In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
```

In [3]:
```
df = pd.read_csv("cleaned_bin_data.csv", parse_dates=['timestamp'])

# Select bins for forecasting
bins_to_forecast = [1510830 , 1511208 , 1511196 ]
df = df[df['Bin ID'].isin(bins_to_forecast)]

# Filter data up to May 3, 2021
train_data = df[df['timestamp'] < "2021-04-26"]
print(train_data)
```

```
      Bin ID  Fullness  fullnessThreshold  timestamp    reason  year  month  \
0    1510830         6                  6 2018-06-26  FULLNESS  2018      6
1    1510830         8                  6 2018-06-29  FULLNESS  2018      6
2    1510830         6                  6 2018-06-30  FULLNESS  2018      6
3    1510830         6                  6 2018-07-06  FULLNESS  2018      7
4    1510830         6                  6 2018-07-11  FULLNESS  2018      7
...      ...       ...                ...        ...       ...   ...    ...
3499 1511208         8                  8 2021-04-15  FULLNESS  2021      4
3500 1511208         8                  8 2021-04-17  FULLNESS  2021      4
3501 1511208        10                  8 2021-04-19  FULLNESS  2021      4
3502 1511208        10                  8 2021-04-22  FULLNESS  2021      4
3503 1511208         8                  8 2021-04-25  FULLNESS  2021      4

      day  day_of_week  fullness_change
0      26            1              0.0
1      29            4              2.0
2      30            5             -2.0
3       6            4              0.0
4      11            2              0.0
...   ...          ...              ...
3499   15            3              0.0
3500   17            5              0.0
3501   19            0              2.0
3502   22            3              0.0
3503   25            6             -2.0

[817 rows x 10 columns]
```

LSTM

In [10]:
```
def prepare_data(series, time_steps=7):
    X, y = [], []
    for i in range(len(series) - time_steps):
        X.append(series[i : i + time_steps])
```

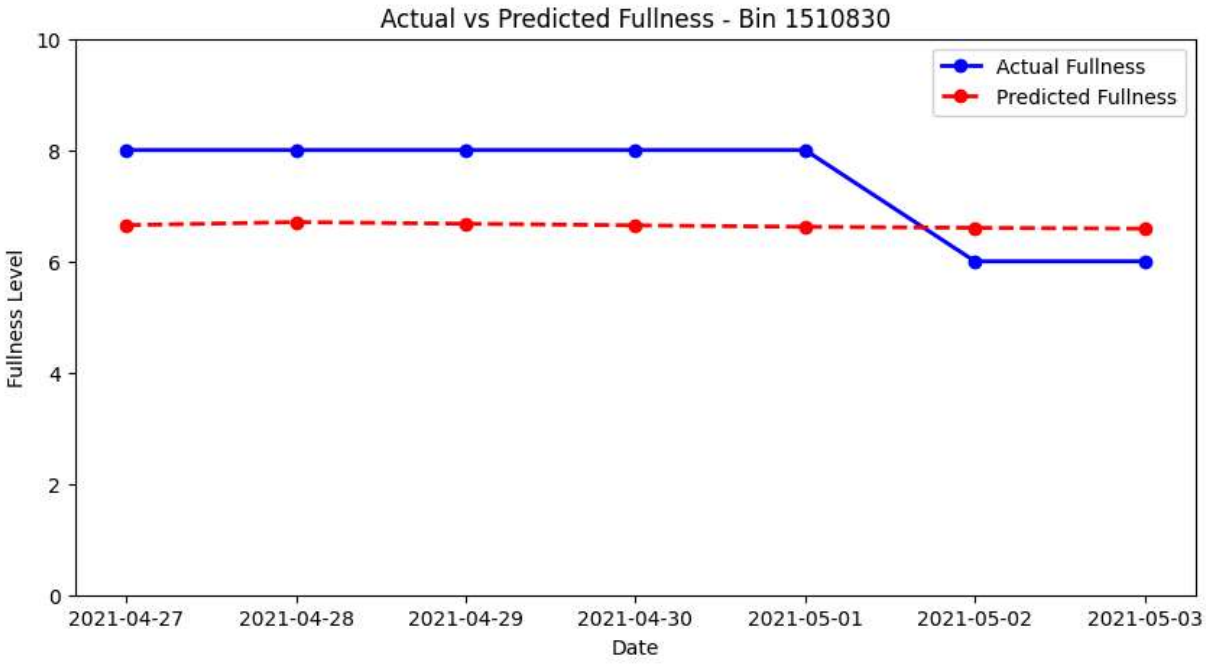
```
        y.append(series[i + time_steps])
    return np.array(X), np.array(y)
```

In [11]:
```
results = {}
```

In [12]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
for bin_id in bins_to_forecast:
    bin_train = train_data[train_data['Bin ID'] == bin_id].sort_values(by='timestam

    # Normalize fullness values (scaling)
    scaler = MinMaxScaler()
    fullness_train = scaler.fit_transform(bin_train[['Fullness']])

    # Prepare training data
    time_steps = 7
    X_train, y_train = prepare_data(fullness_train, time_steps)

    # Reshape input for LSTM
    X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))

    # Build improved LSTM model
    model = Sequential([
        LSTM(100, activation='relu', return_sequences=True, input_shape=(time_steps
        Dropout(0.2),
        LSTM(100, activation='relu'),
        Dropout(0.2),
        Dense(1)
    ])
    model.compile(optimizer='adam', loss='mse')

    # Train the model with more epochs
    model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=0)

    # Predict the next 7 days
    predictions = []
    last_sequence = fullness_train[-time_steps:].reshape(1, time_steps, 1)

    for _ in range(7):
        pred = model.predict(last_sequence)
        predictions.append(pred[0, 0])
        last_sequence = np.append(last_sequence[:, 1:, :], [[[pred[0, 0]]]], axis=1

    # Inverse transform predictions
    predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))

    # Define prediction period: 27th April to 3rd May
    prediction_start = pd.to_datetime("2021-04-27")
    prediction_end = pd.to_datetime("2021-05-03")
    prediction_dates = pd.date_range(start=prediction_start, end=prediction_end, fr

    # Use the full dataset for this bin and ensure 'timestamp' is datetime
    bin_full = df[df['Bin ID'] == bin_id].sort_values(by='timestamp')
    bin_full['timestamp'] = pd.to_datetime(bin_full['timestamp'])
    bin_full = bin_full.set_index('timestamp')
```

```
    # Resample to daily frequency, taking the last available 'Fullness' value of ea
    daily_actual = bin_full['Fullness'].resample('D').last()

    # Reindex to the complete prediction period and fill missing values using both
    daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fi
    prediction_actual = daily_actual.values.reshape(-1, 1)


    # Calculate RMSE using the actual values for the prediction period
    rmse = np.sqrt(mean_squared_error(prediction_actual, predictions))
    results[bin_id] = rmse


    # Generate timestamps for the prediction period
    prediction_dates = pd.date_range(start="2021-04-27", end="2021-05-03", freq='D'

    # Plot actual vs predicted values for the prediction period
    plt.figure(figsize=(10, 5))
    plt.plot(prediction_dates, prediction_actual, 'bo-', label="Actual Fullness", l
    plt.plot(prediction_dates, predictions, 'ro--', label="Predicted Fullness", lin
    plt.ylim(0, 10)
    plt.title(f"Actual vs Predicted Fullness - Bin {bin_id}")
    plt.xlabel("Date")
    plt.ylabel("Fullness Level")
    plt.legend()
    plt.show()


rmse_df = pd.DataFrame.from_dict(results, orient='index', columns=['RMSE'])
rmse_df.index.name = "Bin ID"


# Print RMSE values
print(rmse_df)
```

