

Puzzle Solver: System Design Document

A screenshot of a computer program

AI-generated content may be incorrect.

**1. Architecture Overview**

The system is divided into two main components:

* **Frontend (WPF Application)**: Handles user interactions, input validation, and displays the solution.
* **Backend (Python Scripts)**: Contains the core logic for solving the puzzle using the selected algorithm.

Communication between the frontend and backend is facilitated through standard input/output streams, where the WPF application invokes the Python script as a subprocess, passes the puzzle configuration, and reads the solution output.

**2. Methodology**

**2.1. User Input Handling**

* Users specify the puzzle size and input the initial configuration through the GUI.
* The application validates the input to ensure it contains all necessary tiles and no duplicates.

**2.2. Puzzle Solving Process**

* Based on the selected algorithm, the application invokes the puzzle.py script with the appropriate flag (--astar or --bfs).
* The script reads the puzzle configuration, initializes the State and Game classes, and begins the search process.
* Upon finding a solution, the script outputs the sequence of moves and performance metrics.

**2.3. Displaying Results**

* The WPF application captures the output from the Python script and displays it in the GUI, providing users with a clear view of the solution steps and associated metrics.

**3. Justification for Selected AI Techniques**

* *A Search*\*: Chosen for its efficiency in finding the shortest path using heuristics, making it suitable for larger puzzles where performance is critical.
* **BFS**: Included for comparison purposes and to demonstrate the differences between informed and uninformed search strategies.