

**The Superior University**

📝 Operating Systems Lab – Project Documentation

# 📌 Project Title

*Simulation of First-Come, First-Served (FCFS) Scheduling in Python – Server Request Management System*

# Group Members

* Abdul Rehman (SU92-BSSEM-F23-007)
* Hafiz Shaheer Ahmed (SU92-BSSEM-F23-014)
* Muhammad Bilal Saqib (SU92-BSSEM-F23-022)
* Muhammad Dawood (SU92-BSSEM-F23-002)

# 📂 GitHub Repository

# <https://github.com/MuhammadDawood9/Server_request_management_system>

# <https://github.com/MuhammadBilalSaqib/Server-Management>

# <https://github.com/shaheer1071/Server-Request-Scheduling-using-FCFS-in-Python>

# https://github.com/AbdulR1329/server-request-management-system

**🔧 Scheduling Algorithm Implemented**

* ✅ FCFS (First Come First Serve)
* ⬜ SJF (Shortest Job First – Non-Preemptive)
* ⬜ SJF (Preemptive)
* ⬜ Round Robin

**📄 Project Description**

**Problem Solved:**  
This project simulates how a server processes incoming requests using the **First-Come, First-Served (FCFS)** scheduling algorithm. It helps visualize request handling order and compute key OS-level performance metrics like waiting and turnaround times.

**Inputs Required:**

* Arrival Time
* Processing Time (Burst Time)
* (Request IDs are auto-generated)

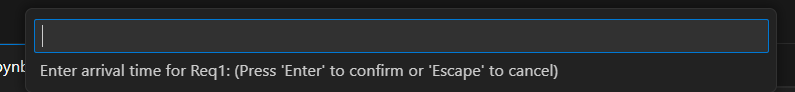
**Outputs Generated:**

* Waiting Time for each request
* Turnaround Time for each request
* Start and Completion Time of each request
* Average Waiting and Turnaround Time
* Server Utilization
* Gantt Chart of request execution

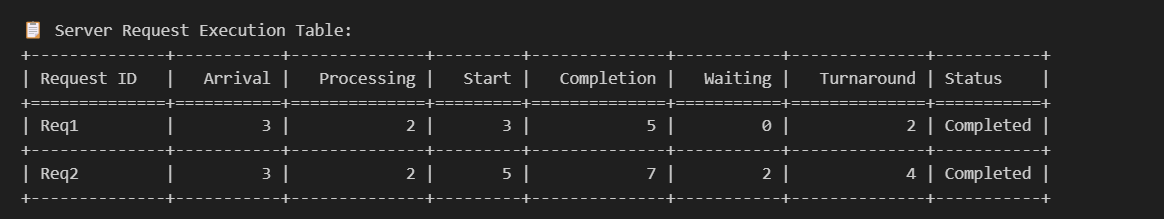
**Implementation Summary:**  
The FCFS algorithm is implemented in Python using lists and basic control structures. Requests are sorted by arrival time, then processed sequentially. Metrics are calculated during execution. **Matplotlib** is used for Gantt chart visualization, and **Tabulate** for tabular output formatting.

**🖼 Output Screenshots**

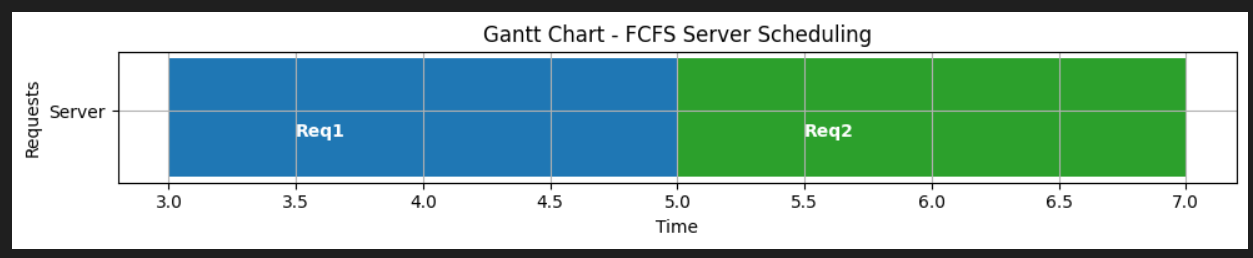
* ✅ Screenshot 1: User Input of Requests



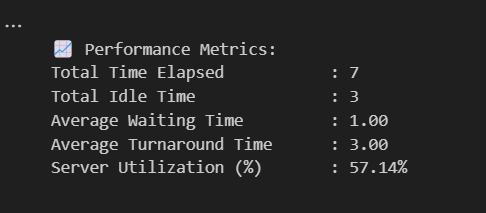
* ✅ Screenshot 2: FCFS Scheduling Execution Table



