RAJALAKSHMI INSTITUTE OF TECHNOLOGY
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)
ACADEMIC YEAR 2025 – 2026
SEMESTER III

ARTIFICIAL INTELLIGENCE LABORATORY

MINI PROJECT REPORT

REGISTER NUMBER	2117240030123
NAME	SAI ADITI . P
PROJECT TITLE	EXAM SEAT ARRANGEMENT SYSTEM
FACULTY IN-CHARGE	Mrs.M.BHAVANI

Signature of Faculty In-charge: _____

INTRODUCTION

The **Exam Seat Arrangement System** is an AI-based application that automates the process of assigning seats to students during examinations. It eliminates manual effort and reduces errors that often occur when arranging large numbers of students across multiple rooms. The system takes input such as student details, register numbers, subjects, and room capacities to generate an organized seating plan.

In educational institutions, manually preparing seating charts for exams is time-consuming and prone to human mistakes. By applying simple algorithms and rule-based logic, the system ensures fair, efficient, and conflict-free seating arrangements. This helps staff save time and maintain a smooth examination process.

The system can be modeled as a **Constraint Satisfaction Problem (CSP)** where students, rooms, and seats act as variables with specific constraints. It applies simple **AI techniques** like **backtracking** or **rule-based allocation** to assign seats fairly.
This ensures **no seat duplication** and maintains **subject-wise separation**. The system thus provides an **efficient and automated** exam seating solution.

PROBLEM STATEMENT

In most educational institutions, preparing **seating arrangements** for exams manually takes a lot of time and effort. Mistakes such as assigning **two students to the same seat**, misplacing **register numbers**, or **seating students from the same subject together** are common.
Hence, there is a need for an **intelligent system** that can automatically assign seats to students based on constraints such as **room capacity**, **student count**, and **subject type**, ensuring proper distribution and fairness.

GOAL

The goal of this project is to develop an Exam Seat Arrangement System that can assign seats to students efficiently by processing their details and room information. The system aims to:

- Reduce manual workload for staff members
- Avoid duplication or subject-based conflicts
- Generate seating plans quickly and accurately

Ultimately, this project shows how automation and AI can make administrative tasks in education faster, easier, and more reliable.

THEORETICAL BACKGROUND

Automation in Education:

Automation has become an essential part of modern educational institutions, improving efficiency in administrative processes like exam management, attendance tracking, and timetable generation. In traditional examination setups, staff members manually prepare seating charts, which can be time-consuming and error-prone. Automation allows this process to be handled by a computer-based system that can quickly analyze data such as the number of students, room capacities, and subject details to generate a clear and organized seating plan. This not only reduces human effort but also minimizes mistakes and ensures fairness in seat allocation.

AI in Administrative Systems:

Artificial Intelligence (AI) is widely used in automation and scheduling tasks that involve constraints and resource optimization. AI-based approaches can make decisions by following specific rules or by solving Constraint Satisfaction Problems (CSPs), where a solution must satisfy a set of predefined conditions. In an exam context, the system ensures that no two students with the same subject sit close to each other and that each room's capacity is properly utilized. Through intelligent allocation, AI helps create efficient, unbiased, and structured seating plans.

Constraint Satisfaction Problems (CSPs):

A CSP consists of a set of variables, domains, and constraints. In this project:

- Variables are the students, rooms, and seat numbers.
- Domains represent possible seat positions or rooms for each student.
- Constraints ensure that students from the same subject are not seated together and that no room exceeds its capacity.

Solving the CSP involves assigning values (seats) to each variable (student) so that all constraints are satisfied. Techniques like backtracking, heuristic search, or rule-based inference can be used to find a valid seating arrangement efficiently

AI Techniques and Rule-Based Logic:

This system uses rule-based logic, where each rule defines how a student should be placed based on specific conditions like room capacity, subject distribution, and roll number order. Rule-based methods are simple yet effective for problems with fixed conditions. In more advanced implementations, machine learning or optimization algorithms can also be used to improve performance when handling large datasets.

