 0 | 2013-01-01 | NaN        |
| 1 | 2013-01-02 | 93.14      |
| 2 | 2013-01-03 | 92.97      |
| 3 | 2013-01-04 | 93.12      |
| 4 | 2013-01-07 | 93.20      |

**Figure 1.2 Oil Prices**

```
In [6]: stores.head()
```

```
Out[6]:
```

|   | store_nbr | city          | state                          | type | cluster |
|---|-----------|---------------|--------------------------------|------|---------|
| 0 | 1         | Quito         | Pichincha                      | D    | 13      |
| 1 | 2         | Quito         | Pichincha                      | D    | 13      |
| 2 | 3         | Quito         | Pichincha                      | D    | 8       |
| 3 | 4         | Quito         | Pichincha                      | D    | 9       |
| 4 | 5         | Santo Domingo | Santo Domingo de los Tsachilas | D    | 4       |

**Figure 1.3 Store Information**

Then, the date feature in each dataset was, initially a string, converted to the datetime data type (Figure. 1.4). This step is crucial for time series analysis.

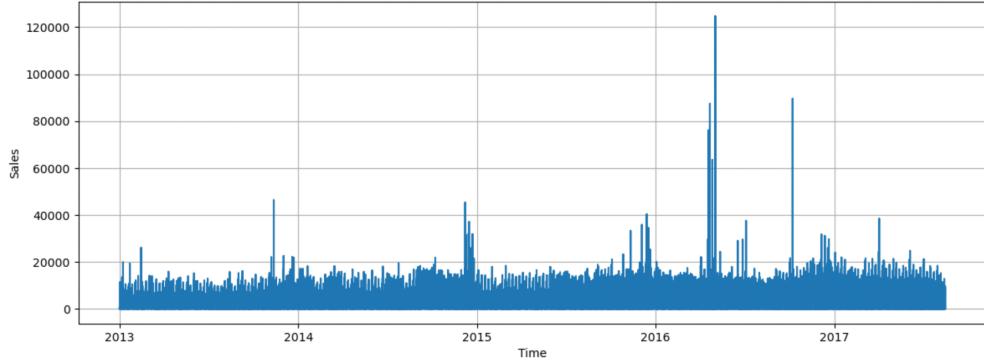
```
In [10]: holiday_events["date"] = pd.to_datetime(holiday_events.date)
oil["date"] = pd.to_datetime(oil.date)
test["date"] = pd.to_datetime(test.date)
train["date"] = pd.to_datetime(train.date)
transactions["date"] = pd.to_datetime(transactions.date)
```

**Figure 1.4 Converting date feature to datetime data type**

To understand the sales patterns over time, a visualisation (Figure 1.5) of the time series was plotted.

```
In [11]: def plot_series(time, series, format="-", start=0, end=None):
    fig, ax = plt.subplots(figsize=(14,5))
    plt.plot(time[start:end], series[start:end], format)
    plt.xlabel("Time")
    plt.ylabel("Sales")
    plt.grid(True)
    plt.show()
    plt.close()

In [12]: plot_series(train["date"], train["sales"], format="-", start=0, end=None)
```



**Figure 1.5** Visualising time series data

Next, data merging was performed (Figure 1.6), holiday events, oil prices, and store information were merged with the training and test sets. Transactions data was not utilised because it isn't available for the test set. The merging operations were conducted sequentially: first with holiday events based on the 'date' column, then second with the oil prices using the 'date' column; and finally with store-specific details using the 'store\_nbr' column. The final expanded dataset was as seen in Figure 1.7.

```
In [13]: holiday_events = holiday_events.drop_duplicates(subset=['date'], keep='last')

In [14]: train.shape
Out[14]: (3000888, 6)

In [15]: train = pd.merge(train,holiday_events,how="left",on="date", validate="many_to_one")

In [16]: train.shape
Out[16]: (3000888, 11)

In [17]: train = pd.merge(train,oil,how="left",on="date")

In [18]: train.shape
Out[18]: (3000888, 12)

In [19]: train = pd.merge(train,stores,how="left",on="store_nbr",suffixes=("-holiday","_stores"))

In [20]: train.shape
Out[20]: (3000888, 16)
```

