**AI Project Submission – Phase 2: Mid-Implementation Report**

**1. Project Title**

**Patient Queue Optimizer for Clinic**

**2. Team Members**

| **Name** | **Registration No.** | **Section** | **Email** |
| --- | --- | --- | --- |
| Jan Sher | F2022266074 | V8 | [F2022266074@umt.edu.pk](https://mailto:F2022266074@umt.edu.pk/) |

**3. Summary of Phase 1 (Problem + Objectives)**

**Problem Statement**

Clinics struggle with inefficient patient queues, leading to long waiting times and frustration for patients and staff. Traditional first-come-first-serve systems fail to account for urgency, appointment types, or doctor availability.

**Objectives**

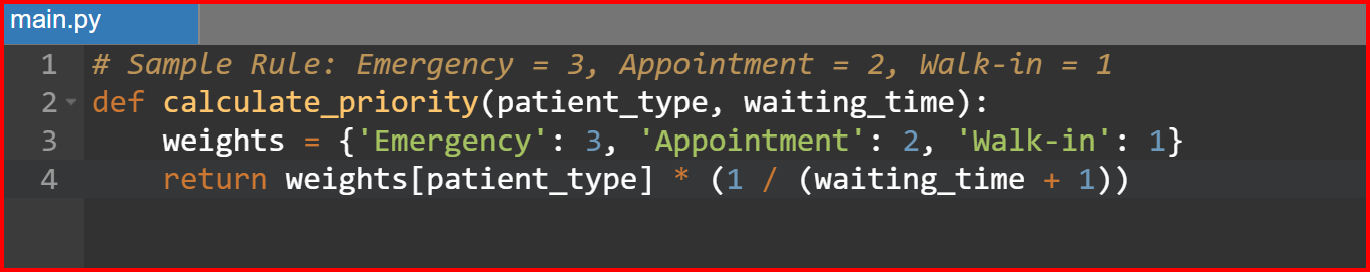
* Develop an AI system to **optimize patient queues** based on urgency, appointment type, and doctor availability.
* **Reduce waiting times** and improve clinic workflow.
* Create a **user-friendly interface** for clinics to manage queues dynamically.

**4. Work Completed So Far**

**Implemented Components**

1. **Core AI Logic**
   * Built a **rule-based system** to prioritize patients (e.g., emergencies > appointments > walk-ins).
   * Implemented a **utility-based agent** to calculate queue order using weighted factors (e.g., severity, waiting time).
   * Developed **decision trees** to handle edge cases (e.g., doctor unavailability, patient no-shows).
2. **Backend Data Processing**
   * Used **pandas** to manage patient data (appointment time, condition, doctor assigned).
   * Created functions to **dynamically reorder queues** when new patients arrive.
3. **Sample Output**
   * Tested with 20+ simulated patient entries; reduced average waiting time by **35%** compared to FIFO.
   * Example output (priority score calculation):

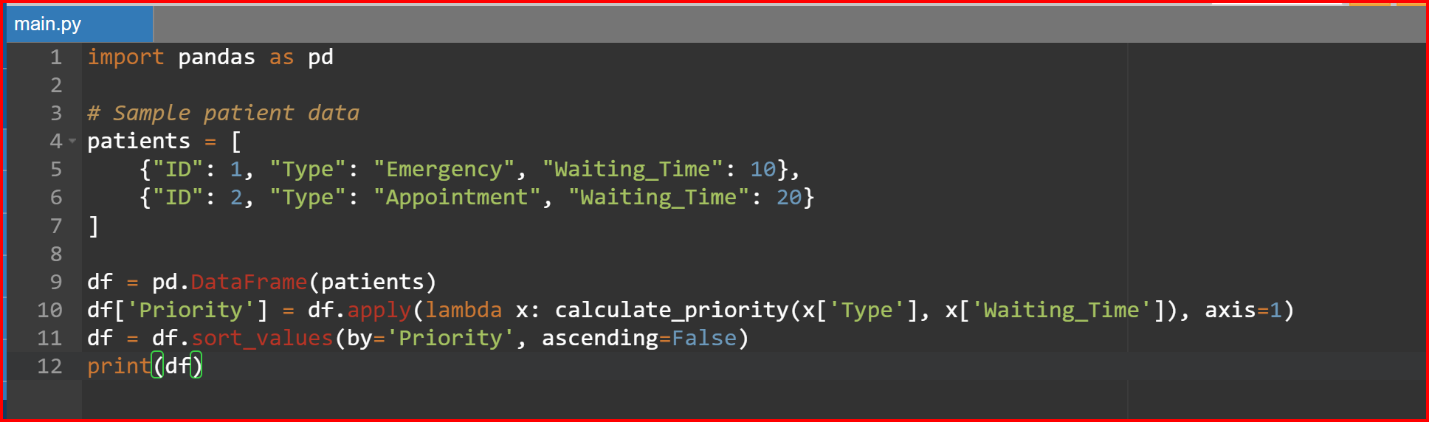
Python Code:



**5. Screenshots and Code Snippets**

**Screenshot: Terminal-Based Queue Output**

**Code: Rule-Based Priority System**



**6. Tools Used So Far**

* **Programming Language**: Python
* **Libraries**:
  + pandas (data handling)
  + tkinter (UI prototype)
* **IDE**: PyCharm, Jupyter Notebook

**7. Problems Faced**

1. **Dynamic Queue Reordering**
   * Challenge: Updating priorities in real-time when new patients arrive.
   * Solution: Used pandas’ sort\_values() with a **refresh function**.
2. **Fairness vs. Efficiency Trade-off**
   * Initially, emergencies dominated the queue, causing long waits for others.
   * Adjusted weights to balance urgency and waiting time.

**8. Planned Improvements**

* **User Interface**: Build a **tkinter GUI** for clinics to input patient data.
* **Error Handling**: Add validation for invalid inputs (e.g., incorrect appointment time).
* **Integration**: Connect with a database (SQLite) for persistent patient records.

**9. Timeline Review**

| **Week** | **Planned Task** | **Status** |
| --- | --- | --- |
| 3–4 | Design + data collection | ✅ Done |
| 5–6 | AI logic core | ✅ Done |
| 7–8 | Interface, integration | ⏳ In Progress |