Waste Management Optimization Using IoT

- **IoT sensor simulation**
- **AI-based bin overflow prediction**
- **Route optimization (mocked)**
- **Data encryption (simulated)**
- **Dashboard interface (console-based output)**

Here is a Python file structure for your project:

```
# waste_management_system.py
import random
import datetime
from cryptography.fernet import Fernet
import matplotlib.pyplot as plt
# --- Simulated Sensor Data Generator ---
def generate_sensor_data(bin_id):
    return {
        "bin_id": bin_id,
        "fill_level": random.randint(0, 100),  # percent
        "gas_level": random.uniform(0, 10),  # ppm
        "timestamp": datetime.datetime.now().isoformat()
    }
# --- Simple AI Model for Overflow Prediction ---
def predict_overflow(fill_level, threshold=80):
    return fill_level > threshold
# --- Route Optimization (mock) ---
def optimize_routes(bin_data):
    full_bins = [b for b in bin_data if predict_overflow(b["fill_level"])]
    return sorted(full_bins, key=lambda x: x["fill_level"], reverse=True)
```

```
# --- Encrypt Data ---
def encrypt_data(data, key):
   fernet = Fernet(key)
   return fernet.encrypt(str(data).enc♥de())
# --- Display Dashb♥ard (C♥ns♥le-based f♥r simplicity) ---
def display_dashb♥ard(bin_data):
   print("\n--- Dashb♥ard ---")
   f♥r b in bin_data:
      status = "❤verfl♥w Risk!" if predict_♥verfl♥w(b["fill_level"]) else "N♥rmal"
      print(f"Bin {b['bin_id']} - Fill: {b['fill_level']}% | Gas: {b['gas_level']:.2f} ppm | Status: {status}")
 # Visualizati♥n
   bin_ids = [b["bin_id"] f♥r b in bin_data]
   levels = [b["fill_level"] f♥r b in bin_data]
   plt.bar(bin_ids, levels)
   plt.xlabel("Bin ID")
   plt.ylabel("Fill Level (%)")
   plt.title("Bin Fill Levels")
   plt.sh♥w()
# --- Main Executi♥n ---
def main():
   # Key generati♥n f♥r encrypti♥n
   key = Fernet.generate_key()
 # Simulate 5 bins
   bins = [generate_sens♥r_data(f"Bin{i}") f♥r i in range(1, 6)]
# Display dashb♥ard
   display_dashb♥ard(bins)
```

```
    # Optimize route
    route = optimize_routes(bins)
    print("\nOptimized Route for Collection:")
    for b in route:
        print(f"- {b['bin_id']} (Fill: {b['fill_level']}%)")
# Encrypt data for secure transmission
    for b in bins:
        encrypted = encrypt_data(b, key)
        print(f"Encrypted data for {b['bin_id']}: {encrypted[:30]}...")
if __name__ == "__main__":
    main()
```

Requirements

Install dependencies using:

```
pip install cryptography matplotlib
```

Next Steps

If you want:

- Integration with **real IoT devices**
- **A web-based dashboard**
- Real **map-based route optimization (e.g., with Google Maps API)**