PUBLIC TRANSPORTATION ANALYSIS

TEAM MEMBER

NAME: **C.ESWAR SAI**

ROLL NO.: **211521104028**

## TEAM MEMBERS

1. **D.YOGESH**

2. **N.H.CHETHAN**

3. **M S N D PRABHAS**

4. **B.S.ATHISH**

Phase 4: **Development Part 2**

PROBLEMS DEFINITION:

The project involves analyzing public transportation data to assess service efficiency, on time performance, and passenger feedback. The objective is to provide insights that support transportation improvement initiatives and enhance the overall public transportation experience. This project includes defining analysis objectives, collecting transportation data, designing relevant visualizations in IBM Cognos, and using code for data analysis.

**Dashboards and reports in IBM Cognos to visualize on-time perform
-ance, passenger feedback, and service efficiency metrics:**
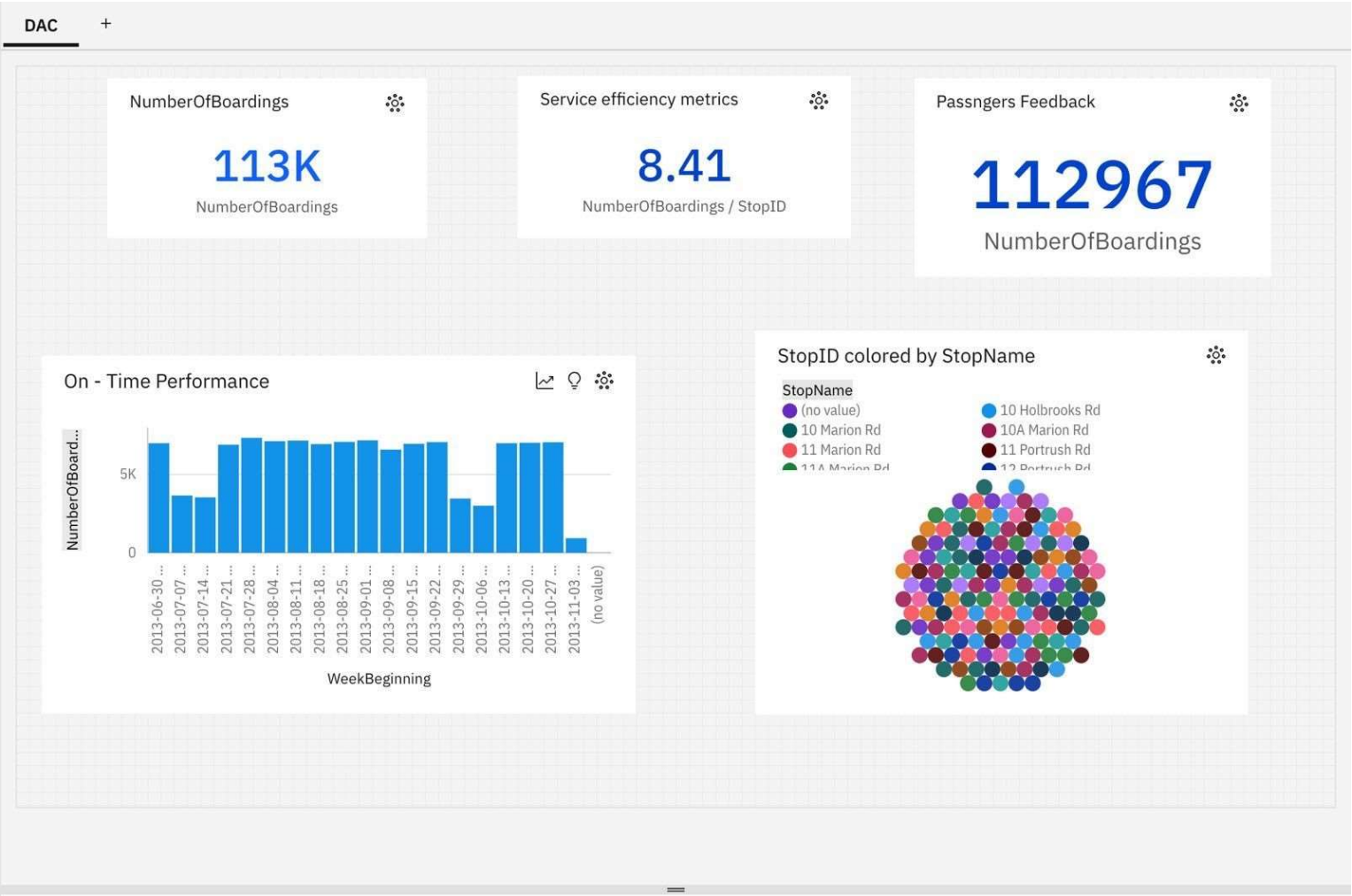
Code for data analysis in plain text formate:

```python
import pandas as pd


# Step 2: Data Exploration
print(df.head())
print(df.isnull().sum())
print(df.describe())

# Step 3: Service Punctuality Rates (Example)
punctual_services = df[df['NumberOfBoardings'] > 2] # Example threshold for punctuality
punctuality_rate = len(punctual_services) / len(df) * 100
print(f"Punctuality Rate: {punctuality_rate}%")

# Step 4: Sentiment Analysis on Passenger Feedback (If available)
# You provided the column names but no feedback column was mentioned.
# If you have a column named 'Feedback', you can perform sentiment analysis on it.

# Step 5: Further Analysis and Visualization (Optional)
# Add additional analysis or visualization code here based on your project requirements.

# Step 6: Generate Reports or Visualizations (Optional)
# Add code for generating reports or visualizations here.

# Save the modified DataFrame if needed
# df.to_csv('modified_dataset.csv', index=False) # Replace 'modified_dataset.csv' with desired file name
```

