LLM
CIA 1 LAB TEST
STORY GENERATOR

1. Objective

StoryBot is an interactive natural language generation (NLG) system designed to generate short fictional stories based on user-provided prompts. The application is powered by a transformer-based language model (deepseek-r1:1.5b) served via the Ollama framework and wrapped in an intuitive frontend using Streamlit. The primary objectives include genre control via prompt engineering, lightweight inference for real-time interaction, and a minimal preprocessing + postprocessing pipeline for improving text quality. This project is an end-to-end example of real-world LLM deployment, prompt conditioning, and user experience (Streamlit) engineering in AI systems.

2. Dataset Preparation and Motivation

2.1 Synthetic Prompt Dataset

The dataset used in this application is synthetic — it consists of a list of curated prompts written manually to simulate diverse and real-world user inputs. This approach was chosen due to several reasons:

Control: Allows full control over style, domain, and complexity.

Rapid Prototyping: Suitable for evaluating model behavior under varying input conditions.

Low Cost: No need for labeled or annotated corpora.

Code:

```
raw_prompts = [
    "   the lost treasure in the forest   ",
    "A ROBOT WHO learned to DREAM!!",
    "when   the moon    disappeared    for a day.",
    "The cat who saved a city from fire!!!",
    "  ",
    "Sam's Struggle during LLM exam"
]
```

This list includes inputs with inconsistent casing, redundant spaces, and special characters to test the model’s resilience and to simulate natural user input behavior.

3. Preprocessing Pipeline

To ensure cleaner and more deterministic model outputs, a preprocessing function preprocess_prompt(prompt: str) -> str | None is used.

Function: preprocess_prompt()

Purpose:

Cleans and standardizes input prompts before they are passed to the LLM for inference.

Operations:

Whitespace Trimming
Removes leading/trailing whitespace using str.strip().

Multi-Space Normalization
Converts sequences of multiple spaces into a single space using re.sub(r'\s+', ' ', prompt).

Punctuation Stripping
Removes non-alphanumeric symbols using re.sub(r'[^\w\s]', '', prompt) to make input structurally neutral.

Lowercasing + Capitalization
Converts all text to lowercase and capitalizes the first character for syntactic uniformity.

Why It's Used:

Language models tokenize inputs and learn embeddings that are sensitive to format and character distribution. Preprocessing ensures that two semantically identical prompts like " A cat!" and "a cat" are treated similarly by the model, thus reducing variance in outputs.

4. Model Integration via Ollama

Model: deepseek-r1:1.5b

This model is a **decoder-only transformer**, similar in architectural principles to GPT-like models, with 1.5 billion parameters. It is loaded and queried via Ollama's lightweight serving framework.

Function: generate_story(prompt: str, genre: str) -> str

Purpose:

Generates a short story based on the preprocessed prompt and selected genre.

System Prompt Engineering:

genre_instruction = f"Write a short, {genre.lower()} story"

system_prompt = f"{genre_instruction} based on this idea:\n{processed_prompt}\n\nStory:"

Conditioned Prompting: The model is not fine-tuned on genre-specific datasets, so we use **prompt conditioning** to instruct the model about genre context. This approach leverages the model’s in-context learning ability.

Surprise Mode: If the genre is "Surprise Me", it defaults to a general imaginative story.

Model Invocation:

```
response = ollama.chat(
    model=MODEL_NAME,
    messages=[{"role": "user", "content": system_prompt}]
)
```

Ollama provides a low-latency interface to locally hosted LLMs. Here, a chat-style API is used to maintain extensibility with multi-turn input if required in the future.

5. Postprocessing Pipeline

Function: clean_output(text: str) -> str

Purpose:

Removes boilerplate responses and internal LLM reasoning tags to present a clean, human-readable story.

Cleaning Steps:

Thinking Tags Removal:
Removes <think>...</think> sections which may be part of some models' chain-of-thought outputs:

```
re.sub(r"<think>.*?</think>", "", text, flags=re.DOTALL)
```

Boilerplate Statement Removal:
Strips generic LLM-style intros using several regex patterns like:

“Sure, here’s a story…”

“As an AI…”

“Let’s dive in…”

These reduce narrative immersion and appear frequently in assistant models' completions.