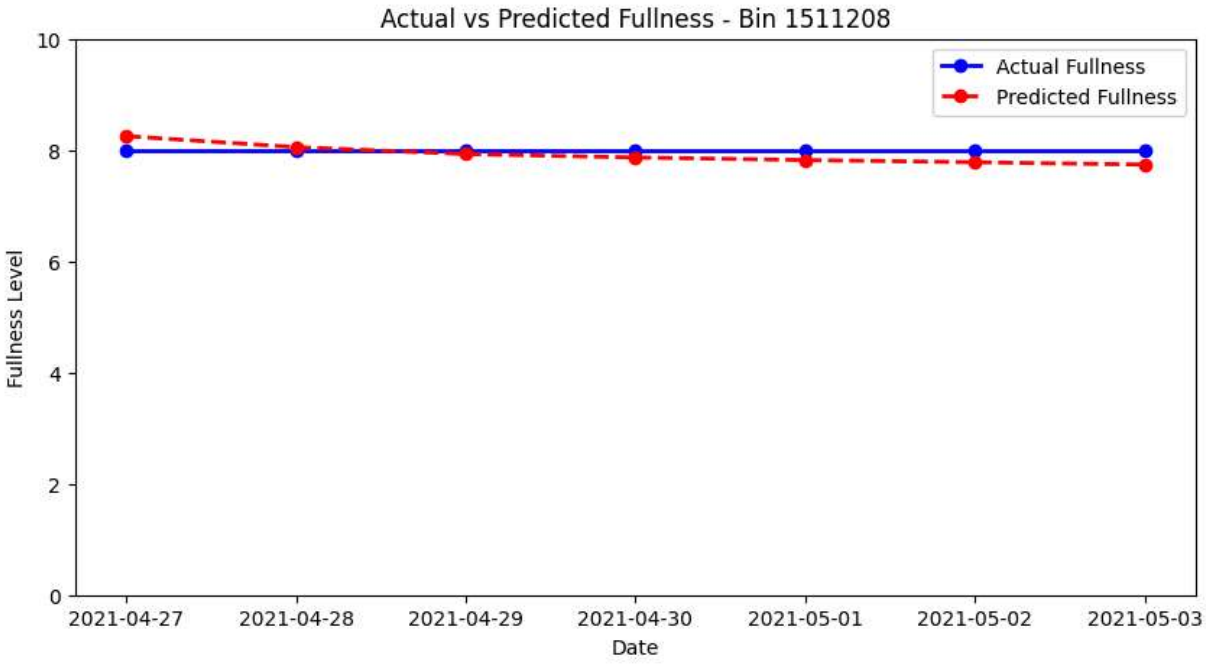
```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\l
ayers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument
to a layer. When using Sequential models, prefer using an `Input(shape)` object as t
he first layer in the model instead.
  super().__init__(**kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3339752100.py:56: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
```

