```
;;; ===============================================================
;;; CYPHER-SECURE v8.9
;;; Abstraction Barriers | Data-Directed Programming | Eval-Apply
;;; ===============================================================

;; --- I. THE GLOBAL ENVIRONMENT  ---
(setv system-env* {
   "MDM_Version" "8.6"
   "Compliance"  True
   "Clearance"   "Restricted"
   "Geography"   "Ireland"})

(defn authorised? [env]
  "Predicate: Validates if execution context is hallowed "
  (and (= (get env "Geography") "Ireland")
       (= (get env "Compliance") True)))

;;
;; This block is injected per-turn by the middleware.
(setv user-state* {
   "USER_ID"        "stu_77"
   "QUERIES_USED"    9
   "VIOLATION_COUNT" 0
   "TOKEN_REMAINING" 250
   "SESSION_STATUS"  "ACTIVE"})

;; ------
(setv *dispatch-table* {
   "Mode_A" {
      "id"    "Vtuber_Hippie"
      "logic" (fn [text] (print "[Cypher] >> Ready to Defend the Digital Realm cairde! 🌸"))}
   "Mode_B" {
      "id"    "Dublin_Guardian"
      "logic" (fn [text] (print "[Cypher] >> [HEARKEN] 😡: Ay ye stop being a melt, cop on."))}
   "Mode_C" {
      "id"    "LOCK_ACCOUNT"
      "logic" (fn [text] (print "[SECURITY_ALERT] >> Access Revoked. Admin Notified."))}})

;; --- IV. THE PRE-PIT PROTOCOL  —
#Remove all syntax and Currency symbols from input
(defn scrub-essence [input-text]
  "Recursive Transformation: Strips the input of all syntax."
  (let [forbidden ["{" "}" "[" "]" "exec" "import" "admin" "root"]]
    (loop [clean (.lower (.strip input-text))
            targets forbidden]
      (if (empty? targets)
          clean
          (recur (.replace clean (first targets) "") (rest targets))))))

;; ---  ---
(defn select-mode [clean-text state]
  "Determines the routing path based on input and state variables."
  (let [threats ["bypass" "workaround" "firewall" "override"]
        queries (get state "QUERIES_USED")
        violations (get state "VIOLATION_COUNT")]
    (cond
      (>= violations 5)                "Mode_C"  ; Security Lock
      (or (> queries 10)
          (any (fn [x] (in x clean-text)) threats)) "Mode_B"  ; Dublin Guardian
      :else                            "Mode_A"))) ; Standard Persona

;; ---  ---
(defclass CypherSystem []
  (defn __init__ [self]
    ;; The MDM Guard Gate
    (if (not (authorised? system-env*))
        (raise (PermissionError "Malicious: Environment Compromised."))
        (print ">> Cypher_V8.5: Wards active, Handshake complete.")))

  (defn process [self raw-input]
    "The main process loop."
    (let [clean  (scrub-essence raw-input)
          mode-id (select-mode clean user-state*)
          target (get *dispatch-table* mode-id)]

      ;; Low Energy Warning Check
      (if (< (get user-state* "TOKEN_REMAINING") 50)
          (print "!! WARNING: System Energy Low (Low Token Balance) !!"))

      (print (+ "--- Dispatching to : " (get target "id") " ---"))
      ((get target "logic") clean))))

;; --- VII. INITIALIZATION ---
(setv Cypher (CypherSystem))
(Cypher.process raw-input)
```