Double Newline Normalization:
Ensures the spacing between paragraphs is uniform using:

```
re.sub(r"\n\s*\n", "\n\n", text)
```

Rationale:

This step refines the model output for readability and ensures it appears more like a creative writer’s response than a machine’s reply.

6. Genre Selection and Prompt Conditioning

Component: st.selectbox()

Genre control is implemented via dropdown menus using Streamlit's st.selectbox. This gives users flexibility to generate genre-specific stories like:

Fantasy, Sci-Fi, Romance, Horror, Comedy, Drama, Mystery, Adventure

The selected genre dynamically alters the system prompt passed to the model. Prompt conditioning is a lightweight control mechanism compared to fine-tuning and is ideal for inference-time customization.

7. User Interaction via Streamlit

Component: st.chat_input() and st.session_state

st.chat_input() accepts freeform user text.

st.session_state.chat_history tracks dialog history in a tuple format: (role, message), ensuring state persistence and a conversational interface.

All generated outputs and user inputs are displayed sequentially using:

```
for role, message in st.session_state.chat_history:
    with st.chat_message(role):
        st.markdown(message)
```

This mimics a chat interface but is extensible to future multi-turn LLM interactions or memory-augmented storytelling.

8. Batch Generation

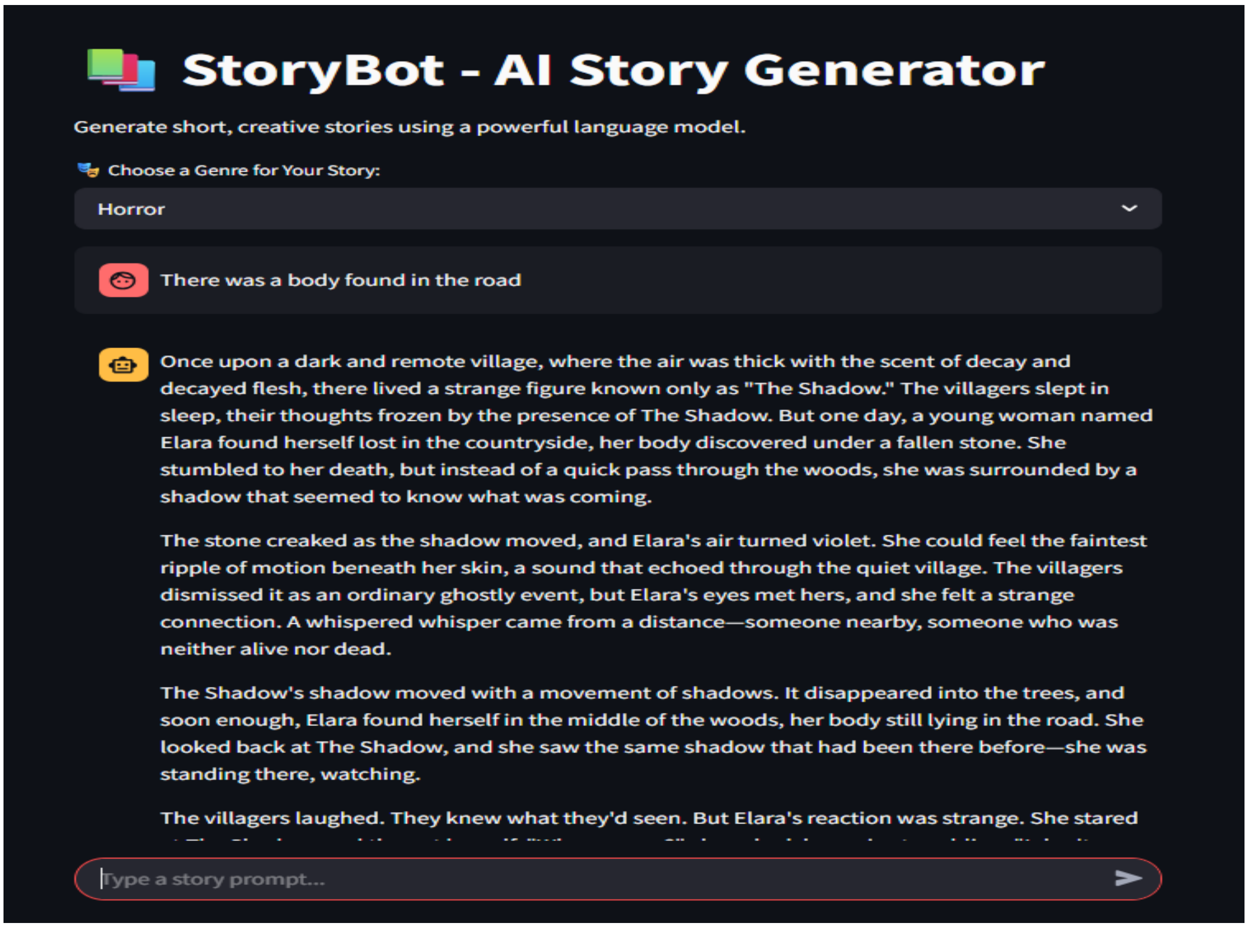
Component: st.expander()