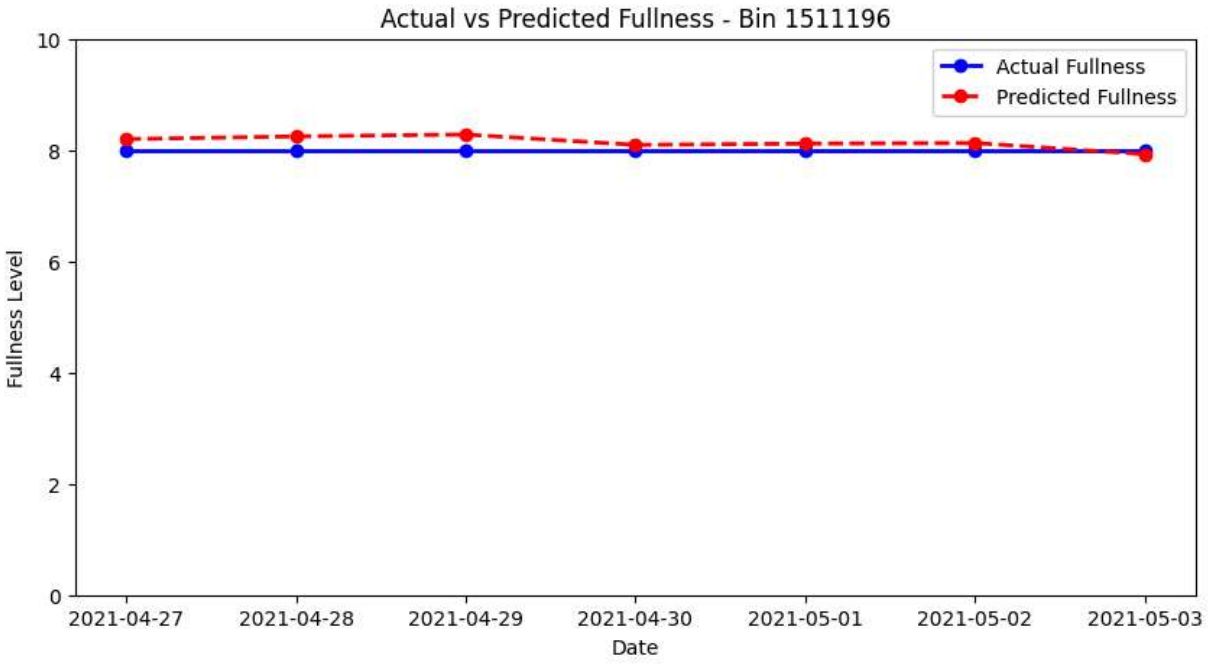
```
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3339752100.py:56: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fillna(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 133ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
```

```
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3339752100.py:56: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fillna(method='bfill')
```



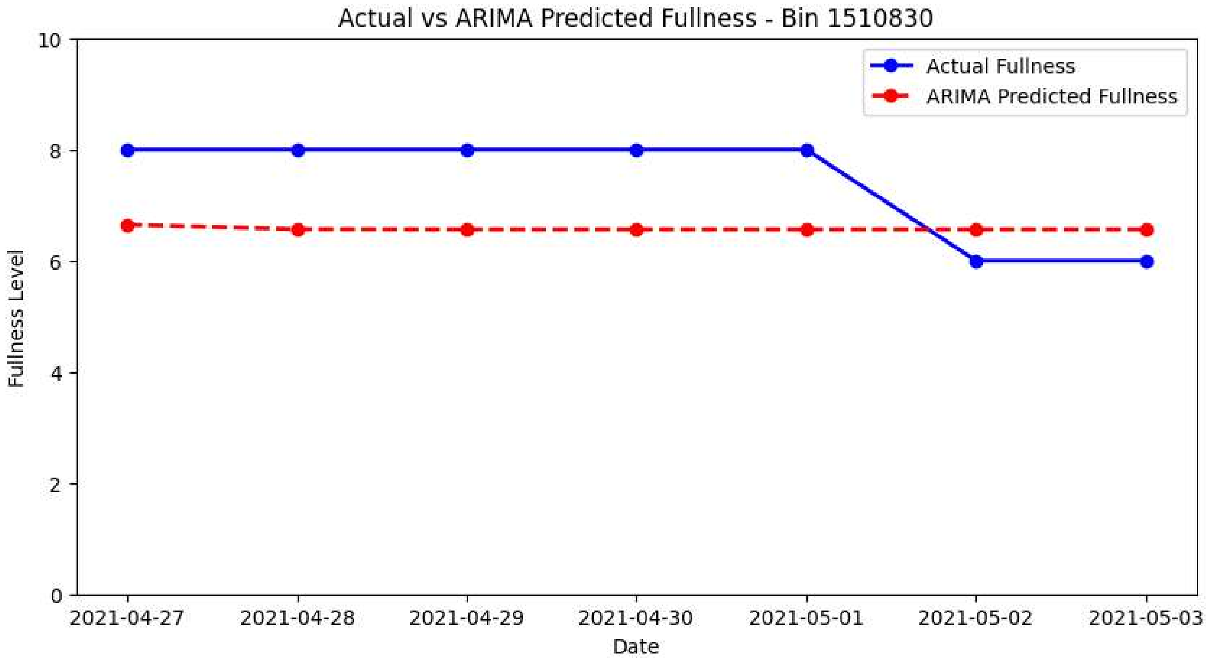

```
            RMSE
Bin ID
1510830  1.176787
1511208  0.183760
1511196  0.177783
```

