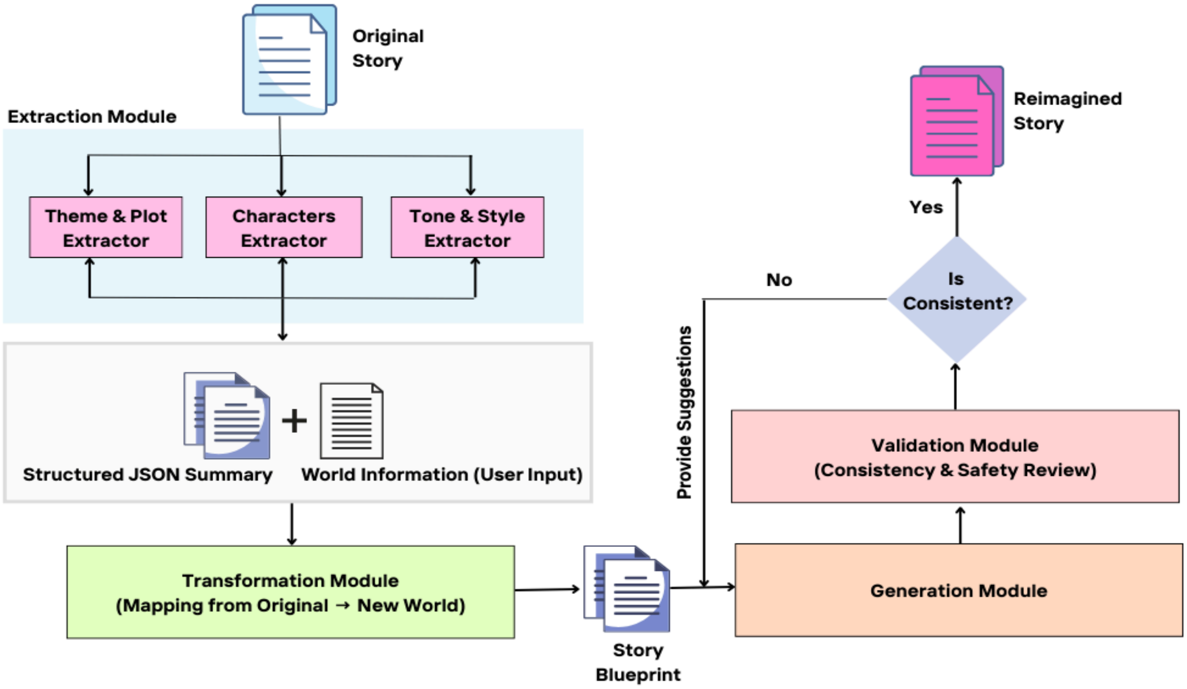
Take-Home Assignment: AI Engineer Intern

Name: Siddharth Nimbalkar Contact: connect.siddharthn@gmail.com

Approach Diagram:



Solution Design:

1. **User Input (Source Story and Target World):**
    a. A public domain story
    b. The new world environment or setting (e.g. cyberpunk city, pandemic affected world, etc)
2. **Extraction Module:**
   Instead of treating the story as one big block, the system breaks it into structured components using parallel LLM calls:
    a. **Theme & Plot Extractor** – identifies the core message and narrative arc
    b. **Characters Extractor** – identifies roles and relationships
    c. **Tone & Style Extractor** – captures how the story "feels"

   The outputs are converted into a **clean JSON summary**. This makes the story machine-readable and ensures consistency in later steps.

3. **Adding World Information:**
   The system loads a small local JSON knowledge base describing the target world. This prevents unrealistic or inconsistent storytelling and keeps the reimagined world logical.
4. **Transformation Module:**
   The system builds a story blueprint by mapping each extracted element into the new context. The result is a structured **blueprint JSON**, not yet a story.
5. **Generation Module:**
   Using the blueprint as input, the LLM powered system generates a full draft story. At this stage, the story is creative, but still checked for quality.
6. **Validation Module:**
   A dedicated critic module that ensures the story is respectful, internally logical, and avoids 'magic fixes' that break the flow. It returns structured feedback (OK / Not OK along with guidance).

If the critic module finds issues, the system makes revisions without changing the core story. It repeats this a few times, and once approved, the story moves forward.

Alternatives Considered:

1. **Structured Pipeline** over Fully Prompt-Based System**.**
2. **Reliability** over Speed.
3. **Parallel Extraction** over Serial Processing.
4. Mixt of **Few-Shot** and **Zero-Shot Prompting** depending on the use.
5. **LLMs** vs. **Classical NLP** for extraction modules.

Challenges and Mitigations:

1. **Single-prompt extraction was unreliable and impossible to debug.** When I tried extracting themes, characters, plot, and tone in one big prompt, some parts came out right, but others were partially wrong. **So, I split extraction into separate role-based pipelines** which helped me debug each extractor easily and independently.
2. **Validation loop sometimes made the story worse.** When validation flagged issues, the rewrite would occasionally rewrite too much, changing tone or story meaning. **I constrained rewrites to *"fix only what validation mentions"* and added a limit on iteration loops.**

Future Improvements:

1. We can wrap each module as a service(Extraction, Transformation, Generation, Validation), which will enable the user to call only the module they need.
2. We can fine-tune smaller models for extraction to reduce hallucinations, while also using in-context examples so the system learns patterns without retraining.