AI Project Proposal

AI Car Parking Puzzle Solver

01286342 Artificial Intelligence

Software Engineering Program

By

66011098 NUTHTHAPAT CHALOEMLARPSOMBUT
66011123 PARIN VESSAKOSOL
66011580 PHURINAT PUNMEROD

AI Project Proposal

AI Car Parking Puzzle Solver

Project Description

The *Car Parking Puzzle Solver* project aims to create an intelligent system capable of solving the **Rush Hour** puzzle automatically. In this puzzle, cars of different sizes and orientations are placed on a parking grid. The goal is to move the **target car** out of the grid by rearranging the other cars that block its path.

Our system will use the **A*** (A-star) search algorithm to find the optimal sequence of moves that allows the target car to exit with the fewest steps possible. Each car movement will create a new state in the search tree, and the algorithm will evaluate the cost and heuristic of each possible move until the goal is reached.

To handle the game's logical constraints, such as "cars can only move forward or backward along their orientation" and "no two cars may overlap," the system will integrate **Prolog** as a rule-based logic engine. Python will handle the search process, visualization, and user interface, while Prolog ensures each move is valid before it is applied in the search.

The project's final output will be a **graphical simulation** showing how the AI finds the solution step by step. Users will be able to load different puzzle configurations, observe the AI's reasoning, and even compare different heuristic functions.

This project demonstrates the combination of **Heuristic Search** and **Knowledge Representation** to solve a constraint-based problem efficiently — a perfect example of hybrid AI system design.

The AI Concepts to Apply in this Project (one or two pages)

1. Heuristic Search (A* Algorithm)

- The A* algorithm will serve as the main search technique for finding the optimal sequence of moves.
- Each puzzle configuration is represented as a **state**, and each car movement generates a **new state**.
- A cost function $f(n) = g(n) + h(n)$ will be used, where:
    - $g(n)$ = number of moves made so far
    - $h(n)$ = estimated distance from the target car to the exit + number of blocking vehicles
- The heuristic must be **admissible** (never overestimate the true cost), ensuring the algorithm always finds the shortest possible solution.
- Multiple heuristics will be tested, such as:
    - **Blocking Heuristic:** number of cars directly blocking the target car
    - **Manhattan Distance Heuristic:** grid distance from the target car to the exit

2. Knowledge Representation and Reasoning (Prolog Rules)

- Prolog will represent the puzzle environment as facts and logical rules.

Example rules:

```
can_move(Car, Direction, NewState) :-
    orientation(Car, horizontal),
    not(blocked(Car, Direction, State)),
    move(Car, Direction, State, NewState).
```

- Prolog will verify if a proposed move is legal:
    - No car overlaps another.
    - The move stays within the board boundaries.
    - Cars move only in the correct direction (horizontal or vertical).
- Prolog reasoning will reduce invalid states, improving A* efficiency.

3. Hybrid AI System Integration

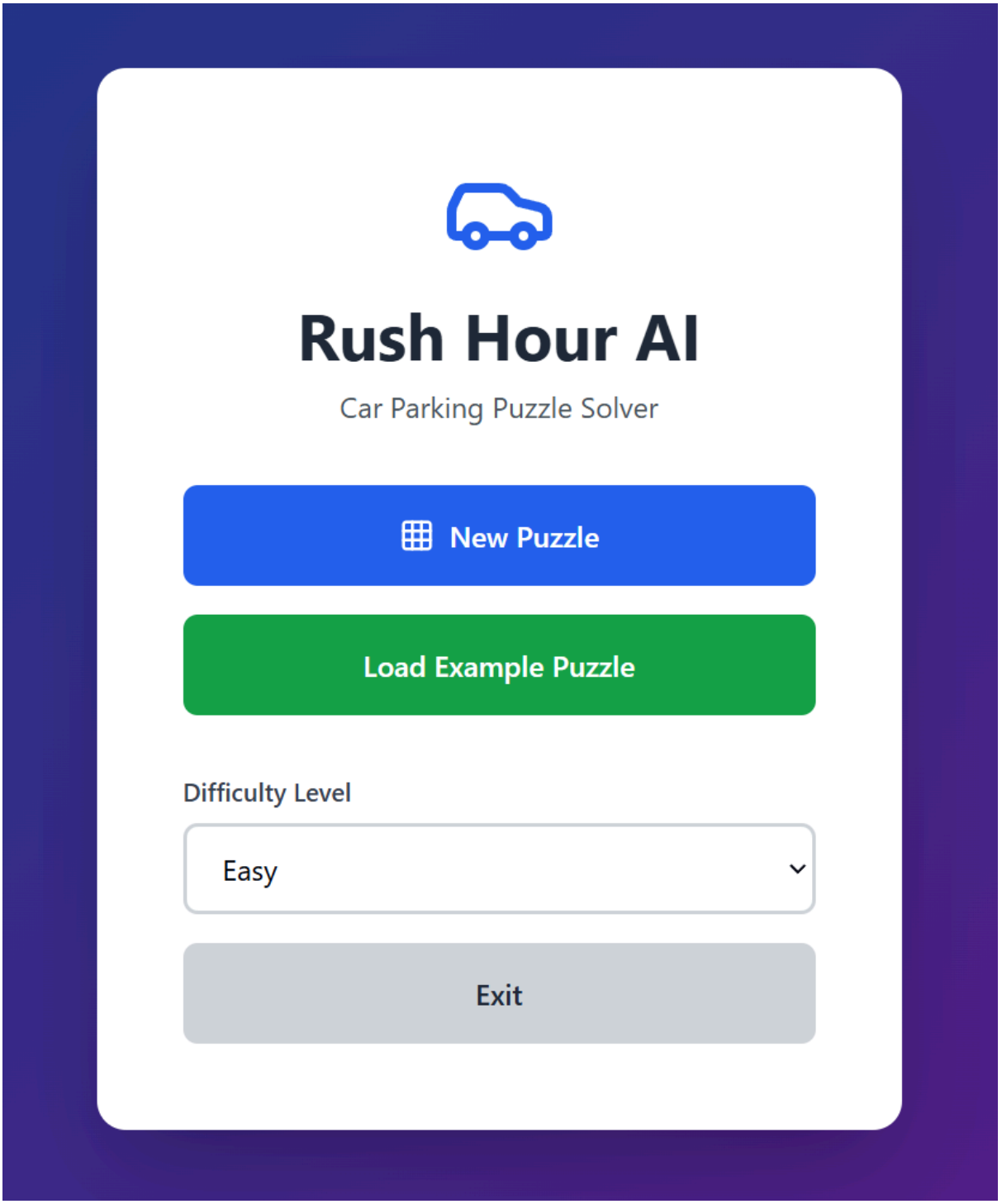
- Python and Prolog will be integrated via **PySwip**, allowing A* to request valid moves from the Prolog knowledge base.
- Python performs the numerical and heuristic computation, while Prolog ensures logical correctness.
- This structure showcases how symbolic reasoning (Prolog) and numerical search (A*) can work together — representing a hybrid intelligent system.