ARIMA

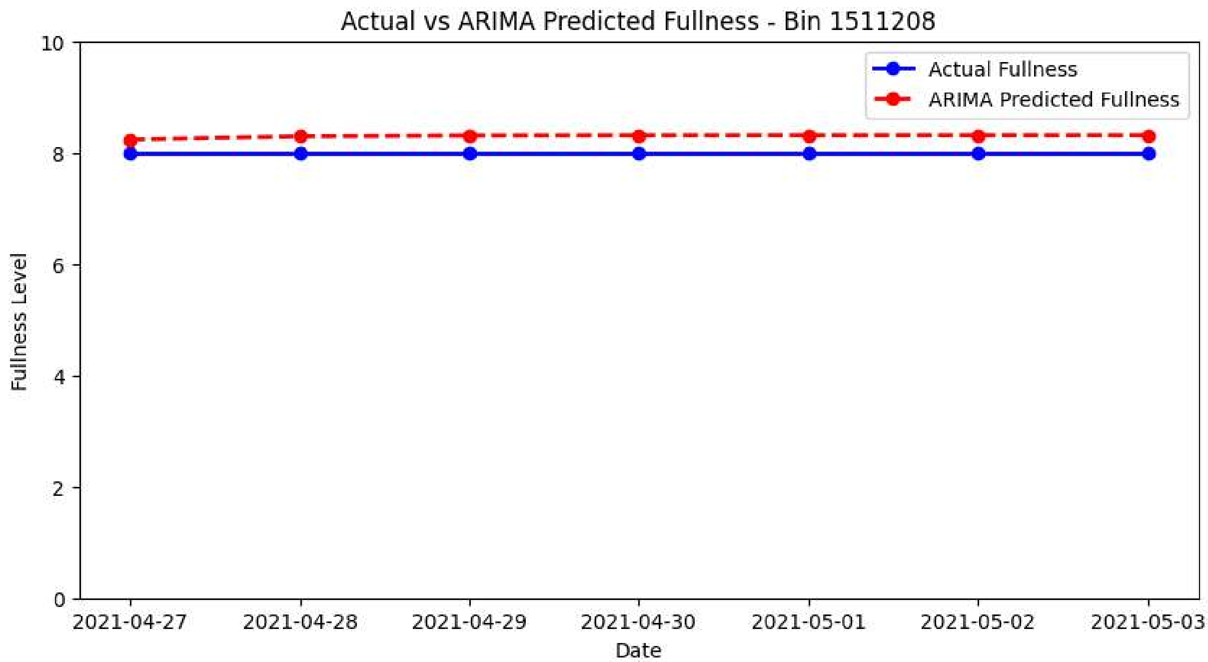
In [13]:
```python
from statsmodels.tsa.arima.model import ARIMA

results_arima = {}
for bin_id in bins_to_forecast:
    # Prepare training data for this bin
    bin_train = train_data[train_data['Bin ID'] == bin_id].sort_values(by='timestam
    bin_train['timestamp'] = pd.to_datetime(bin_train['timestamp'])
    bin_train = bin_train.set_index('timestamp')

    ts = bin_train['Fullness']

    # Fit an ARIMA model (order chosen here is (1,1,1); adjust if needed)
    model_arima = ARIMA(ts, order=(1,1,1))
    model_arima_fit = model_arima.fit()

    # Forecast next 7 days
    forecast_arima = model_arima_fit.forecast(steps=7)
    predictions_arima = forecast_arima.values.reshape(-1, 1)

    # Define prediction period: 27th April to 3rd May
    prediction_start = pd.to_datetime("2021-04-27")
    prediction_end = pd.to_datetime("2021-05-03")
    prediction_dates = pd.date_range(start=prediction_start, end=prediction_end, fr

    # Use the full dataset to get actual values for this bin and prediction period
    bin_full = df[df['Bin ID'] == bin_id].sort_values(by='timestamp')
```

```
    bin_full['timestamp'] = pd.to_datetime(bin_full['timestamp'])
    bin_full = bin_full.set_index('timestamp')
    daily_actual = bin_full['Fullness'].resample('D').last()
    daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fi
    prediction_actual = daily_actual.values.reshape(-1, 1)

    # Calculate RMSE for ARIMA predictions
    rmse_arima = np.sqrt(mean_squared_error(prediction_actual, predictions_arima))
    results_arima[bin_id] = rmse_arima

    # Plot Actual vs ARIMA Predicted Fullness
    plt.figure(figsize=(10, 5))
    plt.plot(prediction_dates, prediction_actual, 'bo-', label="Actual Fullness", l
    plt.plot(prediction_dates, predictions_arima, 'ro--', label="ARIMA Predicted Fu
    plt.ylim(0, 10)
    plt.title(f"Actual vs ARIMA Predicted Fullness - Bin {bin_id}")
    plt.xlabel("Date")
    plt.ylabel("Fullness Level")
    plt.legend()
    plt.show()

# Display ARIMA RMSE values
rmse_arima_df = pd.DataFrame.from_dict(results_arima, orient='index', columns=['RMS
rmse_arima_df.index.name = "Bin ID"
print("ARIMA RMSE:")
print(rmse_arima_df)
```