Justification:

The use of AI and automation in exam management is justified by the need for accuracy, time efficiency, and reduced manual workload. Human errors such as duplicate seat assignments, incorrect numbering, or biased placement can be avoided through an intelligent system. Additionally, automated systems make it easier to regenerate seating plans quickly in case of last-minute changes, ensuring flexibility and reliability.

ALGORITHM EXPLANATION WITH EXAMPLE

Algorithm Overview:

The Exam Seat Arrangement System follows a five-step process:

Step 1 — Input Data:
Collect student details such as Register Number, Name, and Subject, along with Room Details like room number and capacity.

Step 2 — Sorting and Grouping:
Group or sort students based on subject or department to ensure that students from the same subject are not placed together.

Step 3 — Seat Allocation Rule:
Assign seats one by one, alternating between students from different subjects or departments.

Step 4 — Room Filling:
When a room reaches its full capacity, move automatically to the next available room.

Step 5 — Output Generation:
Display or generate a seating plan in tabular format showing Room Number, Seat Number, Student Name, Register Number, and Subject.

Complete Example Workflow:

1. **Input Collection:**

   The administrator uploads or enters the list of students with details such as **Register Number**, **Name**, and **Subject**, along with **Room Details** like **Room Number** and **Capacity**.

2. **Data Processing:**

   The system processes the student list and sorts them by **subject** or **department**. It checks available rooms and assigns each student to a seat, ensuring that no two students from the same subject sit next to each other.

3. **Seat Allocation:**

   The algorithm applies simple **CSP-based logic** or **rule-based allocation** to distribute students fairly across all rooms. If one room reaches its maximum capacity, the system automatically moves to the next room.

4. **Arrangement Generation:**

   The system generates a **final seating plan** that includes details such as **Room Number**, **Seat Number**, **Student Name**, **Register Number**, and **Subject**.

5. **Display / Output:**

   The seating arrangement is displayed on the screen or exported as a **printable seating chart (PDF or Excel)** for staff and invigilators to use during exams.

IMPLEMENTATION AND CODE

The following Python code implements the AI-based medical diagnosis system:

```
# Step 2: Constraints
# 1. Each seat can be assigned to only one student.
# 2. Students of the same subject should not be adjacent (simple version).
# 3. Room capacity cannot be exceeded.
# Step 3: CSP Solver (simple backtracking)
arrangement = []
used_seats = set()
for i, student in enumerate(students):
    for room, seat in domains:
        if (room, seat) not in used_seats:  # constraint 1
            # simple adjacency constraint check
            if arrangement and arrangement[-1]["Subject"] == student["subject"]:
                continue  # skip same subject adjacency
            arrangement.append({
                "Room No": room,
                "Seat No": seat,
                "Name": student["name"],
                "Reg No": student["regno"],
                "Subject": student["subject"]
            })
            used_seats.add((room, seat))
            break
```

```
# Exam Seat Arrangement System using Constraint Satisfaction (CSP) Concept
# Step 1: Define variables (students), domains (available seats), and constraints
students = [
    {"name": "Anjali R", "regno": "21IT001", "subject": "AI"},
    {"name": "Ravi K", "regno": "21CS002", "subject": "DBMS"},
    {"name": "Sneha M", "regno": "21IT003", "subject": "AI"},
    {"name": "Karthik P", "regno": "21CS004", "subject": "DBMS"},
    {"name": "Manoj S", "regno": "21IT005", "subject": "AI"},
    {"name": "Deepa K", "regno": "21CS006", "subject": "DBMS"}
]
rooms = {
    "101": 3,
    "102": 3
}
# Variables → Students
# Domains → All possible (Room, Seat) pairs
domains = []
for room, cap in rooms.items():
    for seat in range(1, cap + 1):
        domains.append((room, seat))
# Step 4: Display final arrangement
print("\n--- Exam Seating Arrangement (CSP-Based) ---\n")
print("Room No | Seat No | Name        | Register No | Subject")
print("-----------------------------------------------------------")
for a in arrangement:
    print(f"{a['Room No']:>6} | {a['Seat No']:>7} | {a['Name']:<10} | {a['Reg No']
        :<11} | {a['Subject']}")
```