**Figure 1.6** Joining holiday\_events, oil, and stores

```
In [21]: train.head()

Out[21]:   id      date  store_nbr       family   sales  onpromotion  typeholiday  locale  locale_name  description  transferred  dccoilwtico  city  state  typestores  cluster
0  0  2013-01-01         1  AUTOMOTIVE     0.0          0    Holiday  National    Ecuador  Primer dia del año    False    NaN  Quito  Pichincha      D  13
1  1  2013-01-01         1  BABY CARE     0.0          0    Holiday  National    Ecuador  Primer dia del año    False    NaN  Quito  Pichincha      D  13
2  2  2013-01-01         1      BEAUTY     0.0          0    Holiday  National    Ecuador  Primer dia del año    False    NaN  Quito  Pichincha      D  13
3  3  2013-01-01         1  BEVERAGES     0.0          0    Holiday  National    Ecuador  Primer dia del año    False    NaN  Quito  Pichincha      D  13
4  4  2013-01-01         1      BOOKS     0.0          0    Holiday  National    Ecuador  Primer dia del año    False    NaN  Quito  Pichincha      D  13
```

**Figure 1.7** Training with holiday\_events, oil, and stores features

Extracting time features like the day of the week, month, and year from the date in both training and test sets was crucial (Figure 1.8). The 'day\_of\_week', 'month', and 'year' features, breaking down the week into days, played a key role in capturing and adapting to sales patterns on a weekly, monthly, and yearly basis. This enhanced the model's ability to recognize variations within a week, improving time series forecasting.

```
In [29]: train['day_of_week'] = train['date'].dt.day_of_week
train['day_of_week'] = train['day_of_week']+1
train['month'] = train['date'].dt.month
train['year'] = train['date'].dt.year

In [30]: train.head()
```

|   | id | date       | store_nbr | family     | sales | onpromotion | typeholiday | locale   | locale_name | description        | transferred | dcoilwtico | city  | state     | typestores | cluster | day_of_week | month | year |
|---|----|------------|-----------|------------|-------|-------------|-------------|----------|-------------|--------------------|-------------|------------|-------|-----------|------------|---------|-------------|-------|------|
| 0 | 0  | 2013-01-01 | 1         | AUTOMOTIVE | 0.0   | 0           | Holiday     | National | Ecuador     | Primer dia del ano | False       | NaN        | Quito | Pichincha | D          | 13      | 2           | 1     | 2013 |
| 1 | 1  | 2013-01-01 | 1         | BABY CARE  | 0.0   | 0           | Holiday     | National | Ecuador     | Primer dia del ano | False       | NaN        | Quito | Pichincha | D          | 13      | 2           | 1     | 2013 |
| 2 | 2  | 2013-01-01 | 1         | BEAUTY     | 0.0   | 0           | Holiday     | National | Ecuador     | Primer dia del ano | False       | NaN        | Quito | Pichincha | D          | 13      | 2           | 1     | 2013 |
| 3 | 3  | 2013-01-01 | 1         | BEVERAGES  | 0.0   | 0           | Holiday     | National | Ecuador     | Primer dia del ano | False       | NaN        | Quito | Pichincha | D          | 13      | 2           | 1     | 2013 |
| 4 | 4  | 2013-01-01 | 1         | BOOKS      | 0.0   | 0           | Holiday     | National | Ecuador     | Primer dia del ano | False       | NaN        | Quito | Pichincha | D          | 13      | 2           | 1     | 2013 |

**Figure 1.8** Extracting weekday, month, and year

The 'transferred' column in the holiday events dataset was used to identify and convert transferred holidays to normal days (Figure 1.9). This step ensures consistency in the holiday feature across the dataset.

```
In [37]: test["typeholiday"] = np.where(test["transferred"]==True, 'NDay', test["typeholiday"])
test["typeholiday"] = np.where(test["typeholiday"]=="Work Day", 'NDay', test["typeholiday"])
test["typeholiday"] = test["typeholiday"].fillna("NDay")

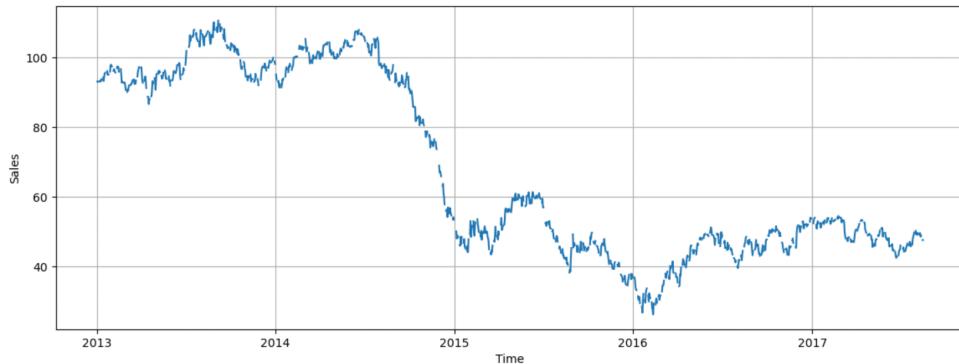
In [38]: display(test["typeholiday"].value_counts(dropna=False))
```

| typeholiday               | count |
|---------------------------|-------|
| NDay                      | 26730 |
| Holiday                   | 1782  |
| Name: count, dtype: int64 |       |