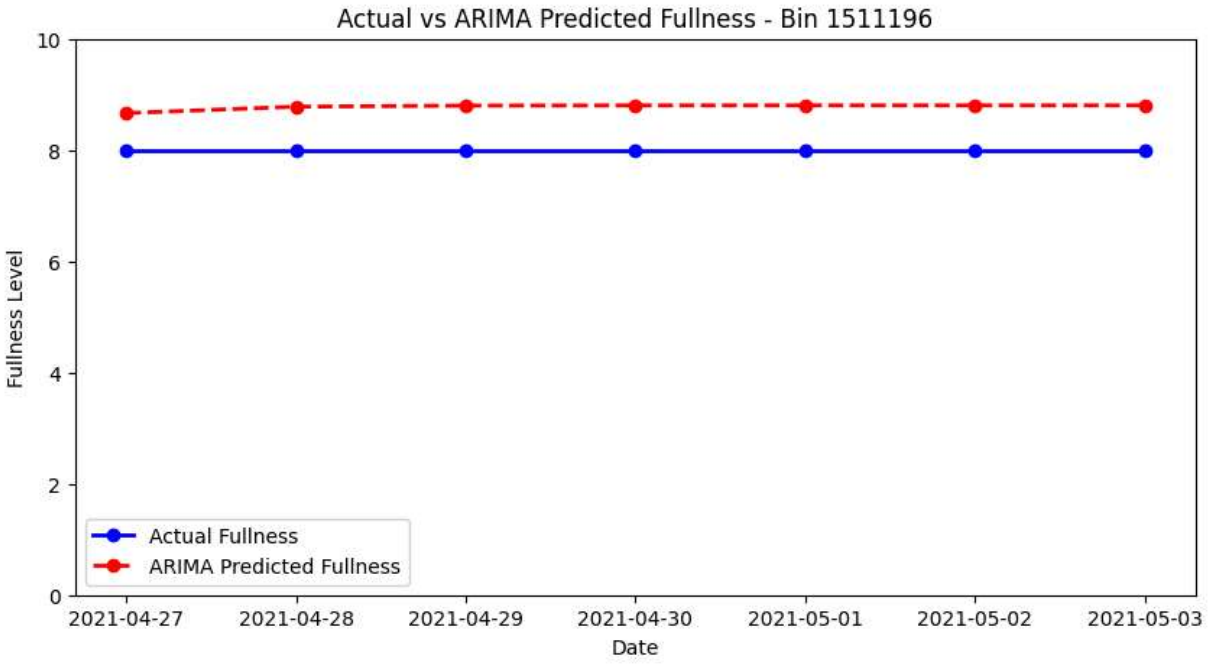
```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\82157723.py:30: FutureWarning: Ser
ies.fillna with 'method' is deprecated and will raise in a future version. Use obj.f
fill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\82157723.py:30: FutureWarning: Ser
ies.fillna with 'method' is deprecated and will raise in a future version. Use obj.f
fill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\82157723.py:30: FutureWarning: Ser
ies.fillna with 'method' is deprecated and will raise in a future version. Use obj.f
fill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```



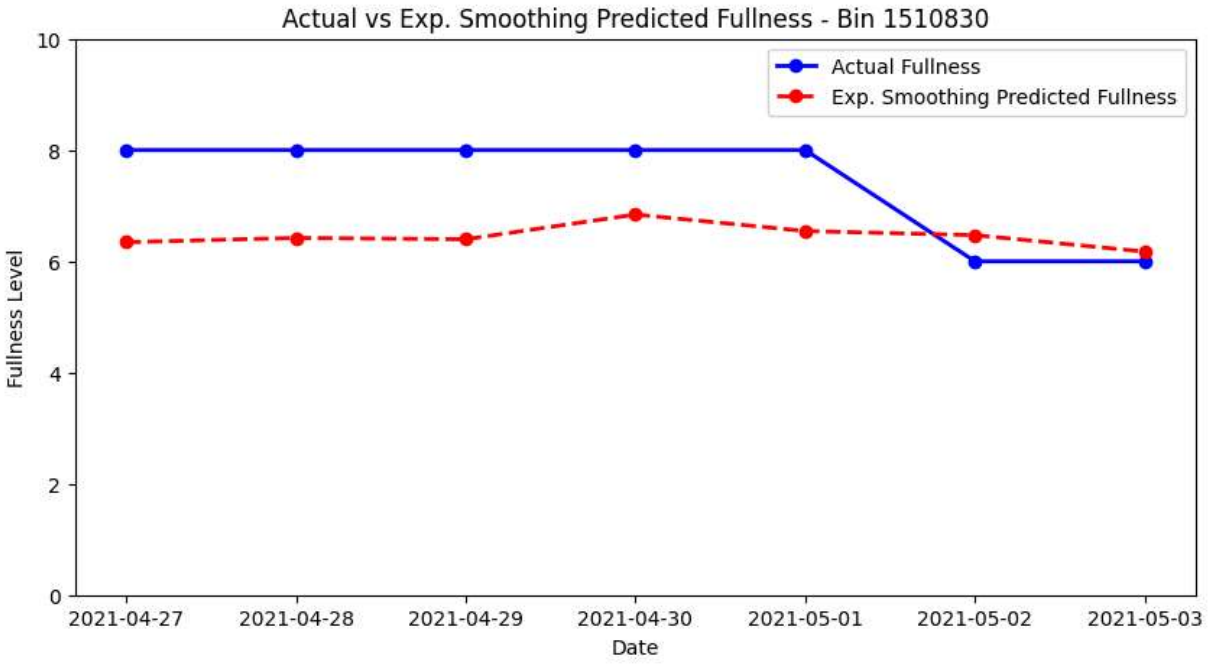

```
ARIMA RMSE:
             RMSE
Bin ID
1510830  1.237859
1511208  0.298722
1511196  0.781512
```

Exponential Smoothing

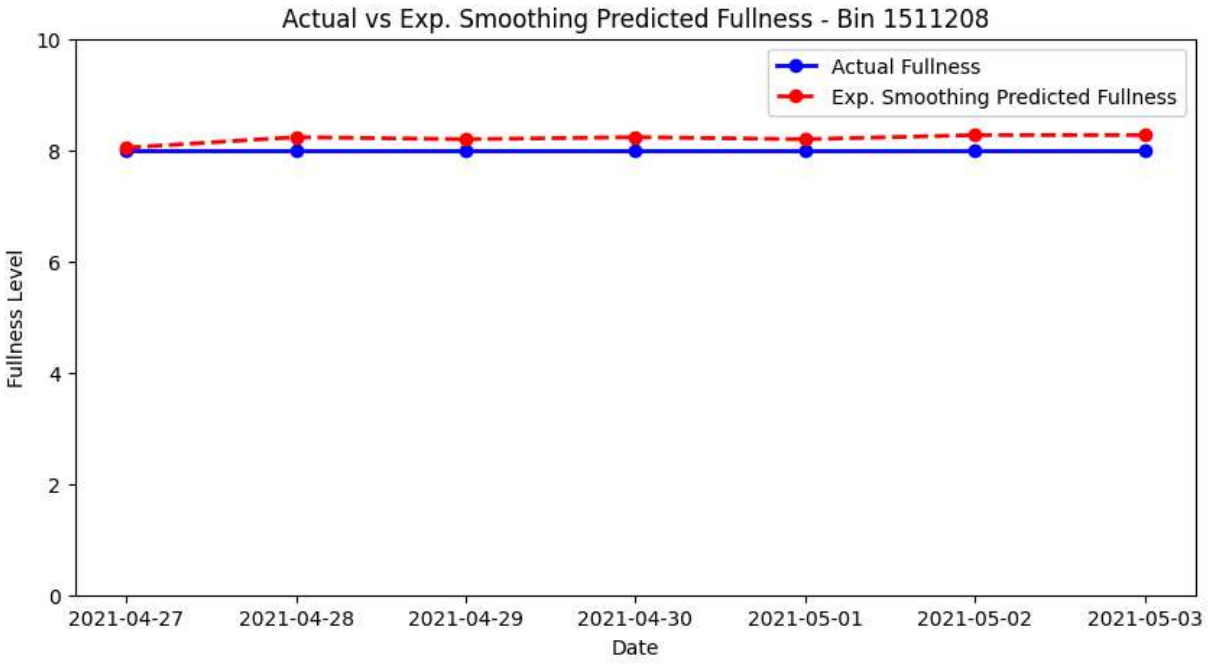
In [14]:
```python
from statsmodels.tsa.holtwinters import ExponentialSmoothing