CODE EXPLANATION:

Step 1: Student details and room capacities are entered as dictionaries.
Step 2: The program initializes variables to track current room and seat numbers.
Step 3: The program assigns each student a seat using a simple **CSP-based rule** — one by one, checking room capacity and moving to the next room when needed.
Step 4: The final **seating plan** is displayed in a neat tabular format.

OUTPUT:

```
--- Exam Seating Arrangement (CSP-Based) ---

Room No | Seat No | Name         | Register No | Subject
-----------------------------------------------------------
   101 |       1 | Anjali R   | 21IT001     | AI
   101 |       2 | Ravi K     | 21CS002     | DBMS
   101 |       3 | Sneha M    | 21IT003     | AI
   102 |       1 | Karthik P  | 21CS004     | DBMS
   102 |       2 | Manoj S    | 21IT005     | AI
   102 |       3 | Deepa K    | 21CS006     | DBMS
```

OUTPUT EXPLANATION:

When the program is executed, it displays the final exam seating arrangement in a clear tabular format that includes Room Number, Seat Number, Student Name, Register Number, and Subject.
Each student is assigned a unique seat based on the defined CSP constraints, ensuring that no two students share the same seat and that students from the same subject are not seated next to eachother.

The output demonstrates how the system applies constraint satisfaction and rule-based logic to automatically generate a valid and fair seating plan.
Each room is filled up to its defined capacity before moving on to the next room, ensuring efficient space utilization.

RESULTS AND FUTURE ENHANCEMENT

Results:

The developed Exam Seat Arrangement System successfully simulates the process of AI-driven exam management. It automatically allocates seats to students based on predefined constraints like room capacity, subject separation, and seat uniqueness. The system ensures that every seat is assigned efficiently, avoiding duplication and maintaining fair distribution across rooms. This demonstrates how CSP (Constraint Satisfaction Problem) and AI-based logic can be applied in educational administration to reduce manual effort, save time, and improve accuracy.

Comparison with Manual Methods:

Traditional exam seating methods rely heavily on manual planning by staff members, which is time-consuming and prone to human error. Rule-based or automated systems, on the other hand, ensure consistency, accuracy, and speed in generating seating charts. While manual methods often require repeated revisions, the proposed system can quickly generate or update seating arrangements with minimal human involvement.

Future Enhancements:

- Integration with Databases: The system can be extended to fetch student and room data directly from a database for real-time updates.
- Graphical User Interface (GUI): A user-friendly interface can be developed for easy input, visualization, and editing of seating plans.
- PDF/Excel Report Generation: The final seating plan can be exported as a printable report for invigilators and staff.
- Dynamic Reallocation: The system can be improved to automatically reassign seats in case of absentees or last-minute changes.
- Optimization Techniques: Advanced AI algorithms like Genetic Algorithms or Heuristic Search can be used to handle large-scale seating arrangements more efficiently.

Git Hub Link of the project and report	https://github.com/Saiaditip/AI_MINIPROJECT

REFERENCES

1. Rajpurkar, P., et al. (2017). *CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning.* Stanford ML Group.
2. Esteva, A., et al. (2017). *Dermatologist-level classification of skin cancer with deep neural networks. Nature*, 542(7639), 115–118.
3. Litjens, G., et al. (2017). *A Survey on Deep Learning in Medical Image Analysis. Medical Image Analysis*, 42, 60–88.
4. IBM Cloud Education. (2023). *What is Artificial Intelligence (AI)?* Available at: https://www.ibm.com/topics/artificial-intelligence
5. • Kaggle Dataset: *Chest X-Ray Images (Pneumonia).* Available at: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia