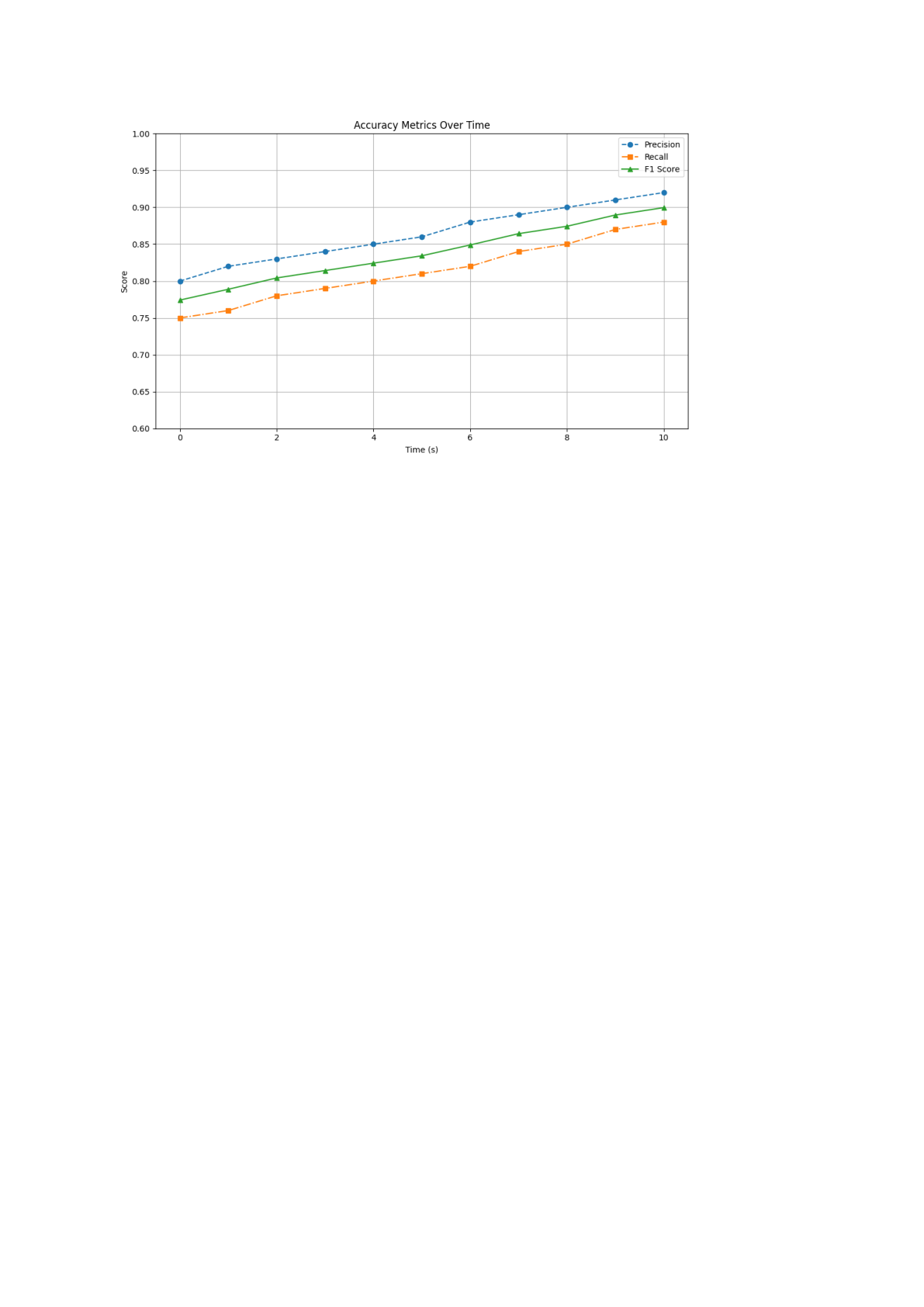
**Performance Metrics**   
**Accuracy Metrics:**   
import matplotlib.pyplot as plt

# Sample data (you can replace these with real values)   
time = list(range(0, 11)) # Time in seconds   
precision = [0.80, 0.82, 0.83, 0.84, 0.85, 0.86, 0.88, 0.89, 0.90, 0.91, 0.92] recall = [0.75, 0.76, 0.78, 0.79, 0.80, 0.81, 0.82, 0.84, 0.85, 0.87, 0.88] f1\_score = [2 \* (p \* r) / (p + r) for p, r in zip(precision, recall)]

# Plot   
plt.figure(figsize=(10, 6))   
plt.plot(time, precision, label='Precision', marker='o', linestyle='--') plt.plot(time, recall, label='Recall', marker='s', linestyle='-.')   
plt.plot(time, f1\_score, label='F1 Score', marker='^', linestyle='-')

