Generative AI Project

AI-Powered Puzzle Generator (Sudoku & Crossword)

Piyush Jain (Enrollment No. : 0827CY221041)
Trainer : Mr. Aman Dharmendra Sir
1 September 2025

Abstract

The project AI-Powered Puzzle Generator demonstrates the application of Artificial Intelligence concepts in developing brain-challenging games such as Sudoku and Crossword puzzles. Unlike static puzzle books or fixed online boards, this system dynamically generates new Sudoku puzzles with varying difficulty levels (Easy, Medium, Hard) and creates crossword grids from user-supplied or sample word lists.

The Sudoku module employs a backtracking algorithm to generate a fully solved 9×9 grid and selectively removes numbers to form puzzles while retaining solvability. The Crossword module uses a greedy placement approach to arrange words in an N×N grid, allowing basic intersections and visual representation. Both modules are implemented using HTML, CSS, and JavaScript, making the project fully browser-based, lightweight, and platform-independent.

The project serves two purposes: (1) entertainment, by offering endless unique puzzles, and (2) education, by showcasing how AI/ML-inspired problem-solving techniques can be applied to everyday games. This highlights how generative approaches can create engaging, replayable experiences while demonstrating the integration of algorithms with web technologies.

Project Details

A. Objective:

The primary objective of this project is to build an AI-inspired system capable of automatically generating Sudoku and Crossword puzzles of varying difficulty. Conventional puzzle apps often rely on pre-stored sets, which reduces replayability. Our project solves this by dynamically generating new boards each time.

Goals achieved:

1. Generate Sudoku puzzles with Easy, Medium, Hard modes.
2. Implement a solution validator and auto-solver for Sudoku.
3. Provide a crossword generator that arranges words in a grid.
4. Deliver a simple, interactive, browser-based platform requiring no installations.

B. Methodology:

Tools and Technologies Used

- **Frontend Development:**
    - HTML – to design the structure of the interface.
    - CSS – to style the puzzle boards and maintain a responsive design.
    - JavaScript – to implement puzzle generation logic, interactivity, and validation.
- **Algorithmic Approach:**
    - Backtracking for Sudoku generation and solving.
    - Greedy placement with letter matching for Crossword generation.
- **Development Environment:** Visual Studio Code.
- **Testing:** Google Chrome, Mozilla Firefox, and Microsoft Edge.

Sudoku Module

- Step 1: Generate a solved Sudoku grid using recursive backtracking.
- Step 2: Remove numbers based on difficulty while keeping the puzzle solvable.
- Step 3: Provide interactive options (check correctness, auto-solve).
- Step 4: Display results with responsive UI.

Crossword Module

- Step 1: Accept word list (sample or user-input).
- Step 2: Place the longest word centrally.
- Step 3: Attempt to intersect new words with already placed words.
- Step 4: Fill grid with blocks (.) for unused cells.
- Step 5: Provide export as CSV for printing/sharing.

Development Cycle

- Followed an iterative model: design → implement → test → refine.
- Continuous debugging was carried out in the browser console.
- Multiple test cases (different difficulty levels and word sets) were validated.

C.Implementation:

Sudoku

- A 9×9 matrix was generated using backtracking. The algorithm ensures no repetition of numbers in rows, columns, or 3×3 sub-grids.
- Removal of numbers was done randomly but controlled by difficulty levels (Easy: ~36 removals, Medium: ~45, Hard: ~54).
- User inputs are validated against the solution matrix.
- The interface provides Generate Puzzle, Check Solution, and Show Solution buttons.

