

|  |
| --- |
| **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** | 2117240030033 |
| **NAME** | Girijan D |
| **PROJECT TITLE** | Hangman Game |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

* It enables systems to **learn, reason, and make decisions** like humans.  
  AI uses techniques such as **logic, learning algorithms, and pattern recognition** to solve problems and automate intelligent behavior.
* The Hangman AI Game is a simple word-guessing game built using Python. The computer selects a random word, and the player tries to guess it letter by letter. Each guess is checked using logical conditions, showing how AI can make smart decisions. This project demonstrates how **AI concepts like reasoning and decision-making** can be applied in games. It aims to create an **interactive and intelligent system** that responds to the player’s actions, making the gameplay more engaging and dynamic.

**PROBLEM STATEMENT**

* Traditional word games do not adapt or respond intelligently to player inputs.  
  There is a need for a **logic-driven system** that can make decisions based on user actions.
* This project aims to develop an **AI-based Hangman game** that uses **conditional and propositional logic** to simulate intelligent reasoning and provide an interactive gameplay experience.

**GOAL**

* A fully functional **interactive Hangman game** that responds intelligently to player guesses.Demonstrates **decision-making and logical reasoning** using conditional/propositional logic.Provides a user-friendly **graphical interface** for engaging gameplay.
* Add **voice or text-based word input** using NLP. Implement **adaptive difficulty** using AI or machine learning. Extend to **AI-vs-AI or multiplayer modes** for more advanced experimentation. Introduce **hint systems or predictive letter suggestions** based on AI logic.

**THEORETICAL BACKGROUND**

* + Simulates AI reasoning by evaluating each guess and updating the game state.
  + Demonstrates **rule-based decision-making**, a foundation of AI.

**Algorithm Theory:**

1. Initialize the game: choose a word, set counters.
2. Display blanks and available letters.
3. Take input (guessed letter).
4. Check the guess:
   1. Correct → reveal letter.
   2. Wrong → increase wrong guess count, draw hangman.
5. Check game status: win or lose.
6. Repeat until game ends. Optionally allow replay.

* **Existing Solutions:** Most Hangman games use **rule-based conditional logic**, where the system checks each guess and updates the game state. Simple, efficient, and demonstrates AI reasoning.
* **Alternative Algorithms:**

1. **Frequency-Based Guessing:** Uses common letter frequencies to predict guesses.
2. **Tree-Based Search:** Represents all possible words and narrows options using a decision tree.
3. **Probabilistic Models / Markov Chains:** Predicts next letters based on probabilities.
4. **Machine Learning / NLP:** Learns patterns to make intelligent, adaptive guesses.

The Hangman game requires **real-time decision-making** based on user input.

**Rule-Based Decision Algorithm (Conditional Logic)** is:

* 1. **Simple and efficient** for checking letter guesses.
  2. **Easy to implement** in Python using if–else statements.
  3. **Perfect for beginners** to demonstrate **AI reasoning and state updates**.

Allows **dynamic game state updates**, including revealing letters and tracking wrong guesses.

Provides a foundation to **extend the game with advanced AI techniques** like prediction, adaptive difficulty, or machine learning in the future.

**ALGORITHM EXPLANATION WITH EXAMPLE**

* **Algorithm Steps:**
* **Start the game** – choose a random word.
* **Display blanks** for each letter (e.g., \_ \_ \_ \_ \_).
* **Take player input** – guess a letter.
* **Check the guess:**
  + ✅ Correct → Reveal the letter
  + ❌ Wrong → Increment wrong guess count & draw hangman
* **Check game status:**
  + All letters guessed → Player wins
  + Max wrong guesses reached → Player loses
* **Repeat** until game ends.
* **Optionally replay** the game.

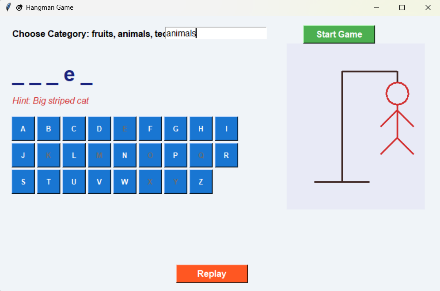
**Example:**

* **Word:** MANGO
* Player guesses: A → \_ A \_ \_ \_ ✅
* Player guesses: E → Wrong → Draw hangman ❌
* Player guesses: M → M A \_ \_ \_ ✅
* Continue until word is complete → Player wins

**IMPLEMENTATION AND CODE**

[Hangman Project PPt.pptx](https://1drv.ms/p/c/981CDF816D363BA6/EemIaioGOg9AqNu09qXyiC8BRB4mW7IpiNguXBfOXyEQjA?e=R5CYdo)

**OUTPUT**

A screenshot of a computer game

AI-generated content may be incorrect.

**RESULTS AND FUTURE ENHANCEMENT**

**Results:**

1. Fully functional **interactive Hangman game** with GUI.
2. Demonstrates **AI-style decision-making** using conditional logic.
3. Tracks player guesses and updates game state dynamically.

**Future Enhancements:**

1. Add **voice or text-based word input** (NLP).
2. Implement **adaptive difficulty** using AI/ML.
3. Introduce **hint systems or predictive letter suggestions**.
4. Extend to **AI-vs-AI or multiplayer modes**.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/Giri6126/Hangman** |

**REFERENCES**

* Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th Edition). Pearson.
* Python Software Foundation. *Python 3 Documentation*. <https://docs.python.org/3/>
* Python Tkinter Documentation. <https://docs.python.org/3/library/tkinter.html>
* GeeksforGeeks. *Hangman Game in Python Tutorial.* <https://www.geeksforgeeks.org/hangman-game-in-python/>
* **Real Python.** *Python GUI Programming: Your Tkinter Tutorial*. [https://realpython.com/python-gui-tkinter/](https://realpython.com/python-gui-tkinter/?utm_source=chatgpt.com)