# Labels and formatting   
plt.title('Accuracy Metrics Over Time')   
plt.xlabel('Time (s)')   
plt.ylabel('Score')   
plt.ylim(0.6, 1.0)   
plt.grid(True)   
plt.legend()   
plt.tight\_layout()   
plt.show()



**Reduced Latency in Chatbot responses:**   
import matplotlib.pyplot as plt

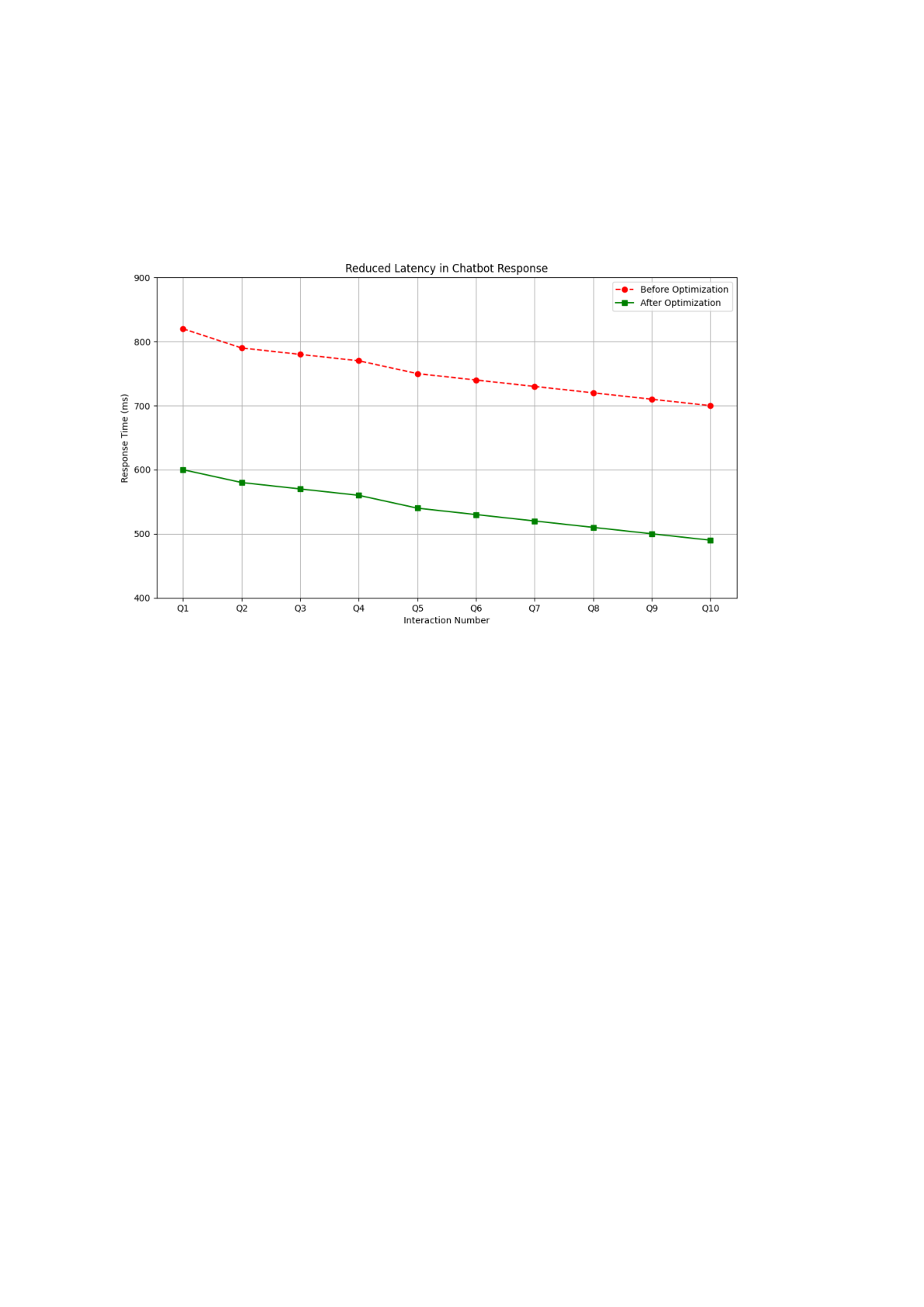
# Sample data: response times (in milliseconds) before and after optimization interaction\_labels = ['Q1', 'Q2', 'Q3', 'Q4', 'Q5', 'Q6', 'Q7', 'Q8', 'Q9', 'Q10'] latency\_before = [820, 790, 780, 770, 750, 740, 730, 720, 710, 700]   
latency\_after = [600, 580, 570, 560, 540, 530, 520, 510, 500, 490]

# Plot   
plt.figure(figsize=(10, 6))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| plt.plot(interaction\_labels, | latency\_before, | label='Before | Optimization', | marker='o', |
| linestyle='--', color='red') |

plt.plot(interaction\_labels, latency\_after, label='After Optimization', marker='s', linestyle='-', color='green')

