ARTIFICIAL INTELLIGENCE (5CS 1025)

MINI PROJECT REPORT

ON

RULE-BASED AI STORY GENERATOR USING FORWARD CHAINING

TEAM MEMBERS:

1. N. Mokshitha - 2023BSE07AED155
2. M. Manasa Chowdary - 2023BSE07AED163
3. T. Gopi Hansika - 2023BSE07AED155
4. R. Pallavi - 2023BSE07AED188
5. S. Jnana praveena - 2023BSE07AED104

SUBMITTED TO

Ms. RADHA R

ASSISTANT PROFESSOR

ALLIANCE COLLEGE OF ENGINEERING AND DESIGN

ALLIANCE UNIVERSITY

BENGALURU

ABSTRACT:

This mini project focuses on designing a simple yet effective Artificial Intelligence (AI) system that can generate short stories based on user input. Unlike modern approaches using Machine Learning (ML) or Generative AI (like ChatGPT), this system is built purely on traditional AI techniques — specifically, a rule-based inference system using forward chaining.

The user provides basic inputs such as a character, setting, and theme, and the system processes these inputs through a series of prewritten IF-THEN rules to generate a logical and meaningful story. The result is a fully offline, explainable AI tool that simulates storytelling in a controlled, structured way.

PROBLEM STATEMENT:

Most story generation tools today rely on advanced generative models, which are often black-box systems, large in size, and dependent on large datasets. Our aim is to create a lightweight, explainable AI system that performs the task of storytelling using only rule-based logic.

How can we generate creative and coherent short stories using only forward chaining and rule-based reasoning — without using any ML or GenAI?

OBJECTIVES:

- To simulate creative story generation using AI without ML.
- To implement a rule-based system with forward chaining logic.
- To allow users to input story elements (character, setting, theme).
- To dynamically generate and display the story on a webpage.
- To demonstrate symbolic AI in a simple and engaging use case.

METHODOLOGY:

The system uses forward chaining, where the input facts (character, setting, and theme) are matched against a predefined rule base. Each rule contributes a part of the story. The entire process is done using:

PYTHON FOR BACKEND LOGIC:

- Flask framework for creating the web interface
- HTML + CSS for frontend layout
- No machine learning is used — the logic is entirely transparent and based on human-authored rules stored in Python.

TECHNOLOGIES USE

- Python 3.13
- Flask Web Framework
- HTML5 and CSS3
- Rule-Based Artificial Intelligence

PROGRAM CODE:

app.py : Here's the code for app.py – a simple rule-based AI story generator using forward chaining, implemented in Python. This version runs in the terminal and builds a short story based on user input:

app.py > ...

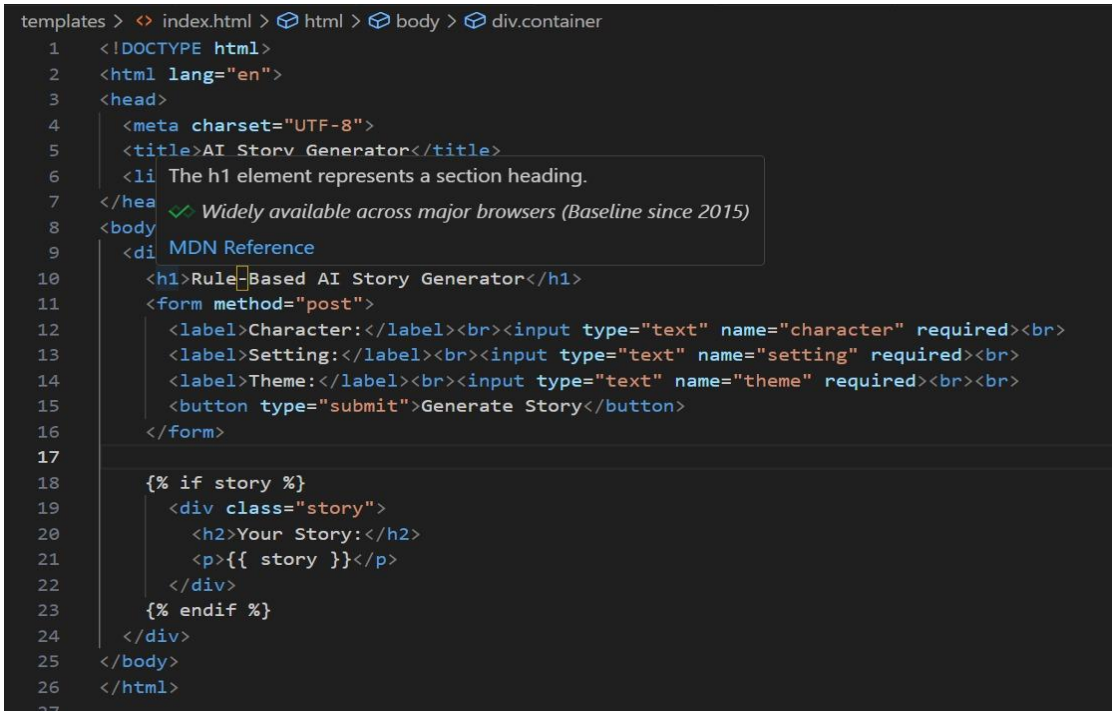
```python
from flask import Flask, render_template, request
from rules import generate_story

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    story = ""
    if request.method == 'POST':
        character = request.form['character'].lower()
        setting = request.form['setting'].lower()
        theme = request.form['theme'].lower()
        facts = {"character": character, "setting": setting, "theme": theme}
        story = generate_story(facts)
    return render_template('index.html', story=story)

if __name__ == '__main__':
    app.run(debug=True)