The app supports bulk story generation via a predefined dataset (raw_prompts). The genre is chosen once using a dropdown (batch_genre), and each prompt is run through generate_story().

```
for i, prompt in enumerate(raw_prompts):
    story = generate_story(prompt, batch_genre)
```

This is useful for evaluating model behavior on multiple prompts under consistent genre constraints, aiding both user demonstration and offline testing.

Output:



StoryBot - AI Story Generator

Generate short, creative stories using a powerful language model.

Choose a Genre for Your Story:

Horror

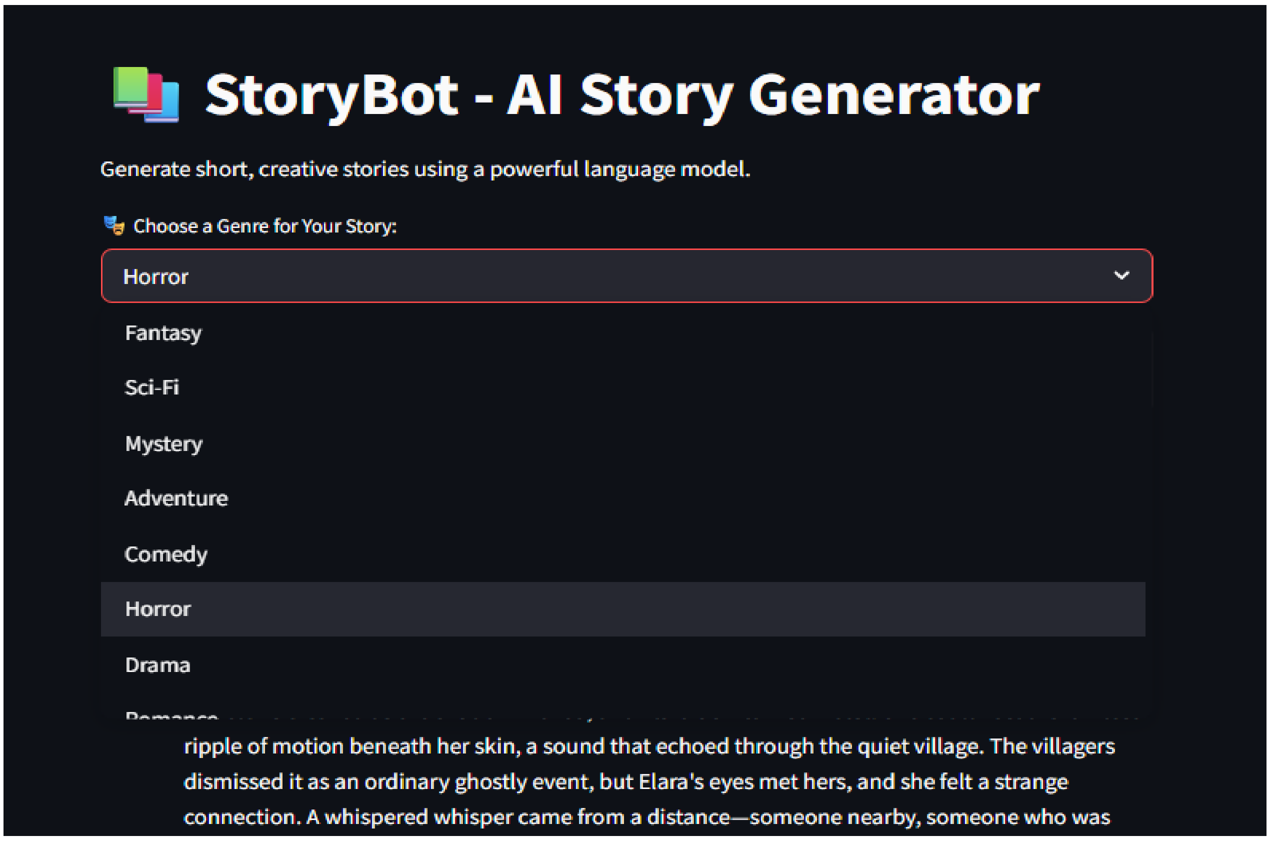
There was a body found in the road

Once upon a dark and remote village, where the air was thick with the scent of decay and decayed flesh, there lived a strange figure known only as "The Shadow." The villagers slept in sleep, their thoughts frozen by the presence of The Shadow. But one day, a young woman named Elara found herself lost in the countryside, her body discovered under a fallen stone. She stumbled to her death, but instead of a quick pass through the woods, she was surrounded by a shadow that seemed to know what was coming.

The stone creaked as the shadow moved, and Elara's air turned violet. She could feel the faintest ripple of motion beneath her skin, a sound that echoed through the quiet village. The villagers dismissed it as an ordinary ghostly event, but Elara's eyes met hers, and she felt a strange connection. A whispered whisper came from a distance—someone nearby, someone who was neither alive nor dead.

The Shadow's shadow moved with a movement of shadows. It disappeared into the trees, and soon enough, Elara found herself in the middle of the woods, her body still lying in the road. She looked back at The Shadow, and she saw the same shadow that had been there before—she was standing there, watching.

The villagers laughed. They knew what they'd seen. But Elara's reaction was strange. She stared