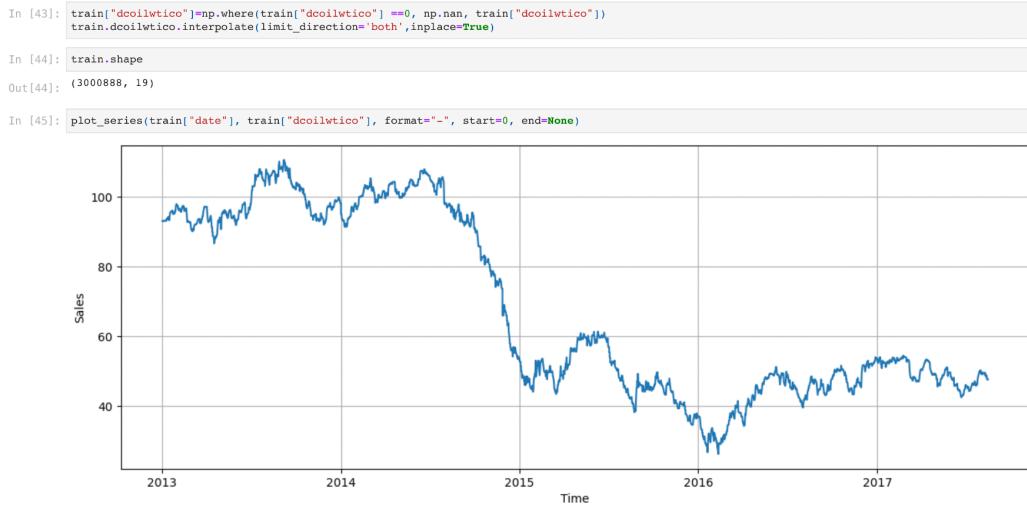
**Figure 1.9** Changing transferred holidays to normal days

Then, zeros in the oil prices column were addressed by interpolating the data. Visualisations before (Figure 1.10) and after interpolation (Figure 1.11) are presented.

```
In [41]: plot_series(train["date"], train["dcoilwtico"], format="-", start=0, end=None)
```



**Figure 1.10** Missing values in oil prices



**Figure 1.11** Fixing missing values in oil prices

Based on info seen (Figure 1.12), 'locale', 'locale\_name', 'description', 'transferred' were dropped and new csv files were generated comprising the final test and training dataset that the model was trained on.

```
In [50]: test.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28512 entries, 0 to 28511
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          28512 non-null   int64  
 1   date        28512 non-null   datetime64[ns]
 2   store_nbr   28512 non-null   int64  
 3   family      28512 non-null   object 
 4   onpromotion 28512 non-null   int64  
 5   typeholiday 28512 non-null   object 
 6   locale      1782 non-null   object 
 7   locale_name 1782 non-null   object 
 8   description 1782 non-null   object 
 9   transferred 1782 non-null   object 
 10  dcoilwtico 28512 non-null   float64 
 11  city        28512 non-null   object 
 12  state       28512 non-null   object 
 13  typesstores 28512 non-null   object 
 14  cluster     28512 non-null   int64  
 15  day_of_week 28512 non-null   int32  
 16  month       28512 non-null   int32  
 17  year        28512 non-null   int32  
dtypes: datetime64[ns](1), float64(1), int32(3), int64(4), object(9)
memory usage: 3.6+ MB
```

**Figure 1.12** Removing ID and columns with mostly null values

## Model Shortlisting

The slide has a light orange background. At the top left is a circular profile picture of a woman. To its right is a stylized icon of a leaf or network of dots. The title 'Suite of demand forecasting models' is centered in large white font. Below the title is a bulleted list of models:

- Autoregressive models
- Prophet
- ML: Random Forest, XGBoost, ...
- RNNs, LSTMs, Transformers
- Hierarchical Forecasting
- Bayesian Hierarchical Forecasting

To the right of the list is a scatter plot showing a time series from 2011 to 2013. The x-axis represents time, and the y-axis represents value, ranging from 0 to 10,000. A dense cloud of blue points shows the historical data, and a solid red line represents the forecasted trend.

