AI-Based Employee Shift Scheduler

Objective:

Build an AI-driven scheduling system that assigns employees to available shifts based on their **availability**, **preferred working hours**, and **daily/weekly hour limits**. The goal is to maximize employee satisfaction and shift coverage while respecting all constraints.

🧩 Problem Description:

You are managing a workforce of employees who each provide:

- **Available working hours per day** (e.g., 9 AM to 5 PM)
- **Preferred shift times** (e.g., prefers morning shifts)
- **Max hours per week/day**
- **Skill levels** (optional—can be assumed to match all shifts for this assignment)

You are given:

- A list of **open shifts** (e.g., 3 shifts per day: Morning, Afternoon, Night)
- Each shift requires **a fixed number of employees**

Your task is to **generate an optimal weekly shift schedule** that:

- Covers all required shifts
- Respects employee constraints
- Maximizes employee preferences

📘 Requirements:

Inputs:

- JSON or CSV files containing:

`employees.json`:
json

```
[
  {
    "id": "E001",
    "name": "Alice",
    "availability": {
```

```
      "Mon": ["09:00", "17:00"],
      "Tue": ["09:00", "17:00"],
      ...
    },
    "preferred_shifts": ["Morning"],
    "max_daily_hours": 8,
    "max_weekly_hours": 40
  }
]
```

-

shifts.json:
json

```
[
  {
    "day": "Mon",
    "shift_type": "Morning",
    "start": "09:00",
    "end": "13:00",
    "required_employees": 2
  }
]
```

-

Constraints to Handle:

- Employee must be **available** during a shift
- Cannot exceed **daily or weekly hour limits**
- Try to assign shifts based on **preferred shift times**
- Each shift must be **fully staffed** (if possible)

🛠️ What You'll Build:

- A Python program or Jupyter notebook that:
  - Parses input data
  - Uses a constraint-based algorithm (e.g., greedy, backtracking, or linear programming) to assign employees to shifts
  - Outputs a **schedule table** (CSV or printed matrix format)

📊 Bonus (Optional):

- Score each schedule based on **employee satisfaction** (how closely it aligns with their preferences)
- Implement a **simple Gantt chart** or **heatmap** visualization of the weekly schedule

---

✅ Deliverables:

1. **Code** (Python script or notebook)
2. **Sample input files**
3. **Output schedule** (CSV or table)
4. **Brief README** explaining:
    - Your algorithm
    - Assumptions
    - How to run the code

🕒 Timeline:

- Submit within **7 days**