In [1]:

```python
import numpy as np
import pandas as pd
import random
from datetime import datetime, timedelta

# Define the states in the logistics process
states = ["Sorting", "Quality Check", "Packing", "Labeling", "Dispatch"]

# Number of synthetic packages
num_packages = 1000

# Starting date for timestamps
start_date = datetime(2025, 1, 1)

# Adjusted transition probabilities (bias certain states)
def generate_transition_probabilities(state):
    probs = np.zeros(len(states))

    if state == "Sorting":
        probs = [0.2, 0.4, 0.3, 0.1, 0]  # More likely to move to Quality Check or Packing
    elif state == "Quality Check":
        probs = [0.3, 0.3, 0.1, 0.2, 0.1]  # Higher probability of looping
    elif state == "Packing":
        probs = [0.1, 0.2, 0.3, 0.3, 0.1]  # Balanced transition
    elif state == "Labeling":
        probs = [0, 0.1, 0.2, 0.4, 0.3]  # Faster exit to Dispatch
    elif state == "Dispatch":
        probs = [0, 0, 0, 0, 1]  # Absorbing state

    return np.array(probs) / np.sum(probs)  # Normalize

# Adjusted elapsed time ranges
elapsed_time_ranges = {
    "Sorting": (5, 20),
    "Quality Check": (30, 60),  # Increased range to reflect longer times
    "Packing": (10, 30),
    "Labeling": (5, 15),  # Shorter range to exit faster
}

# Generate synthetic dataset
data = []
for i in range(num_packages):
```

Home

Manage

```
    package_id = f"P{i+1}"
    current_state = "Sorting"
    current_time = start_date

    while current_state != "Dispatch":
        # Generate new transition probabilities based on current state
        transition_probs = generate_transition_probabilities(current_state)

        # Randomly choose the next state
        next_state = random.choices(states, weights=transition_probs, k=1)[0]

        # Generate elapsed time based on the state's range
        elapsed_time = random.randint(*elapsed_time_ranges[current_state])
        current_time += timedelta(minutes=elapsed_time)

        # Store data
        data.append([package_id, current_state, next_state, current_time, elapsed_time])
        current_state = next_state

# Convert dataset to DataFrame
df = pd.DataFrame(data, columns=["Package ID", "From State", "To State", "Timestamp", "Elapsed Time"])

# Display first few rows of the synthetic data
print("Synthetic Data:")
print(df.head())

# Save synthetic data
df.to_excel("synthetic_logistics_data.xlsx", index=False)
print("\nSynthetic data saved as 'synthetic_logistics_data.xlsx'.")
```

```
Synthetic Data:
  Package ID     From State       To State            Timestamp  Elapsed Time
0         P1        Sorting  Quality Check 2025-01-01 00:17:00            17
1         P1  Quality Check       Labeling 2025-01-01 01:17:00            60
2         P1       Labeling        Packing 2025-01-01 01:24:00             7
3         P1        Packing        Packing 2025-01-01 01:46:00            22
4         P1        Packing  Quality Check 2025-01-01 01:57:00            11

Synthetic data saved as 'synthetic_logistics_data.xlsx'.
```

In [2]:

```
# Load synthetic data
df = pd.read_excel("synthetic_logistics_data.xlsx")

# Create the transition frequency matrix (count of transitions)
```

```
transition_counts = pd.crosstab(df["From State"], df["To State"], rownames=["From State"], colnames=["To State"])

# Ensure all states are present
states = ["Sorting", "Quality Check", "Packing", "Labeling", "Dispatch"]
transition_counts = transition_counts.reindex(index=states, columns=states, fill_value=0)

# Normalize to get transition probabilities (row-wise)
transition_matrix = transition_counts.div(transition_counts.sum(axis=1), axis=0).fillna(0)

# Ensure "Dispatch" is an absorbing state
transition_matrix.loc["Dispatch"] = 0
transition_matrix.loc["Dispatch", "Dispatch"] = 1

# Display transition matrix
print("\nTransition Matrix:")
print(transition_matrix)

# Save transition matrix
transition_matrix.to_excel("transition_matrix_synthetic.xlsx")
print("\nTransition matrix saved as 'transition_matrix_synthetic.xlsx'.")
```