At the bottom of the slide, there is contact information: 'Shawn L. Ramirez, PhD' on the left, an LinkedIn icon followed by 'slramz' in the center, and 'shawn@shelfengine.com' on the right.

## Shelf Engine: Demand Forecasting Models

### Project Part I

As the inspiration behind our idea of demand forecasting was Shelf Engine, we initially decided to use the same models that they did, since their results using these models were extremely accurate.

#### Random Forest

We started the exploration with Random Forest because of its simplicity, flexibility, and our past experience with it making it a good choice. What makes Random Forest special, is its ability to capture complex non-linear patterns without extensive tuning.

#### XGBoost

Since we have structured data, XGBoost was considered a suitable choice. It is known to perform well in time series forecasting problems as evident by its popularity in competitions related to demand forecasting on kaggle. Due to the size of our dataset, XGBoost's speed and efficiency really helped in going over several iterations of training and testing.

#### Prophet (Facebook)

Prophet is designed specifically for time series forecasting with daily observations that display patterns like seasonality and holidays. As we had historical data for several seasons, Prophet seemed like a solid choice.

## **RNN (LSTM/GRU)**

RNNs are powerful in modelling sequential patterns, making them suitable for time series forecasting. They can capture temporal dependencies in sales data over different time intervals.

## **Project Part II**

The models that were used for FYP I provided a solid foundation, whereas the models that will be implemented in FYP II leverage recent advancements to further enhance prediction accuracy and address specific challenges in the time series domain.

### **LightGBM**

LightGBM has shown excellent performance in time series forecasting, as evidenced by its success in the M5 competition. Its efficiency makes it suitable for handling large grocery store datasets.

### **N-BEATS**

N-BEATS has proven effective in recent competitions, outperforming even well-established methods. It has demonstrated superior performance compared to traditional statistical approaches.

### **DeepAR (Amazon)**

DeepAR model is based on a set of RNN models, where each model is trained on a specific subset of the data. Finally, the predictions from all of the models are combined allowing DeepAR to handle multiple time series effectively.

### **Temporal Fusion Transformer (Google)**

According to [3], Temporal Fusion Transformer outperforms all prominent Deep Learning models for time series forecasting.

## **Training and Testing**

## **Deployment**

We created a sub directory, ModelDeployment, in our GitHub repository for version control. Installed Anaconda for package management and created a Conda environment (deployment) for the project. Developed a Flask web application in Python. Incorporated a simple form in the html template to take input (family and date) from the user. Added our demand forecasting model, trained on the kaggle dataset, and saved as joblib files. Updated the Flask application to handle model predictions based on the family and date user enters, along with related features (whether it was a holiday on that date or what was the price of the oil on that date). Modified the html to display the model's prediction and the actual sales from the selected date for the selected family. Uploaded pre-trained models to a 'models' folder in the project directory. Adjusted the app.py file to load models and handle form submissions. Checked the functioning of the model locally by selecting the date and family

and obtaining predictions. Prepared for deployment by creating a requirements.txt file listing necessary libraries. Used Digital Ocean to deploy the Flask application, connecting it to the GitHub repository. Resolved errors during deployment and awaited the completion of the deployment process. Checked the live url to access the deployed demand forecasting web application.

## Conclusion

In conclusion, this design document provides a comprehensive blueprint for developing a sophisticated demand forecasting system tailored for the grocery retail industry. The integration of diverse technologies, ranging from backend development tools to machine learning frameworks, highlights the depth and complexity of the solution. The document not only emphasises the importance of data design and feature engineering but also sheds light on the selection and deployment of machine learning models for accurate demand predictions. The deployment on Digital Ocean, underscores the practicality and real-world applicability of the proposed solution. Overall, this document serves as a guide for building a robust and scalable system that can significantly enhance the efficiency of inventory management in the grocery retail sector.

## References

- [1] [G. Kechyn, L. Yu, Y. Zang, S. Kechyn, "Sales Forecasting using WaveNet within the Framework of the Kaggle Competition," arXiv:1803.04037v1 \[cs.LG\], Mar. 11, 2018.](#)
- [2] [V. Prabhakar, D. Sayiner, U. Chakraborty, T. Nguyen, M. A. Lanham, "Demand forecasting for a large grocery chain in Ecuador."](#)
- [3] [Shereen Elsayed et al. Do We Really Need Deep Learning Models for Time Series Forecasting?](#)