PROJECT PROPOSAL

Artificial Intelligence

BS(CS)-5B

AIR UNIVERSITY ISLAMABAD

Group Members

Name	Registration-ID
Afia Aziz	231561
Zumer Dhillun	231597
Zoya Azad	231579

Submitted to: Ma'am Hareem Kibriya

Project Proposal: Medi-Match

AI-Driven Hospital Resource Optimization & Clinical Triage System

1. Problem Statement

Modern hospitals face a "Resource Allocation Paradox": while they have the staff and beds, the manual process of matching patient needs to doctor specialties often leads to bottlenecks, improper referrals, and uneven doctor workloads. Furthermore, the high-pressure environment of an Emergency Room requires an objective, AI-driven method to calculate patient priority and immediate specialist needs without human bias.

2. Objectives

- **Multi-Algorithmic Scheduling:** Deploy both Heuristic and Evolutionary (Genetic) algorithms to automate patient-doctor assignments.
- **Clinical Triage Assessment:** Implement a rule-based AI to calculate urgency scores and clinical actions.
- **Intelligent Specialist Mapping:** Automatically identify and recommend the correct specialist (e.g., Pulmonologist, Trauma Surgeon) based on emergency symptoms.
- **System Convergence Visualization:** Provide a visual evidence-base for the AI's performance through real-time quality graphs.

3. Technical Implementation & Algorithms

The Medi-Match engine utilizes four distinct algorithmic layers to ensure hospital efficiency:

3.1. Fuzzy Logic Inference (Urgency Fuzzification)

The system employs a linear membership function to transform discrete urgency inputs (integers 1–10) into a continuous fuzzy set [0,1]. This **Fuzzy Score** allows the scheduling engine to apply weighted priorities, ensuring that a "Level 9" patient receives mathematically higher priority in the doctor-scoring algorithm than a "Level 5" patient.

3.2. Rule-Based Expert System (Specialty Mapping)

The core logic utilizes a predefined **Clinical Knowledge Base**. This system maps specific diseases and emergency symptoms to corresponding medical departments.

- **Technique:** Literal and Keyword matching against a specialty-condition matrix.
- **Impact:** Ensures 100% accuracy in "Perfect Match" assignments and identifies when a "Referral" is necessary.

3.3. Greedy Heuristic Optimization (Heuristic Scheduler)

For rapid, real-time scheduling, the system uses a **Weighted-Scoring Greedy Heuristic**. It evaluates every possible Doctor-Patient pair based on:

- Specialty Compatibility (Base Weight: 40)
- Dynamic Load Balancing (Weight: up to 30)
- Seniority/Urgency scaling (Weight: up to 30)
  The algorithm makes the locally optimal choice for each patient to ensure a fast and reliable baseline schedule.

3.4. Genetic Algorithm (Evolutionary Optimization)

For complex scenarios, the **Genetic Algorithm (GA)** module is employed to find a global optimum.

- **Process:** It creates a "Population" of random schedules and uses **Tournament Selection**, **Two-Point Crossover**, and **Random Mutation** over 120+ generations.
- **Goal:** To maximize the "Fitness Function," which balances perfect matches against doctor utilization rates.

3.5. Round-Robin Resource Allocation

A cyclic **Round-Robin algorithm** is utilized for bed management, ensuring that hospital physical resources are distributed evenly across the facility to prevent department-specific overcrowding.

4. System Outcomes

- **Optimal Doctor Utilization:** The system calculates a utilization metric (patients/doctor) to prevent physician burnout.
- **Actionable Triage Reports:** Automated generation of HTML/Text reports including **AI-Score**, **Clinical Action**, and **Recommended Specialist**.
- **Visual Performance Tracking:** A dynamic "Convergence Graph" that plots the AI's learning curve, proving the mathematical validity of the generated schedule.

5. Conclusion

Medi-Match represents a synthesis of classic Expert Systems and modern Evolutionary Computing. By moving away from manual "pen-and-paper" scheduling toward an AI-integrated approach, healthcare providers can ensure that every patient is seen by the right specialist at the right time, maximize the use of hospital assets, and ultimately save lives through faster triage response.

Technical Stack:

- **Frontend:** C# / WPF (XAML)
- **AI Engine:** Python 3.x
- **Data Format:** JSON / CSV
- **Libraries:** Matplotlib (Visualization), NumPy (Data Processing)