4. Visualization and Explainable AI Concept

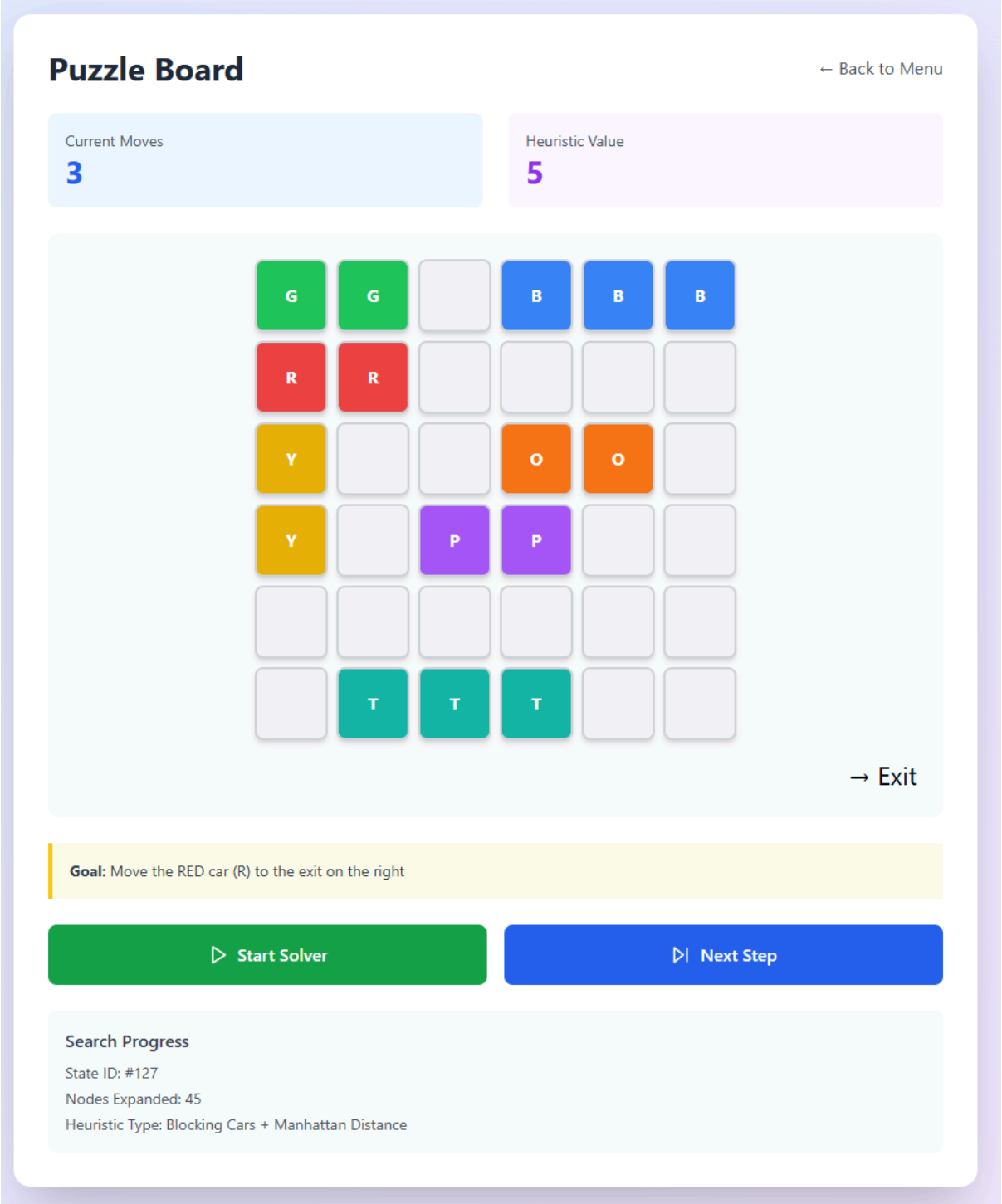
- The project will visualize the solving process on-screen, showing the parking grid, car movements, and the search progress.
- Users can view how the AI decides which car to move next and understand the effect of different heuristics — promoting explainability and transparency in AI decision-making.

Draft of Screen Design

Screen 1 - Main Menu



Screen 2 - Puzzle Board



Screen 3 - Result Screen