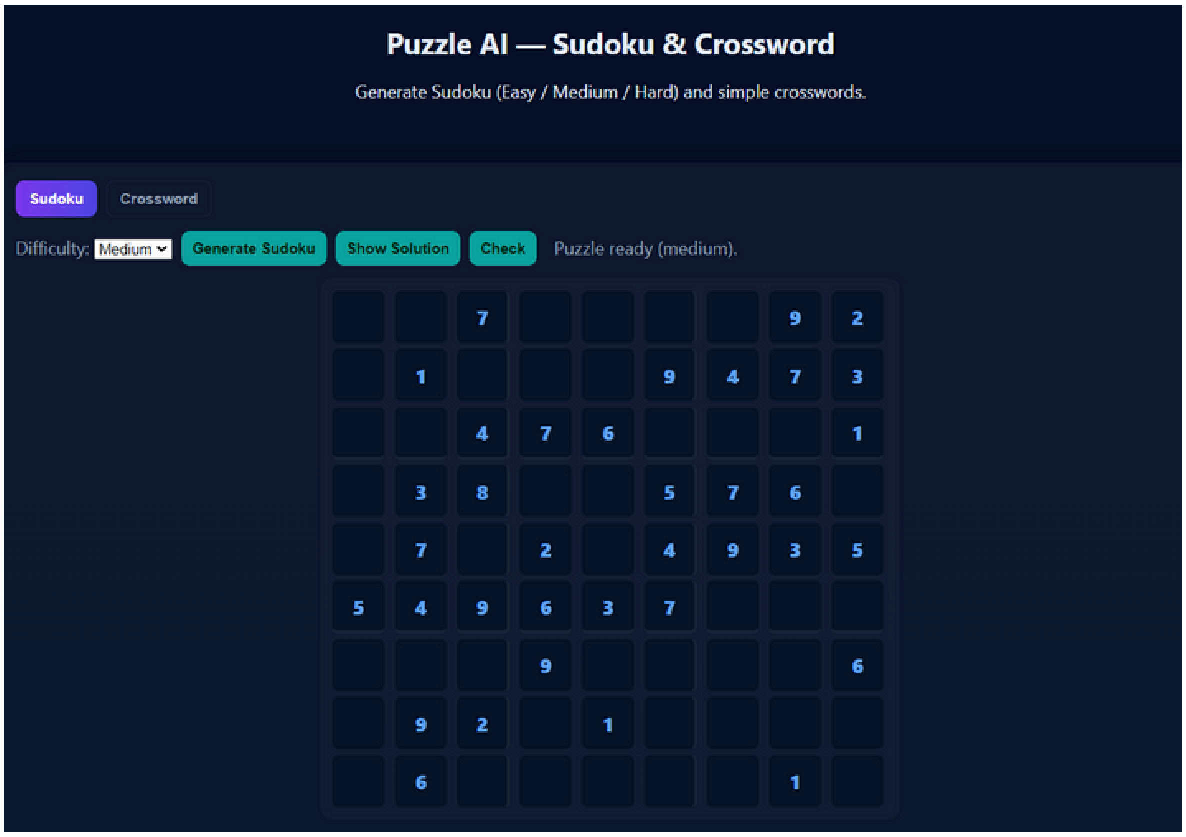
Crossword

- Input is taken via a text area (one word per line).
- Words are placed starting from the center, ensuring intersections at common letters where possible.
- Grid size (11×11, 13×13, 15×15) is user-selectable.
- Output is displayed visually and can be exported as CSV.

Frontend UI

- Tabs separate Sudoku and Crossword.
- Responsive design ensures usability on both desktop and mobile screens.
- Styling uses gradients, borders for Sudoku 3×3 boxes, and block highlighting for Crossword grids.

Screenshots (for report)