results_exp = {}
for bin_id in bins_to_forecast:
    # Prepare training data for this bin
    bin_train = train_data[train_data['Bin ID'] == bin_id].sort_values(by='timestam
    bin_train['timestamp'] = pd.to_datetime(bin_train['timestamp'])
    bin_train = bin_train.set_index('timestamp')

    ts = bin_train['Fullness']

    # Fit an Exponential Smoothing model with additive trend and seasonality (seaso
    model_exp = ExponentialSmoothing(ts, trend='add', seasonal='add', seasonal_peri
    model_exp_fit = model_exp.fit()

    # Forecast next 7 days
    forecast_exp = model_exp_fit.forecast(steps=7)
    predictions_exp = forecast_exp.values.reshape(-1, 1)

    # Define prediction period: 27th April to 3rd May
    prediction_start = pd.to_datetime("2021-04-27")
    prediction_end = pd.to_datetime("2021-05-03")
    prediction_dates = pd.date_range(start=prediction_start, end=prediction_end, fr

    # Use the full dataset to get actual values for this bin and prediction period
```

```
    bin_full = df[df['Bin ID'] == bin_id].sort_values(by='timestamp')
    bin_full['timestamp'] = pd.to_datetime(bin_full['timestamp'])
    bin_full = bin_full.set_index('timestamp')
    daily_actual = bin_full['Fullness'].resample('D').last()
    daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fi
    prediction_actual = daily_actual.values.reshape(-1, 1)

    # Calculate RMSE for Exponential Smoothing predictions
    rmse_exp = np.sqrt(mean_squared_error(prediction_actual, predictions_exp))
    results_exp[bin_id] = rmse_exp

    # Plot Actual vs Exponential Smoothing Predicted Fullness
    plt.figure(figsize=(10, 5))
    plt.plot(prediction_dates, prediction_actual, 'bo-', label="Actual Fullness", l
    plt.plot(prediction_dates, predictions_exp, 'ro--', label="Exp. Smoothing Predi
    plt.ylim(0, 10)
    plt.title(f"Actual vs Exp. Smoothing Predicted Fullness - Bin {bin_id}")
    plt.xlabel("Date")
    plt.ylabel("Fullness Level")
    plt.legend()
    plt.show()

# Display Exponential Smoothing RMSE values
rmse_exp_df = pd.DataFrame.from_dict(results_exp, orient='index', columns=['RMSE'])
rmse_exp_df.index.name = "Bin ID"
print("Exponential Smoothing RMSE:")
print(rmse_exp_df)
```

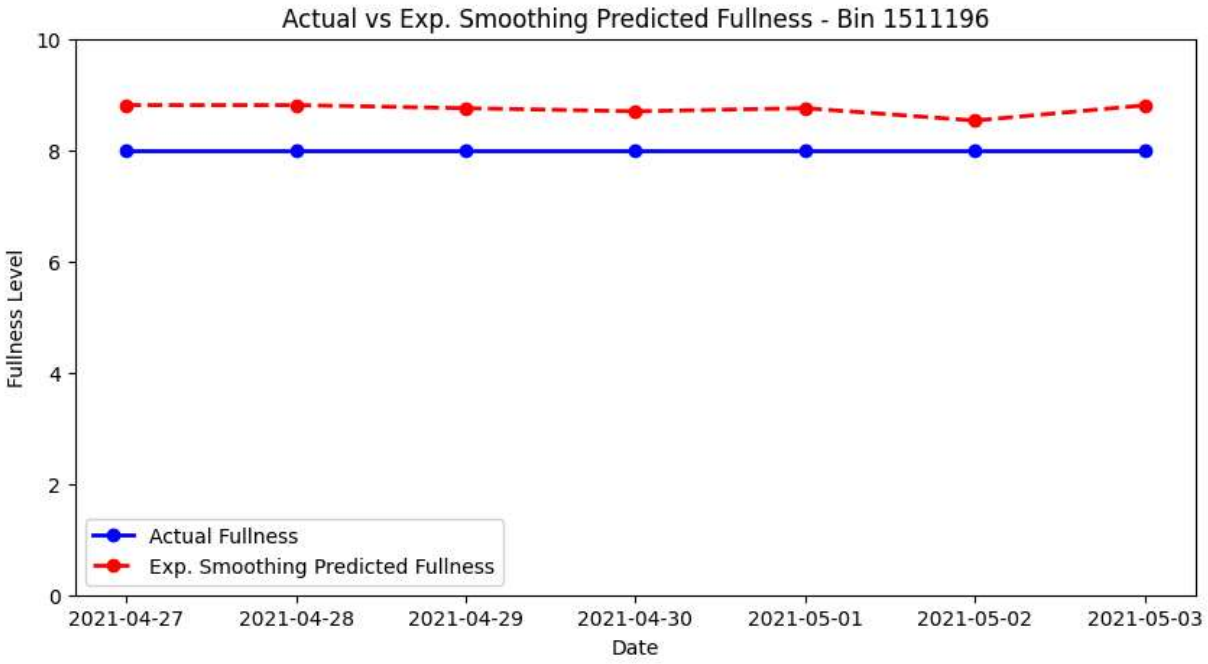
```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\2295670061.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\2295670061.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\2295670061.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```