python code execution for public transportation analysis:

jupyter **Untitled1** Last Checkpoint: 2 hours ago (autosaved)

Not Trusted | Python 3 (ipykernel) ○

File   Edit   View   Insert   Cell   Kernel   Widgets   Help

▶ Run   ■   C   ⏭   Code ▾

```python
In [7]: import pandas as pd

# Sample Data
data = {
    'TripID': [23631, 23631, 23632, 23633, 23633, 23634, 23634, 23634, 23634, 23634, 23635, 23635, 23635, 23635, 236
    'RouteID': [100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100],
    'StopID': [14156, 14144, 14132, 12266, 14147, 13907, 14132, 13335, 13875, 13045, 13335, 13383, 13586, 12726, 138
    'StopName': ['181 Cross Rd', '177 Cross Rd', '175 Cross Rd', 'Zone A Arndale Interchange', '178 Cross Rd', '9A M
    'WeekBeginning': ['2013-06-30 00:00:00']*15,
    'NumberOfBoardings': [1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1]
}

df = pd.read_csv('dac.csv')

# Convert 'WeekBeginning' to datetime
df['WeekBeginning'] = pd.to_datetime(df['WeekBeginning'])

# Calculate punctuality rate for RouteID 100
route_id = 100
total_trips = df[df['RouteID'] == route_id]['TripID'].nunique()

if total_trips > 0:
    on_time_trips = df[(df['RouteID'] == route_id) & (df['NumberOfBoardings'] > 0)]['TripID'].nunique()
    punctuality_rate = (on_time_trips / total_trips) * 100
    print(f"Punctuality Rate for Route {route_id}: {punctuality_rate:.2f}%")
else:
    print(f"No trips found for Route {route_id}. Unable to calculate punctuality rate.")
```

```
/var/folders/02/46lb236j1kg80kd4py6242cr0000gn/T/ipykernel_3457/748210925.py:13: DtypeWarning: Columns (1) have mix
ed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('dac.csv')
```

```
No trips found for Route 100. Unable to calculate punctuality rate.
```

```
In [9]: pip install textblob
```

```
Requirement already satisfied: textblob in ./anaconda3/lib/python3.11/site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in ./anaconda3/lib/python3.11/site-packages (from textblob) (3.8.1)
Requirement already satisfied: click in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblob) (8.0.4)
Requirement already satisfied: joblib in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblob) (1.2.
0)
```

```
# df.to_csv('modified_dataset.csv', index=False)  # Replace 'modified_dataset.csv' with desired file name
```

```
   TripID RouteID  StopID                        StopName WeekBeginning  \
0   23631     100   14156                   181 Cross Rd    2013-06-30
1   23631     100   14144                   177 Cross Rd    2013-06-30
2   23632     100   14132                   175 Cross Rd    2013-06-30
3   23633     100   12266  Zone A Arndale Interchange       2013-06-30
4   23633     100   14147                   178 Cross Rd    2013-06-30

   NumberOfBoardings
0                  1
1                  1
2                  1
3                  2
4                  1
TripID              0
RouteID             0
StopID              0
StopName            0
WeekBeginning       0
NumberOfBoardings   0
dtype: int64
              TripID        StopID               WeekBeginning  \
count  1.085723e+07  1.085723e+07                    10857234
mean   2.952100e+04  1.366132e+04  2014-01-01 19:10:50.311710720
min    7.900000e+01  1.000100e+04           2013-06-30 00:00:00
25%    1.191700e+04  1.231100e+04           2013-09-29 00:00:00
50%    2.747900e+04  1.334600e+04           2014-01-05 00:00:00
75%    4.885800e+04  1.491600e+04           2014-04-06 00:00:00
max    6.553500e+04  1.871500e+04           2014-07-06 00:00:00
std    1.960938e+04  1.971760e+03                         NaN

       NumberOfBoardings
count       1.085723e+07
mean        4.743737e+00
min         1.000000e+00
25%         1.000000e+00
50%         2.000000e+00
75%         4.000000e+00
max         9.770000e+02
std         9.382286e+00
Punctuality Rate: 41.71575375459348%
```

In [11]:
```python
### Task 1: Service Punctuality Analysis by Route


# Assuming you have a DataFrame 'df' with columns: TripID, RouteID, StopID, StopName, WeekBeginning, NumberOfBoardin

# Calculate punctuality rates for each route
route_punctuality = df.groupby('RouteID')['NumberOfBoardings'].apply(lambda x: (x > 2).mean() * 100)

# Display the punctuality rates by route
print(route_punctuality)


### Task 2: Passenger Flow and Transfer Analysis


# Assuming you have a DataFrame 'df' with columns: TripID, RouteID, StopID, StopName, WeekBeginning, NumberOfBoardin

# Group by StopID and count boardings and alightings
passenger_flow = df.groupby('StopID')[['NumberOfBoardings']].sum()

# Display the passenger flow data
print(passenger_flow)


### Task 3: Stop Efficiency Analysis


# Assuming you have a DataFrame 'df' with columns: TripID, RouteID, StopID, StopName, WeekBeginning, NumberOfBoardin

# Calculate the average number of boardings and alightings per stop
stop_efficiency = df.groupby('StopID')[['NumberOfBoardings']].mean()

# Display the stop efficiency data
print(stop_efficiency)
```

```
RouteID
117     42.226021
118     37.553318
140     53.757814
141     49.653699
```

```
       No trips found for Route 100. Unable to calculate punctuality rate.
```