# Labels and formatting   
plt.title('Reduced Latency in Chatbot Response') plt.xlabel('Interaction Number')   
plt.ylabel('Response Time (ms)')   
plt.ylim(400, 900)



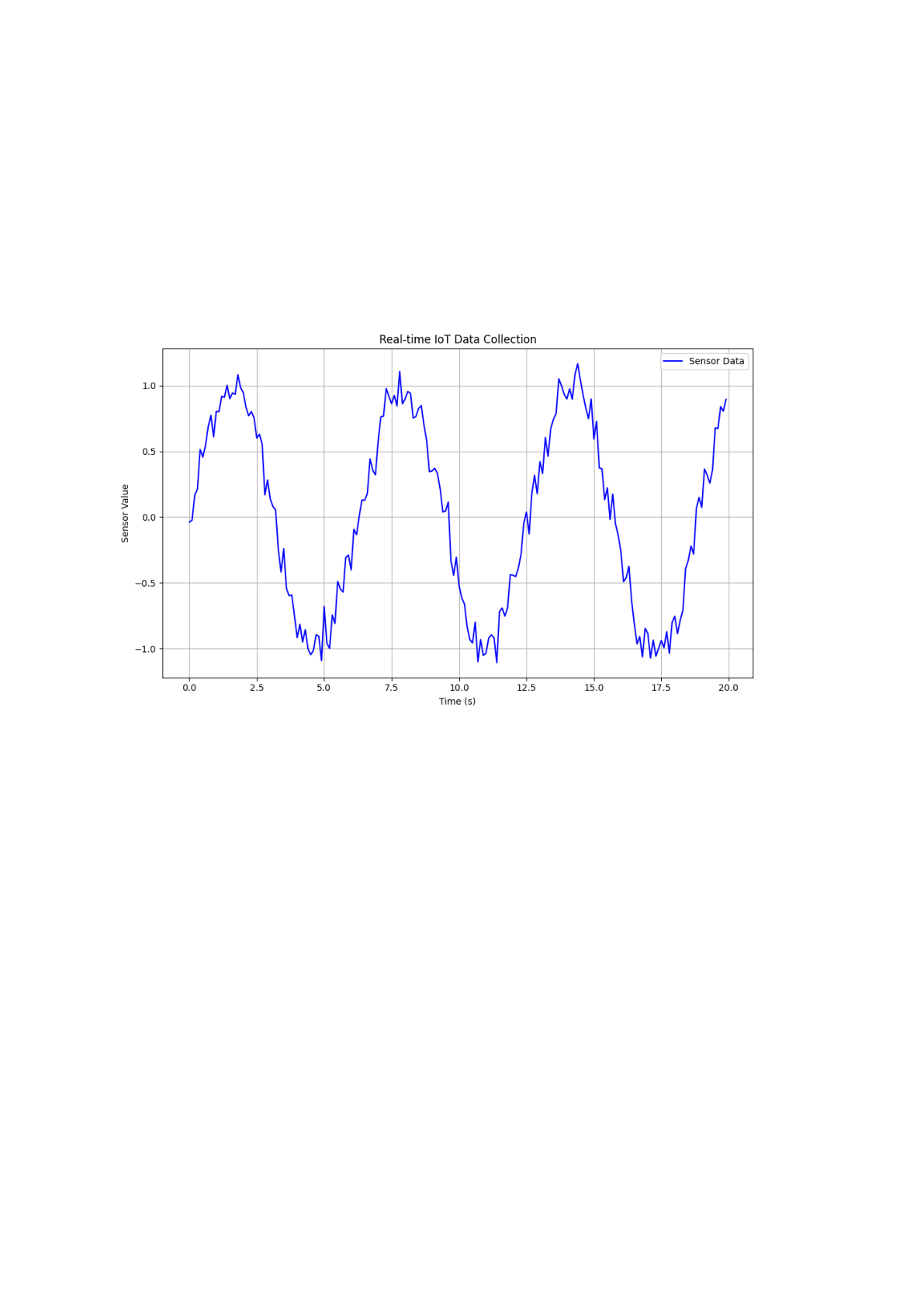
plt.grid(True)   
plt.legend()   
plt.tight\_layout()   
plt.show()

**Real-time IoT data collection:**   
import matplotlib.pyplot as plt   
import numpy as np

# Sample data: time-series of IoT sensor readings (replace with actual IoT data)   
time = np.arange(0, 20, 0.1) # Time in seconds   
sensor\_data = np.sin(time) + 0.1 \* np.random.randn(len(time)) # Simulated sensor readings

# Plotting   
plt.figure(figsize=(10, 6))   
plt.plot(time, sensor\_data, label='Sensor Data', color='blue')

# Labels and formatting   
plt.title('Real-time IoT Data Collection')



plt.xlabel('Time (s)')   
plt.ylabel('Sensor Value')   
plt.grid(True)   
plt.legend()   
plt.tight\_layout()   
plt.show()