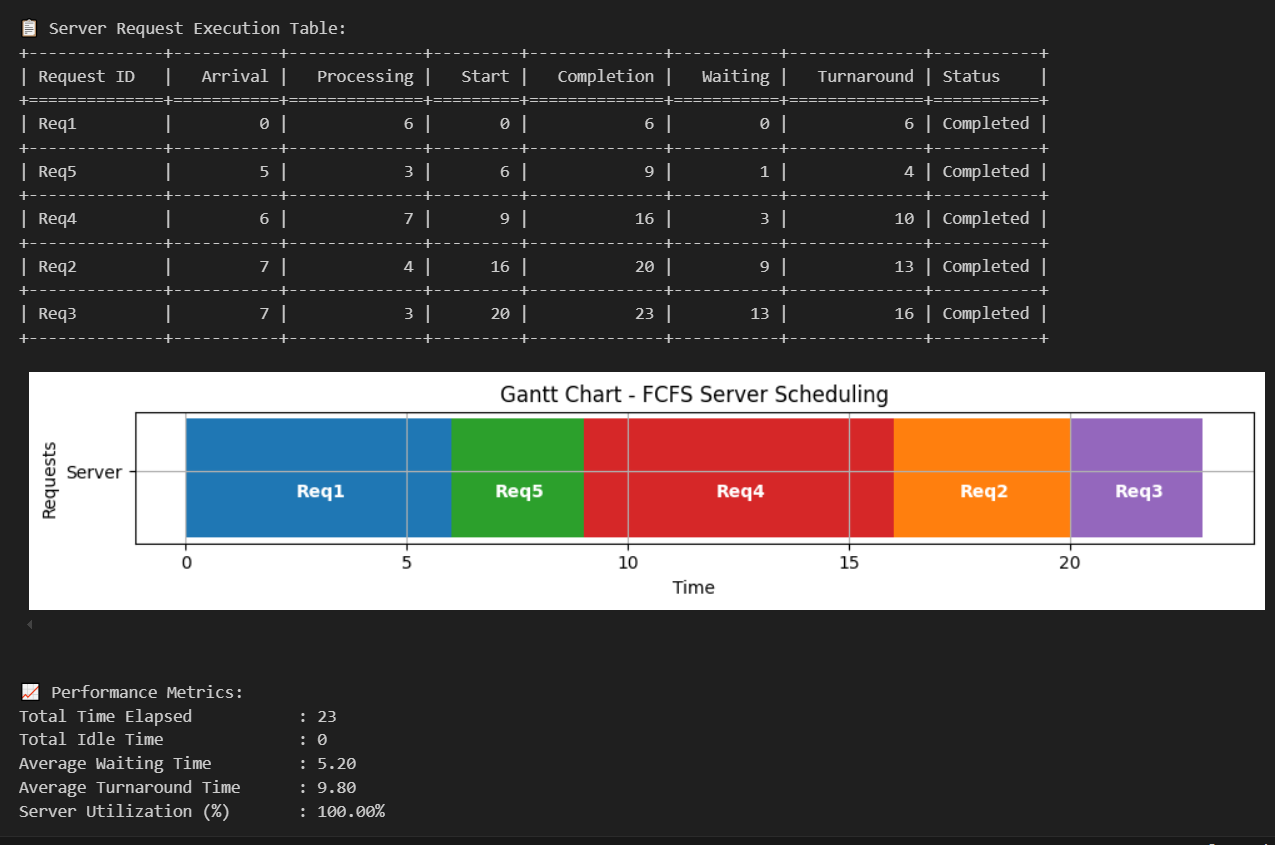
* ✅ Screenshot 3: Gantt Chart Visualization



* ✅ Screenshot 4: Performance Metrics Summary



**Output #02:**

****

**🧾 Code Structure & Explanation**

**Functions Used:**

* get\_requests\_input() – Accepts and stores request arrival and processing times
* fcfs\_schedule() – Performs FCFS scheduling and calculates all timing metrics
* display\_request\_table() – Displays the final scheduling results in a table
* draw\_gantt\_chart() – Visualizes execution using a Gantt chart
* calculate\_metrics() – Computes averages and utilization

**Core Logic:**

* Requests are sorted by **arrival time**
* Server processes requests in order using a **current time tracker**
* If the server is idle, it waits until the next request arrives
* Metrics like **waiting time**, **turnaround time**, and **utilization** are calculated per request

**External Libraries Used:**

* matplotlib – For Gantt chart plotting
* tabulate – For neatly formatted console tables

**📊 Performance Metrics**

| **Metric** | **Value (Example)** |
| --- | --- |
| Average Waiting Time | e.g., 3.33 |
| Average Turnaround Time | e.g., 6.67 |
| Total Idle Time | e.g., 2 |
| Server Utilization (%) | e.g., 83.33% |
| Time Quantum (if RR) | Not Applicable |

**🛠 Challenges Faced**

1. **Accurate Time Tracking**
   * **Challenge:** Handling idle periods where the CPU must wait
   * **Solution:** Used max(current\_time, arrival\_time) to start processing properly
2. **Maintaining Request Identity**
   * **Challenge:** Ensuring request IDs stay aligned after sorting
   * **Solution:** Encapsulated each request as a dictionary with ID included
3. **Correct Metric Calculation**
   * **Challenge:** Validating turnaround and waiting times
   * **Solution:** Verified with manual test cases and step-by-step debugging