```

rules.py: Here is a clean version of rules.py, where all the forward chaining rules for the AI story generator are stored separately. This makes your code modular and easy to maintain.

rules.py > ...

```python
def generate_story(facts):
    story_parts = []

    if facts['character'] == "wizard" and facts['setting'] == "forest":
        story_parts.append("Once upon a time in a mysterious forest, there lived a wise old wizard.")
    elif facts['character'] == "knight" and facts['setting'] == "castle":
        story_parts.append("In a mighty castle, a brave knight stood guard day and night.")
    elif facts['character'] == "alien" and facts['setting'] == "space":
        story_parts.append("Far beyond Earth, an alien roamed the galaxy in search of answers.")
    else:
        story_parts.append(f"In a place called {facts['setting']}, there lived a {facts['character']}.

    if facts['theme'] == "adventure":
        story_parts.append("One day, a call for an exciting adventure arrived.")
    elif facts['theme'] == "mystery":
        story_parts.append("A strange event puzzled the land and demanded investigation.")
    elif facts['theme'] == "friendship":
        story_parts.append("A surprising encounter led to an unexpected friendship.")

    if facts['character'] == "wizard":
        story_parts.append("The wizard cast a powerful spell to uncover hidden secrets.")
    elif facts['character'] == "knight":
        story_parts.append("The knight drew his sword and fought valiantly.")
    elif facts['character'] == "alien":
        story_parts.append("The alien decoded an ancient signal leading to a forgotten planet.")

    story_parts.append("And so, peace and wonder returned, leaving behind a tale to remember.")
    return " ".join(story_parts)

```

index.html: HTML5 used in the design of the index.html frontend template.




```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>AI Story Generator</title>
  <li
</hea
<body
  <di
    <h1>Rule-Based AI Story Generator</h1>
    <form method="post">
      <label>Character:</label><br><input type="text" name="character" required><br>
      <label>Setting:</label><br><input type="text" name="setting" required><br>
      <label>Theme:</label><br><input type="text" name="theme" required><br><br>
      <button type="submit">Generate Story</button>
    </form>

    {% if story %}
      <div class="story">
        <h2>Your Story:</h2>
        <p>{{ story }}</p>
      </div>
    {% endif %}
  </div>
</body>
</html>
```

Style.css:

```
body {
  font-family: Arial, sans-serif;
  background-color: #f5f5f5;
  text-align: center;
  padding: 30px;
}
.container {
  background-color: white;
  width: 400px;
  margin: auto;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0 0 10px #999;
}
input, button {
  padding: 10px;
  width: 80%;
  margin-top: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
button {
  background-color: #1d72b8;
  color: white;
  border: none;
  cursor: pointer;
}
button:hover {
  background-color: #135a96;
}
```

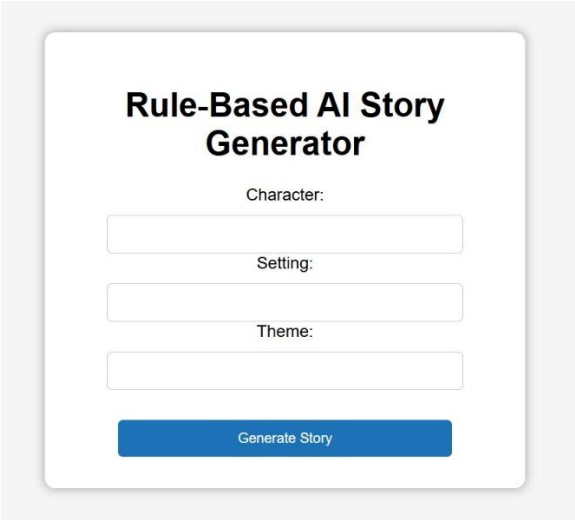
```css
.story {
  margin-top: 20px;
  background-color: #e6f2ff;
  padding: 15px;
  border-radius: 10px;
}
```

