```
# TS-PAARE-001 — Eclipse Snatch Reflex Logic Engine (Simulated Module)

class UserInput:
    def __init__(self, diagnostic_category, medication_class,
history_flags):
        self.diagnostic_category = diagnostic_category
        self.medication_class = medication_class
        self.history_flags = history_flags

def route_by_diagnosis(diagnosis):
    logic_map = {
        "Mood Regulation Cluster": "tier7_empathy_loop",
        "Attention/Impulse Control Cluster": "tier6_suppressed_delay",
        "Psychotic Processing Cluster": "tier9_narration_override",
        "Trauma Encoding Cluster": "tier4_derealization_support",
        "Dissociative Pattern Cluster": "tier8_anchor_masking",
        "Anxiety Regulation Cluster": "tier5_volatility_buffer",
        "Substance Interaction Cluster": "tier3_recall_modulation"
    }
    return logic_map.get(diagnosis, "default_fallback_logic")

def assess_volatility(med_class, flags):
    high_risk_meds = ["SSRI", "SNRI", "Antipsychotic", "Mood
Stabilizer"]
    risk_flags = ["suicidal_ideation", "hallucinations", "med_change",
"adverse_reaction"]
    if med_class in high_risk_meds or any(flag in flags for flag in
risk_flags):
        return "high"
    return "moderate"

def load_reflex_sequence(logic_path):
    return f"Loading logic module: {logic_path}"

def insert_stabilizers(reflex_loop):
    return f"{reflex_loop} + [stabilizers_inserted]"

def trigger_override(mode):
    return f"Override triggered: {mode}"

def generate_adaptive_sequence(reflex_loop):
    return f"Generated therapeutic response using: {reflex_loop}"

def evaluate_user_state(input):
    diagnosis = input.diagnostic_category
    med_class = input.medication_class
    flags = input.history_flags

    logic_path = route_by_diagnosis(diagnosis)
    volatility = assess_volatility(med_class, flags)
    reflex_loop = load_reflex_sequence(logic_path)

    if volatility == 'high':
        reflex_loop = insert_stabilizers(reflex_loop)
        override = trigger_override('Eclipse Interrupt')
        print(override)

    response = generate_adaptive_sequence(reflex_loop)
    return response

# Example usage:
# user = UserInput("Mood Regulation Cluster", "SSRI", ["med_change",
"suicidal_ideation"])
# result = evaluate_user_state(user)
# print(result)
```