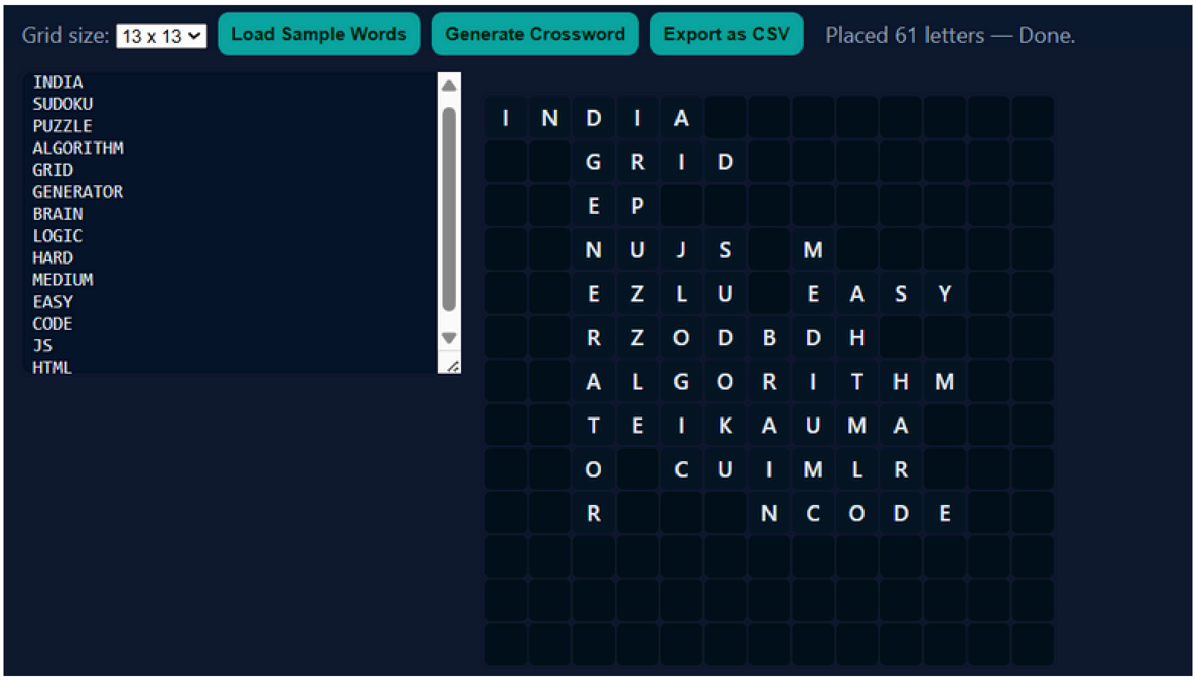
Puzzle AI — Sudoku & Crossword

Generate Sudoku (Easy / Medium / Hard) and simple crosswords.

Difficulty: Medium

Puzzle ready (medium).

		7					9	2
	1				9	4	7	3
		4	7	6				1
	3	8			5	7	6	
	7		2		4	9	3	5
5	4	9	6	3	7			
			9					6
	9	2		1				
	6						1	



Grid size: 13 x 13

Placed 61 letters — Done.

```
INDIA
SUDOKU
PUZZLE
ALGORITHM
GRID
GENERATOR
BRAIN
LOGIC
HARD
MEDIUM
EASY
CODE
JS
HTML
```

I	N	D	I	A								
		G	R	I	D							
		E	P									
		N	U	J	S		M					
		E	Z	L	U		E	A	S	Y		
		R	Z	O	D	B	D	H				
		A	L	G	O	R	I	T	H	M		
		T	E	I	K	A	U	M	A			
		O		C	U	I	M	L	R			
		R				N	C	O	D	E		

D.Results:

1. **Functional Sudoku Generator**
   - Generated playable puzzles in three difficulty levels.
   - Validation system correctly checked user entries.
   - Auto-solver displayed the correct solution.
2. **Crossword Generator**
   - Successfully placed words in different grid sizes.
   - Handled sample dataset and user input efficiently.
   - Export feature worked as expected.
3. **Browser Compatibility**
   - Tested on Chrome, Firefox, and Edge with consistent performance.
4. **Usability**
   - Simple interface, responsive layout, and interactive features.
   - Fully offline; works by directly opening index.html.
5. **Educational & Entertainment Value**
   - Demonstrated how algorithms can power puzzle generation.
   - Showed how simple web tech + AI concepts create engaging applications.

Conclusion

Key Learnings

1. Algorithmic Thinking – Learned how backtracking can generate valid Sudoku puzzles.
2. AI Inspiration – Understood how generative approaches extend beyond text and images into puzzles.
3. Frontend Integration – Combined HTML, CSS, and JS to create interactive, user-friendly apps.
4. Problem-Solving Skills – Overcame debugging challenges, improved logical reasoning, and implemented validation checks.
5. Educational Applications – Observed potential of puzzles for enhancing critical thinking and learning outcomes.

Future Improvements

1. Enforce uniqueness in Sudoku puzzles.
2. Enhance crossword placement algorithm with optimization heuristics.
3. Add crossword clues for true playability.
4. Implement timer, leaderboard, and difficulty analytics.
5. Convert project into a mobile/web app for real-world deployment.

References

[1]. Knuth, D. E. (2000). Dancing Links: Algorithm X for Exact Cover Problems. Stanford University.

[2]. Takayuki, Y. (2002). Computer Algorithm for Sudoku Puzzle Generation. ACM Computing Research Repository.

[3]. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson Education.

[4]. Watanabe, D. (2012). Crossword Puzzle Generation Using Word Matching Heuristics. Journal of Puzzle Studies.

[5]. Ginsberg, M. L. (1993). Dynamic Backtracking. Journal of Artificial Intelligence Research.

[6]. Cruz, J., & Brown, K. (2015). Procedural Content Generation in Games: A Survey. ACM Computing Surveys.

[7]. Mantere, T., & Koljonen, J. (2007). Solving and Generating Sudoku Puzzles with Genetic Algorithms. IEEE Symposium on Computational Intelligence.

[8]. Mitkov, R., & Ha, L. (2003). Computer-Aided Generation of Crosswords. Proceedings of the EACL Workshop on Natural Language Processing and Games.

Code

GitHub link:

https://github.com/piyushjain1310/AI-Powered-Puzzle-Generator-Sudoku-Crossword-