

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

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240070121 |
| **NAME** | A.B.JAHNAVI |
| **PROJECT TITLE** | EXAM HALL SEATING ARRANGEMENT USING BACKTRACKING |
| **DATE OF SUBMISSION** | 28.10.2025 |
| **FACULTY IN-CHARGE** | **Mrs. M. Rubina begam** |

**Signature of Faculty In-charge**

**INTRODUCTION**

Artificial Intelligence (AI) is a field of computer science that focuses on creating systems capable of performing tasks that typically require human intelligence. These include problem-solving, learning, reasoning, planning, and perception. AI techniques such as search algorithms, constraint satisfaction, and reasoning mechanisms are widely used to solve complex real-world problems efficiently.

Arranging students in an exam hall manually can be challenging, especially when there are multiple departments and the seating must be organized to prevent malpractice. This project applies an **AI-based Constraint Satisfaction Problem (CSP)** approach to automate the **Exam Hall Seating Arrangement** process. By using the **Backtracking algorithm**, the system ensures that no two adjacent students belong to the same department.

**PROBLEM STATEMENT**

To develop a system that automatically arranges students in an exam hall such that **no two students from the same department are seated adjacent (horizontally or vertically)**, using the **Backtracking algorithm** as the constraint-solving technique.

In many academic institutions, students from different departments write their examinations in shared halls. Random allocation may lead to clustering of students from the same department, increasing the risk of unfair practices. Hence, a logical, automated arrangement system is essential.

This project aims to design a seating plan generator that assigns students to seats under given constraints, ensuring fairness and efficiency in seating arrangements.

**GOAL**

1. To generate a valid and optimized seating plan for an exam hall.
2. To ensure that adjacent seats are occupied by students from different departments.
3. To demonstrate how **Constraint Satisfaction** and **Backtracking** can be applied in real-world administrative problems.

**THEORETICAL BACKGROUND**

### **Backtracking Algorithm**

Backtracking is a **depth-first search technique** used to find solutions by incrementally building candidates and abandoning a candidate (“backtracking”) as soon as it fails to satisfy the constraints.  
It is widely used in problems like N-Queens, Sudoku solving, and scheduling.

### **Justification for Choosing Backtracking**

Backtracking is chosen because:

It is simple to implement for grid-based problems.

It guarantees a valid solution if one exists.

It efficiently handles constraint satisfaction by exploring all possible configurations with pruning.

**ALGORITHM EXPLANATION WITH EXAMPLE**

### **Algorithm Steps**

Represent the exam hall as a 2D grid.

Assign students to seats one by one.

Before placing a student, check:

Is there a student from the same department to the left or above?

If no conflict → place the student.

If conflict → backtrack to the previous seat and try another student.

Continue until all students are seated successfully.

### **Example**

**Departments:** CSE, ECE, IT, EEE  
**Seats:** 2 rows × 4 columns

If a CSE student is seated at (0,0), the next adjacent position (0,1) cannot have another CSE student.  
Thus, the arrangement is made step by step ensuring constraints are met.

**IMPLEMENTATION AND CODE**

### **Language:** Python

### **IDE Used:** VS Code

# Exam Hall Seating Arrangement using Backtracking

students = [

("S1", "CSE"), ("S2", "CSE"), ("S3", "ECE"), ("S4", "ECE"),

("S5", "IT"), ("S6", "IT"), ("S7", "EEE"), ("S8", "EEE")

]

rows, cols = 2, 4

hall = [["" for \_ in range(cols)] for \_ in range(rows)]

def is\_safe(r, c, dept):

# Check left seat

if c > 0 and hall[r][c-1].endswith(dept):

return False

# Check upper seat

if r > 0 and hall[r-1][c].endswith(dept):

return False

return True

def arrange\_seats(index):

if index == len(students):

return True

name, dept = students[index]

for r in range(rows):

for c in range(cols):

if hall[r][c] == "" and is\_safe(r, c, dept):

hall[r][c] = f"{name}-{dept}"

if arrange\_seats(index + 1):

return True

hall[r][c] = "" # Backtrack

return False

if arrange\_seats(0):

print("✅ Exam Hall Seating Arrangement:\n")

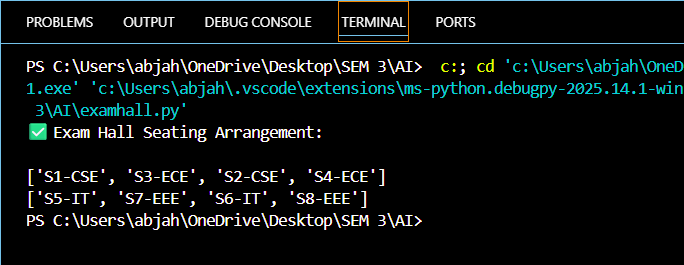
for row in hall:

print(row)

else:

print("❌ No valid arrangement found.")

**OUTPUT**



**RESULTS AND FUTURE ENHANCEMENT**

### **Results**

The system successfully generates a valid seating arrangement that satisfies the adjacency constraint. The backtracking approach ensures correctness and flexibility for different hall sizes and departments.

### **Future Enhancements**

* Add gender or roll number constraints.
* Fetch student data directly from a database.
* Generate printable seating charts or hall layouts.
* Develop a GUI for visualization using Tkinter or Flask.

|  |  |
| --- | --- |
| **Git Hub Link of the**  **project and report** | **[https://github.com/JAHNAVIBOOPATHY/Exam-Hall-Seating-Backtracking](https://github.com/YourUsername/Exam-Hall-Seating-Backtracking" \t "_new)** |

**REFERENCES**

* Stuart Russell & Peter Norvig, Artificial Intelligence: A Modern Approach, Pearson, 4th Edition, 2021.
* Ian Millington, Artificial Intelligence for Games, CRC Press, 2nd Edition, 2009.
* TutorialsPoint, Backtracking Algorithms – Concepts and Examples, www.tutorialspoint.com
* GeeksforGeeks, Constraint Satisfaction and Backtracking in Python, www.geeksforgeeks.org
* Coursera – AI Search and Problem Solving Techniques, University of London, 2023.