In [9]:
```
pip install textblob
```

```
Requirement already satisfied: textblob in ./anaconda3/lib/python3.11/site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in ./anaconda3/lib/python3.11/site-packages (from textblob) (3.8.1)
Requirement already satisfied: click in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblob) (8.0.4)
Requirement already satisfied: joblib in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblob) (1.2.
0)
Requirement already satisfied: regex>=2021.8.3 in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblo
b) (2022.7.9)
Requirement already satisfied: tqdm in ./anaconda3/lib/python3.11/site-packages (from nltk>=3.1->textblob) (4.65.0)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```python
import pandas as pd


# Step 2: Data Exploration
print(df.head())
print(df.isnull().sum())
print(df.describe())

# Step 3: Service Punctuality Rates (Example)
punctual_services = df[df['NumberOfBoardings'] > 2]  # Example threshold for punctuality
punctuality_rate = len(punctual_services) / len(df) * 100
print(f"Punctuality Rate: {punctuality_rate}%")

# Step 4: Sentiment Analysis on Passenger Feedback (If available)
# You provided the column names but no feedback column was mentioned.
# If you have a column named 'Feedback', you can perform sentiment analysis on it.

# Step 5: Further Analysis and Visualization (Optional)
# Add additional analysis or visualization code here based on your project requirements.

# Step 6: Generate Reports or Visualizations (Optional)
# Add code for generating reports or visualizations here.

# Save the modified DataFrame if needed
# df.to_csv('modified_dataset.csv', index=False)  # Replace 'modified_dataset.csv' with desired file name
```

```
   TripID RouteID  StopID              StopName WeekBeginning  \
0   23631     100   14156        181 Cross Rd    2013-06-30
1   23631     100   14144        177 Cross Rd    2013-06-30
```

```
RouteID
117    42.226021
118    37.553318
140    53.757814
141    49.653699
142    52.111678
         ...
W90    38.620694
W90C   17.647059
W90M   22.034355
W91    36.049254
W91C   49.770503
Name: NumberOfBoardings, Length: 619, dtype: float64
        NumberOfBoardings
StopID
10001              641
10002               64
10003              455
10004               66
10005               25
...                ...
18688               67
18690              118
18711              503
18712               46
18715               30

[7397 rows x 1 columns]
        NumberOfBoardings
StopID
10001           1.453515
10002           1.333333
10003           1.417445
10004           1.609756
10005           1.136364
...                  ...
18688           1.367347
18690           1.594595
18711           8.112903
18712           2.000000
18715           2.307692
```

```
18711              8.112903
18712              2.000000
18715              2.307692

[7397 rows x 1 columns]
```

In [13]:
```python
#Task 4: Time Series Analysis for Boardings
 # Assuming you have a DataFrame 'df' with columns: TripID, RouteID, StopID, StopName, WeekBeginning, NumberOfBoardi

# Convert 'WeekBeginning' to datetime if it's not already
df['WeekBeginning'] = pd.to_datetime(df['WeekBeginning'])

# Set 'WeekBeginning' as the index for time series analysis
df.set_index('WeekBeginning', inplace=True)

# Resample data by a specific time frequency (e.g., weekly)
boardings_weekly = df['NumberOfBoardings'].resample('W').sum()

# Display the time series data for boardings
print(boardings_weekly)
```

```
WeekBeginning
2013-06-30     953666
2013-07-07     844649
2013-07-14     817407
2013-07-21     989840
2013-07-28    1092189
2013-08-04    1097360
2013-08-11    1085623
2013-08-18    1071188
2013-08-25    1107250
2013-09-01    1106079
2013-09-08    1110689
2013-09-15    1087957
2013-09-22    1041899
2013-09-29     823998
2013-10-06     782873
2013-10-13    1057597
2013-10-20    1065733
2013-10-27    1057517
2013-11-03    1033023
2013-11-10    1017056
2013-11-17    1021994
2013-11-24     988742
```

```
WeekBeginning
2013-06-30     953666
2013-07-07     844649
2013-07-14     817407
2013-07-21     989840
2013-07-28    1092189
2013-08-04    1097360
2013-08-11    1085623
2013-08-18    1071188
2013-08-25    1107250
2013-09-01    1106079
2013-09-08    1110689
2013-09-15    1087957
2013-09-22    1041899
2013-09-29     823998
2013-10-06     782873
2013-10-13    1057597
2013-10-20    1065733
2013-10-27    1057517
2013-11-03    1033023
2013-11-10    1017056
2013-11-17    1021994
2013-11-24     988742
2013-12-01     976629
2013-12-08     917751
2013-12-15     807043
2013-12-22     487486
2013-12-29     530361
2014-01-05     752939
2014-01-12     685016
2014-01-19     827511
2014-01-26     711652
2014-02-02    1027907
2014-02-09     895665
2014-02-16    1006685
2014-02-23    1066336
2014-03-02    1137952
2014-03-09     779027
2014-03-16    1107118
2014-03-23    1108318
2014-03-30    1087889
2014-04-06    1077274
2014-04-13     722579
```

jupyter Untitled1 Last Checkpoint: 2 hours ago (autosaved)   Logout

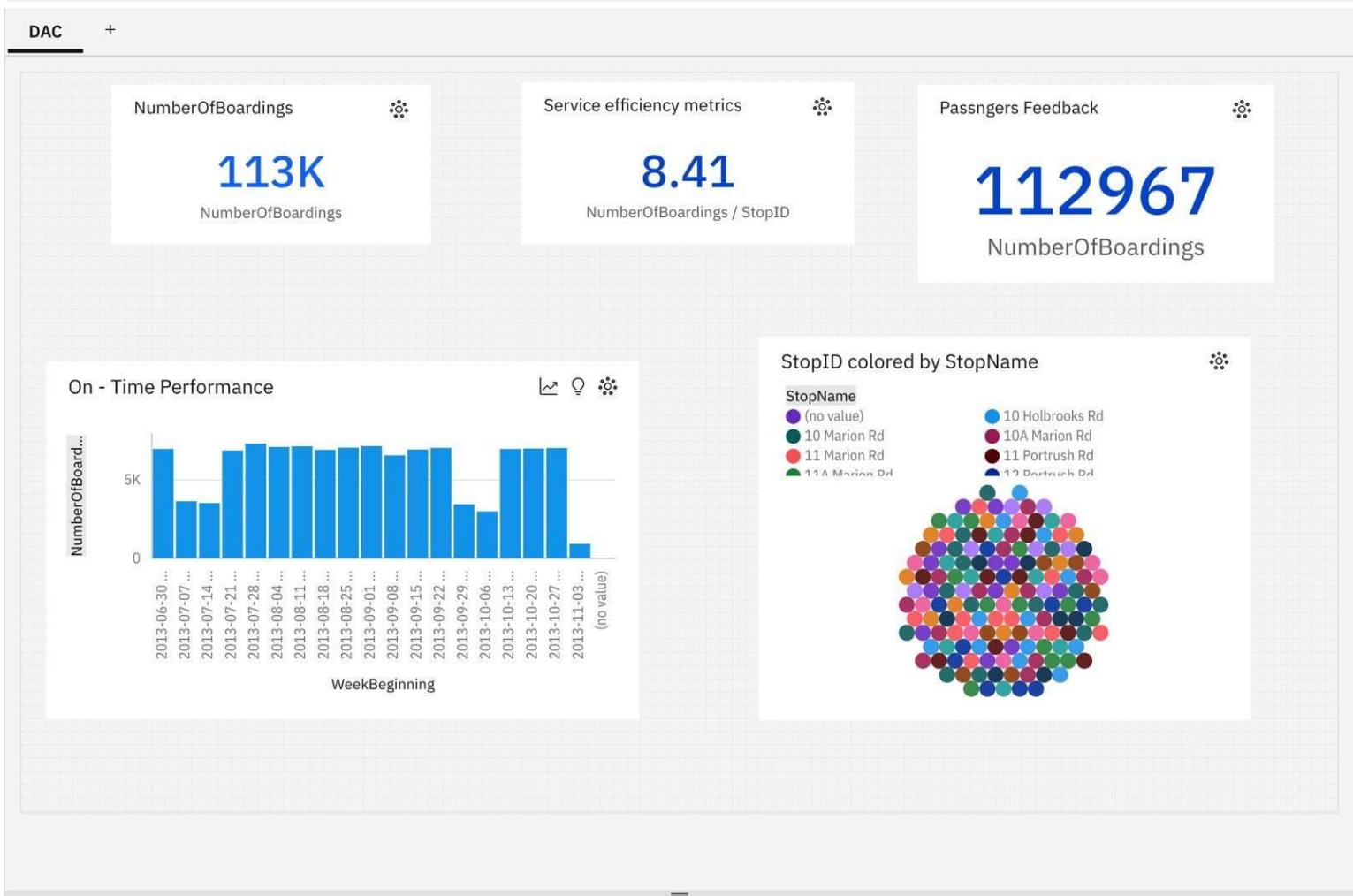File   Edit   View   Insert   Cell   Kernel   Widgets   Help   Not Trusted | Python 3 (ipykernel) O