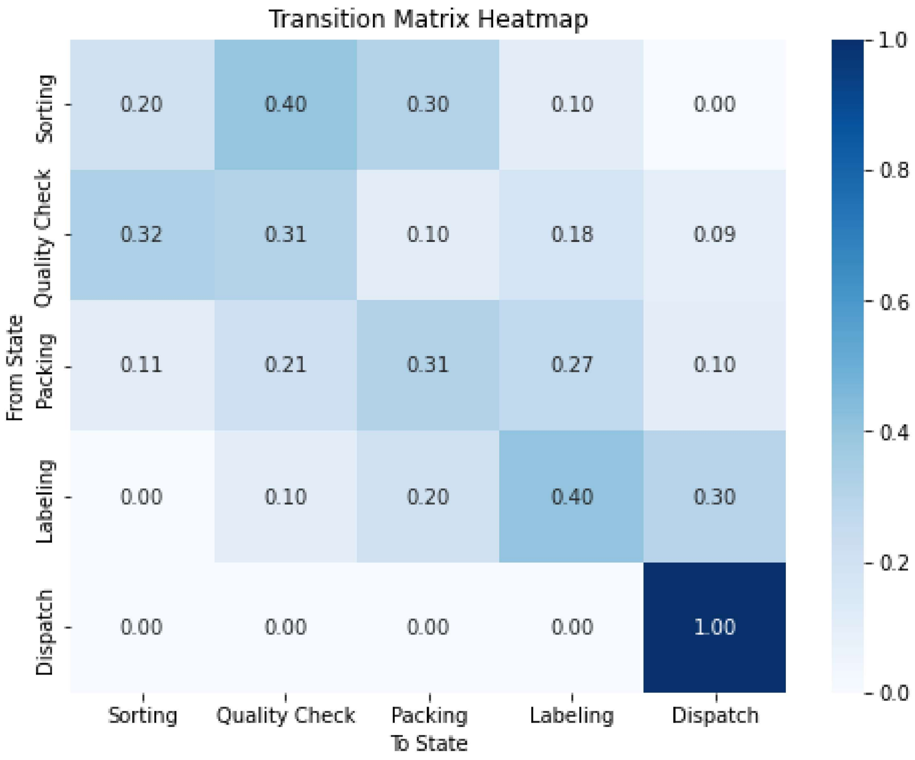
Home

```
Transition Matrix:
To State        Sorting  Quality Check   Packing  Labeling  Dispatch
From State
Sorting        0.195688       0.395522  0.304726  0.104063  0.000000
Quality Check  0.321381       0.305002  0.103586  0.178398  0.091633
Packing        0.108354       0.207089  0.312911  0.269367  0.102278
Labeling       0.000000       0.104916  0.196655  0.398885  0.299544
Dispatch       0.000000       0.000000  0.000000  0.000000  1.000000

Transition matrix saved as 'transition_matrix_synthetic.xlsx'.
```

Manage

In [3]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot the transition matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(transition_matrix, annot=True, cmap="Blues", fmt=".2f", cbar=True, xticklabels=states, yticklabels=states)
plt.title("Transition Matrix Heatmap")
plt.xlabel("To State")
plt.ylabel("From State")
plt.show()
```



Home

In [4]:

```
# Load transition matrix
transition_matrix = pd.read_excel("transition_matrix_synthetic.xlsx", index_col=0)

# Define transient states (exclude Dispatch)
transient_states = ["Sorting", "Quality Check", "Packing", "Labeling"]
P = transition_matrix.loc[transient_states, transient_states].to_numpy()

# Compute Fundamental Matrix: S = (I - P)^-1
I = np.eye(len(P))
I_minus_P = I - P

try:
    S = np.linalg.inv(I_minus_P)  # Inverse of (I - P)
except np.linalg.LinAlgError:
    print("Matrix inversion error. The matrix might be singular.")
    S = None
if S is not None:
    # Compute mean time spent in each transient state (sum of rows in N)
```

Manage

```
    mean_time_spent = S.sum(axis=1)

    print('The matrix S')
    print(S)
    print()

    # Convert to pandas Series
    mean_time_spent_series = pd.Series(mean_time_spent, index=transient_states)

    # Save results
    mean_time_spent_series.to_excel("mean_time_in_transient_states.xlsx")
    print("\nMean time saved as 'mean_time_in_transient_states.xlsx'.")

    # Display results
    print("\nMean Time Spent in Transient States (steps):")
    print(mean_time_spent_series)
```

Home

```
The matrix S
[[2.412      2.259      1.975      1.973     ]
 [1.42119839 3.01127806 1.6180106  1.86476372]
 [1.03926175 1.68615361 2.71324844 1.89616641]
 [0.58804522 1.0772025  1.17004097 2.60937492]]


Mean time saved as 'mean_time_in_transient_states.xlsx'.

Mean Time Spent in Transient States (steps):
Sorting          8.619000
Quality Check    7.915251
Packing          7.334830
Labeling         5.444664
dtype: float64
```

Manage

```
In [ ]:
```