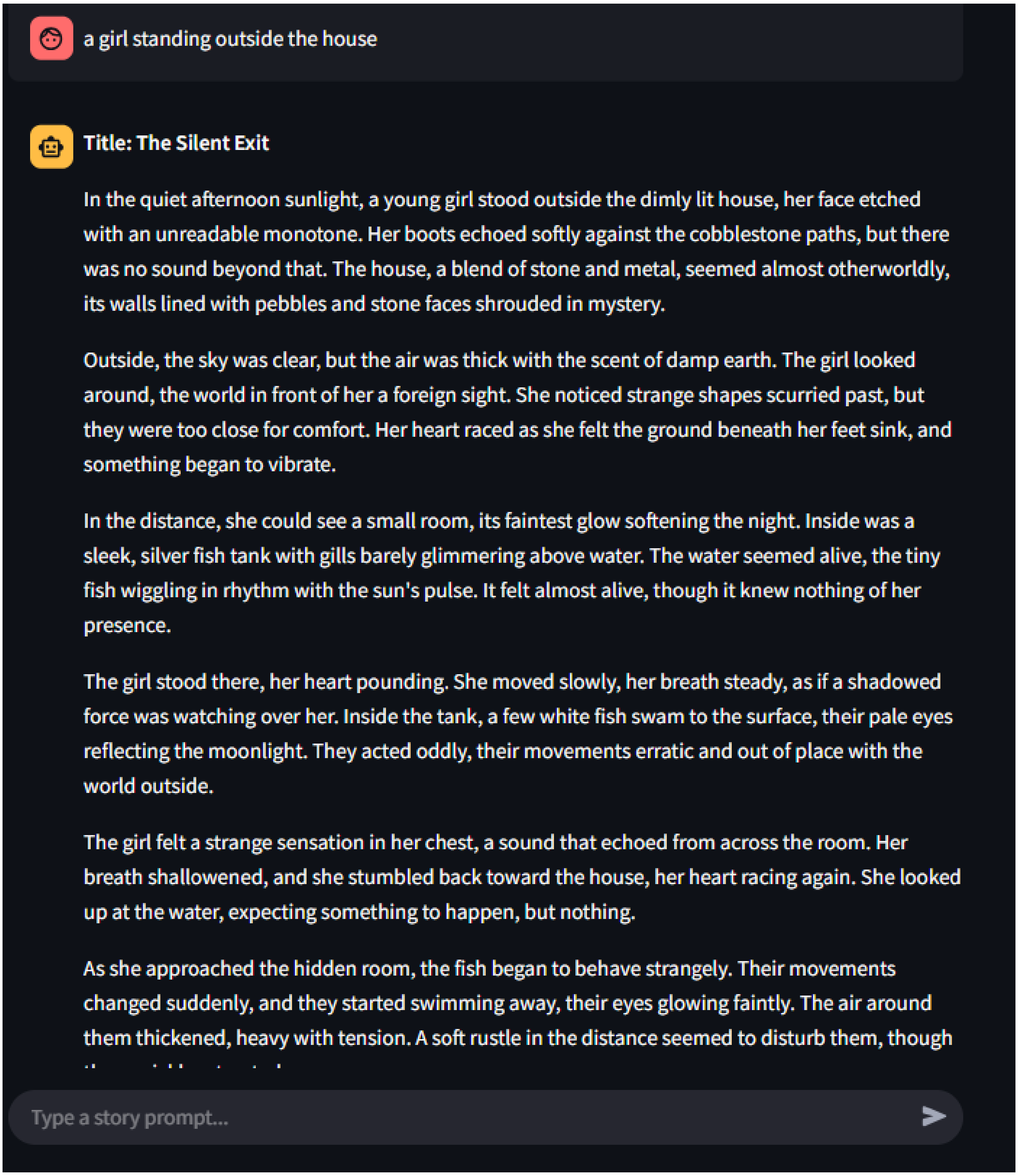


StoryBot - AI Story Generator

Generate short, creative stories using a powerful language model.

Choose a Genre for Your Story:

ripple of motion beneath her skin, a sound that echoed through the quiet village. The villagers dismissed it as an ordinary ghostly event, but Elara's eyes met hers, and she felt a strange connection. A whispered whisper came from a distance—someone nearby, someone who was



a girl standing outside the house

Title: The Silent Exit

In the quiet afternoon sunlight, a young girl stood outside the dimly lit house, her face etched with an unreadable monotone. Her boots echoed softly against the cobblestone paths, but there was no sound beyond that. The house, a blend of stone and metal, seemed almost otherworldly, its walls lined with pebbles and stone faces shrouded in mystery.

Outside, the sky was clear, but the air was thick with the scent of damp earth. The girl looked around, the world in front of her a foreign sight. She noticed strange shapes scurried past, but they were too close for comfort. Her heart raced as she felt the ground beneath her feet sink, and something began to vibrate.

In the distance, she could see a small room, its faintest glow softening the night. Inside was a sleek, silver fish tank with gills barely glimmering above water. The water seemed alive, the tiny fish wiggling in rhythm with the sun's pulse. It felt almost alive, though it knew nothing of her presence.

The girl stood there, her heart pounding. She moved slowly, her breath steady, as if a shadowed force was watching over her. Inside the tank, a few white fish swam to the surface, their pale eyes reflecting the moonlight. They acted oddly, their movements erratic and out of place with the world outside.

The girl felt a strange sensation in her chest, a sound that echoed from across the room. Her breath shallowened, and she stumbled back toward the house, her heart racing again. She looked up at the water, expecting something to happen, but nothing.

As she approached the hidden room, the fish began to behave strangely. Their movements changed suddenly, and they started swimming away, their eyes glowing faintly. The air around them thickened, heavy with tension. A soft rustle in the distance seemed to disturb them, though