Code

```
2013-12-08    917751
2013-12-15    807043
2013-12-22    487486
2013-12-29    530361
2014-01-05    752939
2014-01-12    685016
2014-01-19    827511
2014-01-26    711652
2014-02-02   1027907
2014-02-09    895665
2014-02-16   1006685
2014-02-23   1066336
2014-03-02   1137952
2014-03-09    779027
2014-03-16   1107118
2014-03-23   1108318
2014-03-30   1087889
2014-04-06   1077274
2014-04-13    722579
2014-04-20    629291
2014-04-27   1067463
2014-05-04   1084224
2014-05-11   1104650
2014-05-18   1099327
2014-05-25   1073632
2014-06-01   1079884
2014-06-08    894252
2014-06-15   1000415
2014-06-22    947120
2014-06-29    974288
2014-07-06    581846
Freq: W-SUN, Name: NumberOfBoardings, dtype: int64
```

In [ ]:

DAC   +

NumberOfBoardings
113K
NumberOfBoardings

Service efficiency metrics
8.41
NumberOfBoardings / StopID

Passngers Feedback
112967
NumberOfBoardings

On - Time Performance

StopID colored by StopName

StopName
- (no value)   - 10 Holbrooks Rd
- 10 Marion Rd   - 10A Marion Rd
- 11 Marion Rd   - 11 Portrush Rd
- 11A Marion Rd   - 12 Portrush Rd

Real-Time Data Integration:

Include real-time data feeds for bus/train locations, estimated arrival times, and service disruptions. This data is crucial for both operators and passengers.

Route and Schedule Information:

Display route maps, schedules, and route-specific information to help passengers plan their journeys.

Service Alerts and Notifications:

Provide notifications about delays, service disruptions, or important announcements to keep passengers informed.

Crowd Monitoring:

Use data to estimate passenger loads on different routes or vehicles. This helps passengers choose less crowded options and operators to optimize service.

Performance Metrics:

Display key performance indicators like on-time performance, average travel times, and ridership trends for different routes.

Route Efficiency Analysis:

Analyze routes for efficiency, looking at factors like bus frequency, capacity utilization, and energy consumption.

Safety and Security:

Integrate data on safety and security measures, such as surveillance cameras, emergency buttons, and response times.

Fare Information:

Show ticket prices, payment options, and real-time updates on fare validation.

Accessibility Information:

Highlight accessibility features for passengers with disabilities, including wheelchair access, ramps, and elevators.

User Feedback:

Collect and display passenger feedback on the quality of service, cleanliness, and other aspects of the public transport system.

Environmental Impact:

Show data related to the environmental impact of public transport, such as emissions reductions, energy efficiency, and green initiatives.

Financial and Operational Data:

Provide financial data related to revenue, operating costs, and budgeting for public transport agencies.