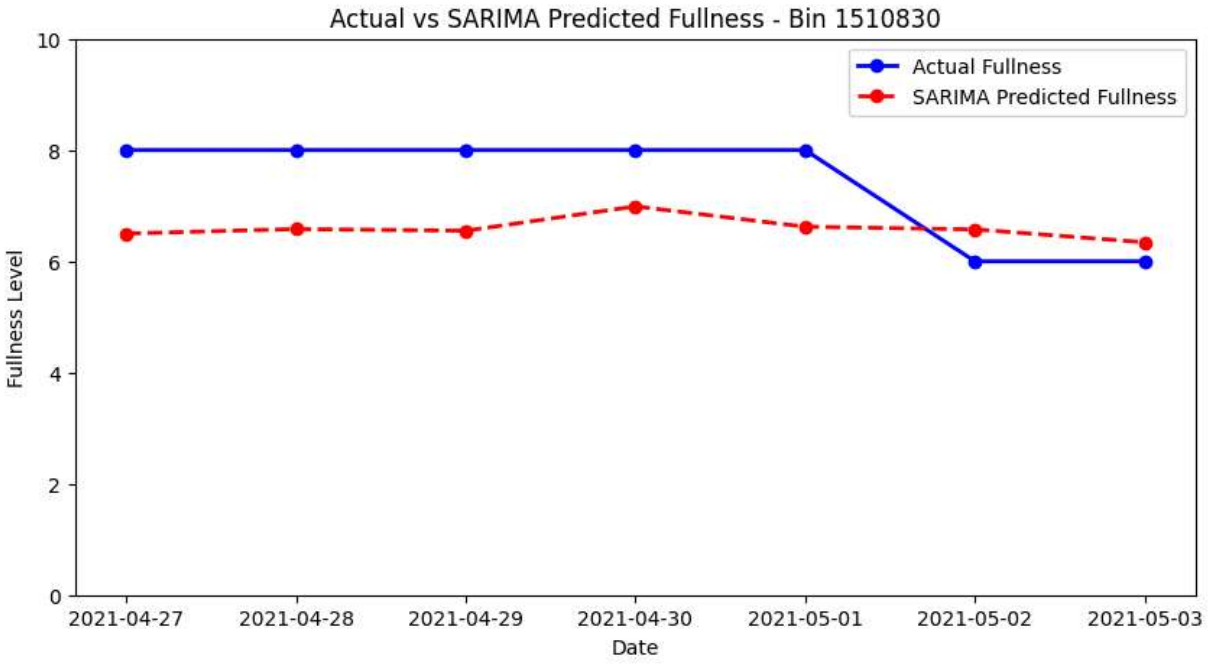

```
Exponential Smoothing RMSE:
             RMSE
Bin ID
1510830  1.283014
1511208  0.217160
1511196  0.742272
```

SARIMA

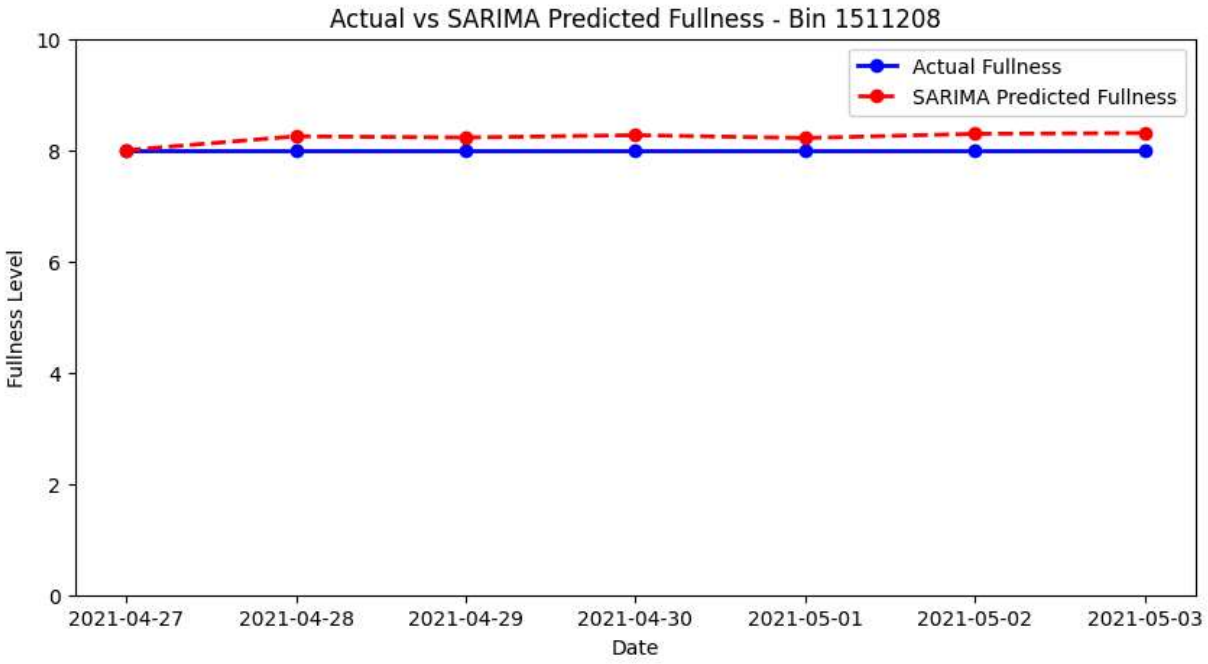
In [15]:
```python
from statsmodels.tsa.statespace.sarimax import SARIMAX

results_sarima = {}
for bin_id in bins_to_forecast:
    # Prepare training data for this bin
    bin_train = train_data[train_data['Bin ID'] == bin_id].sort_values(by='timestam
    bin_train['timestamp'] = pd.to_datetime(bin_train['timestamp'])
    bin_train = bin_train.set_index('timestamp')

    ts = bin_train['Fullness']

    # Fit a SARIMA model (non-seasonal order (1,1,1) and seasonal_order (1,1,1,7))
    model_sarima = SARIMAX(ts, order=(1,1,1), seasonal_order=(1,1,1,7))
    model_sarima_fit = model_sarima.fit(disp=False)

    # Forecast the next 7 days
    forecast_sarima = model_sarima_fit.forecast(steps=7)
    predictions_sarima = forecast_sarima.values.reshape(-1, 1)

    # Define prediction period: 27th April to 3rd May
    prediction_start = pd.to_datetime("2021-04-27")
    prediction_end = pd.to_datetime("2021-05-03")
    prediction_dates = pd.date_range(start=prediction_start, end=prediction_end, fr

    # Use the full dataset to get actual values for the prediction period
```

```
    bin_full = df[df['Bin ID'] == bin_id].sort_values(by='timestamp')
    bin_full['timestamp'] = pd.to_datetime(bin_full['timestamp'])
    bin_full = bin_full.set_index('timestamp')
    daily_actual = bin_full['Fullness'].resample('D').last()
    daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').fi
    prediction_actual = daily_actual.values.reshape(-1, 1)

    # Calculate RMSE using the actual values for the prediction period
    rmse_sarima = np.sqrt(mean_squared_error(prediction_actual, predictions_sarima)
    results_sarima[bin_id] = rmse_sarima

    # Plot actual vs predicted values for the prediction period
    plt.figure(figsize=(10, 5))
    plt.plot(prediction_dates, prediction_actual, 'bo-', label="Actual Fullness", l
    plt.plot(prediction_dates, predictions_sarima, 'ro--', label="SARIMA Predicted
    plt.ylim(0, 10)
    plt.title(f"Actual vs SARIMA Predicted Fullness - Bin {bin_id}")
    plt.xlabel("Date")
    plt.ylabel("Fullness Level")
    plt.legend()
    plt.show()

# Display SARIMA RMSE values
rmse_sarima_df = pd.DataFrame.from_dict(results_sarima, orient='index', columns=['R
rmse_sarima_df.index.name = "Bin ID"
print("SARIMA RMSE:")
print(rmse_sarima_df)
```