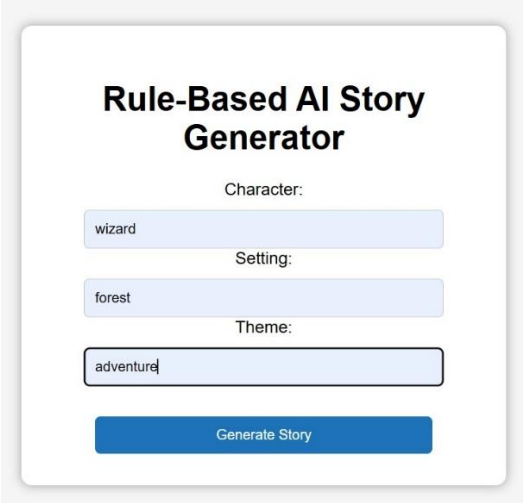
OBSERVATIONS:

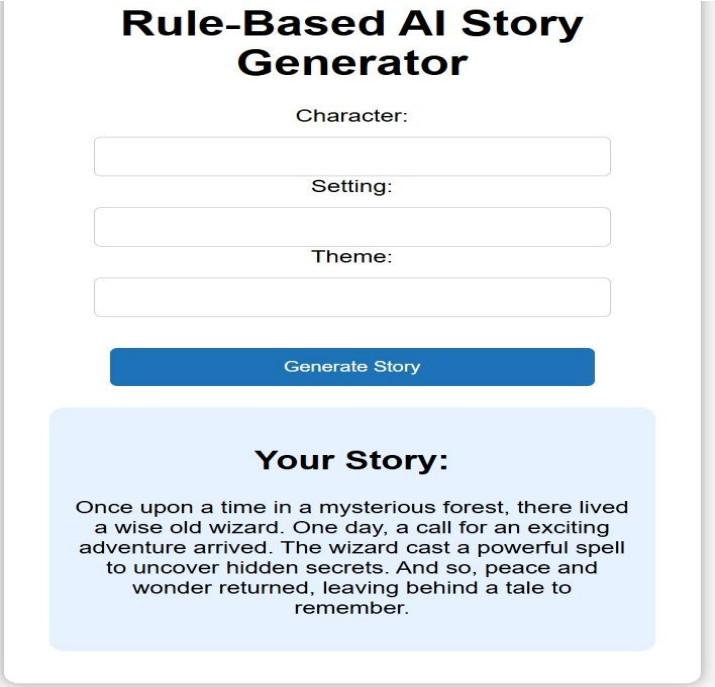
1. Story Generation Works Accurately
   The system successfully generates meaningful and logically connected short stories based on the user's input values (character, setting, theme).
2. Forward Chaining Produces Predictable Results
   Since the project uses forward chaining, the system always selects story lines in a fixed logical flow, ensuring consistency in story structure.
3. No Internet or AI Models Required
   The entire system runs offline and doesn't use any generative models, making it light, fast, and completely explainable.
4. Changing Input Changes, the Output
   Each time the user changes any input (e.g., character = knight vs alien), the story changes accordingly, which shows the rule-based engine is working dynamically.
5. Rules Can Be Easily Expanded
   Adding more combinations or themes is simple by editing rules.py, which shows the project is easily extendable and beginner friendly.

6. Form + Flask Integration is Smooth
   The integration between HTML form and Python backend through Flask works smoothly — the input values are successfully passed, processed, and returned.
7. Simple UI Enhances User Experience
   The clean and minimal user interface (created using HTML and CSS) makes it easy for users to enter data and read stories.

SAMPLE INPUT AND OUTPUT:







ADVANTAGES:

- Simple and transparent logic — easy to explain
- Fully offline and lightweight
- No training data required
- Highly customizable (just edit rules to expand)
- Educational tool to understand symbolic AI

APPLICATIONS

- Kids' storytelling apps
- Text-based role-playing games
- Offline creative tools
- Education to demonstrate AI logic
- AI-assisted writing prompts

FUTURE ENHANCEMENTS

- Add more characters, settings, and themes
- Expand to generate multi-paragraph stories
- Add GUI or mobile app interface
- Allow user-defined characters and custom rules
- Include voice generation using text-to-speech (TTS)

CONCLUSION

This project successfully demonstrates the ability to generate creative content using **rule-based artificial intelligence** and **forward-chaining logic**. It proves that AI applications don’t always require complex ML models — sometimes, symbolic logic is enough. This system is transparent, extendable, and serves as a perfect example of explainable AI in creative domains.