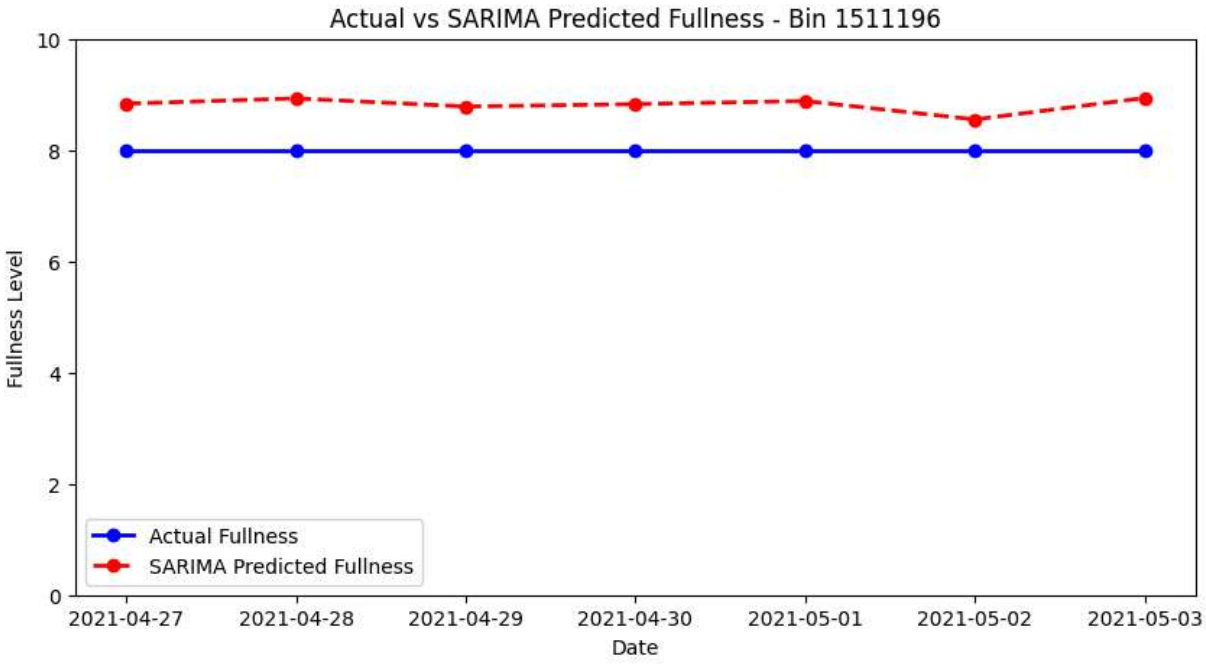
```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3763544895.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3763544895.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```




```
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it has
no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: ValueWarning: No supported index is available. Predictio
n results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\dhyey\AppData\Local\Programs\Python\Python312\Lib\site-packages\statsmodels
\tsa\base\tsa_model.py:836: FutureWarning: No supported index is available. In the n
ext version, calling this method in a model without a supported index will result in
an exception.
  return get_prediction_index(
C:\Users\dhyey\AppData\Local\Temp\ipykernel_44576\3763544895.py:30: FutureWarning: S
eries.fillna with 'method' is deprecated and will raise in a future version. Use ob
j.ffill() or obj.bfill() instead.
  daily_actual = daily_actual.reindex(prediction_dates).fillna(method='ffill').filln
a(method='bfill')
```



SARIMA RMSE:

Bin ID	RMSE
1510830	1.179961
1511208	0.242028
1511196	0.829326