SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

A RESILIENT HYBRID-AI ARCHITECTURE FOR PERSONALISED DIGITAL WELLNESS: DESIGN AND EVALUATION OF HYPNARA

Thi Hien PHAN[1], Viet Tien LE[2]
[1]*University of Transport and Communications, Vietnam*
[2]*Luong The Vinh Secondary and High School , Vietnam*

Article DOI: https://doi.org/10.36713/epra31823
DOI No: 10.36713/epra31823

ABSTRACT
Digital wellness applications increasingly rely on large language models (LLMs) for personalised guidance, yet dependence on remote services introduces significant reliability risks. This study presents Hypnara, a full-stack digital wellness application using a resilient hybrid-AI architecture that integrates a cloud LLM with a local deterministic rule engine. Following design science research, the system was evaluated through specification-based verification and controlled fault-injection experiments. The architecture incorporates bounded behavioural context, a unified response contract, deterministic rule prioritisation, and allow-list prompt projection to restrict externally transmitted data. Under injected HTTP errors and service delays, the cloud-only configuration achieved valid-response rates of 90%, 70%, and 50% at fault rates of 10%, 30%, and 50%, respectively, whereas the hybrid configuration maintained 100% valid responses. When provider delays exceeded the one-second timeout, the cloud-only path failed completely, while the hybrid path preserved service through fallback, with p95 latency near the timeout threshold. Context-window testing further showed that shorter histories substantially reduced payload size. The design therefore supports graceful degradation during external AI service disruptions. Overall, Hypnara demonstrates that hybrid AI improves service resilience by preserving deterministic support while retaining cloud-based personalisation when available.
KEYWORDS: *Hybrid AI; digital wellness; resilient software architecture; fault injection; graceful degradation; large language models; rule-based systems; personalised recommendation.*

1. INTRODUCTION

Digital technologies are embedded in everyday work, learning, communication, and sleep routines. This creates a design tension for digital wellness systems: the same device can support self-monitoring while also enabling prolonged screen exposure, frequent checking, and late-night use. Digital well-being is therefore broader than a screen-time limit and concerns whether technology use remains compatible with wider physical, psychological, and social functioning (Smits et al., 2022). Reviews accordingly emphasise awareness, self-regulation, friction, and context-sensitive support rather than a single restriction mechanism (Almoallim & Sas, 2022).

Sleep-related routines are a suitable application domain because device use and sleep behaviour often interact. Greater smartphone use is associated with poorer self-reported sleep outcomes (Chu et al., 2023), while sleep-focused mobile interventions commonly combine monitoring, feedback, goals, and planning (Arroyo & Zawadzki, 2022). These findings motivate systems that connect digital activity with daily routines, but they do not justify treating a wellness application as a clinical diagnostic tool.

Artificial intelligence has expanded the design space for such applications. Large language models can convert structured behavioural context into fluent, varied, and seemingly personalised natural-language guidance. In principle, an LLM can interpret multiple signals at once: sleep duration, screen time, exercise, phone disengagement before bedtime, gaming, mood, and recent behavioural history. The difficulty is that personalisation then becomes dependent on an external service. Authentication failures, rate limits, provider outages, malformed responses, long response times, and network loss can all propagate into a user-visible failure if the LLM lies directly on the critical request path. In health-related computing, reliability, privacy, interpretability, and response consistency are already recognised concerns for conversational LLM systems (Wang et al., 2024). Resilient AI research therefore argues that systems should be designed for uncertainty and partial degradation rather than assuming that an intelligent component will always be available (Sáez et al., 2024; Taimoor & Rehman, 2022).

Hypnara addresses this problem as a software-architecture artefact. The application combines a cloud LLM with a deterministic local rule engine, but the intended contribution is more specific than a simple try-catch fallback. The revised architecture treats external AI as an enrichment dependency behind a failure-aware orchestration contract. Both cloud and local paths receive the same bounded behavioural context, return the same application-level response type, and expose provenance so that the client can identify degraded operation. The local path is formalised as a prioritised rule system that selects a small number of actionable recommendations from seven behavioural domains. The cloud path receives only an allow-listed projection of the user

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

context. Time-zone-aware calendar logic, semantic missing-data handling, and commitment-state constraints are placed outside the cloud dependency so that these correctness properties remain local and testable.

Passing unit tests demonstrates correctness for known cases but does not quantify behaviour when the external provider becomes unreliable. Fault injection is therefore needed to determine whether faults are contained rather than merely anticipated in code (Hsueh et al., 1997; Natella et al., 2016). The study injects HTTP failures, rate-limit responses, malformed payloads, and delays beyond the client timeout, comparing cloud-only, rule-only, and hybrid configurations through valid-response rate, fallback coverage, and latency percentiles.

Four research questions guide the study. RQ1 asks how a cloud-plus-local architecture can preserve a valid personalised-wellness response when the external model fails. RQ2 asks how the deterministic fallback can be formalised so that its decision process is reproducible and inspectable. RQ3 asks how a bounded context window and prompt-projection layer can balance recent behavioural information, transmission size, and privacy exposure. RQ4 asks what availability and latency trade-offs arise when faults and delays are introduced into the external AI path.

The paper makes four contributions. First, it presents a failure-aware hybrid-AI reference architecture in which generative AI is non-essential to the minimum recommendation service. Second, it formalises the local recommendation engine as a prioritised rule-selection and template-filling process rather than as an unspecified backup function. Third, it provides quantitative evidence from a controlled fault-injection campaign, including response validity and p50/p95/p99 latency under increasing external failure and delay. Fourth, it introduces an explicit bounded-context and privacy-projection design, together with a sensitivity analysis of seven-, ten-, fourteen-, and thirty-day history windows. The contribution remains within software engineering: the study evaluates service continuity, correctness, and architecture-level trade-offs, not clinical efficacy or long-term behavioural change.

2. LITERATURE REVIEW

Digital wellness research provides the application context for Hypnara. Smits et al. (2022) distinguish digital well-being from the narrower idea of delivering health care through digital technology. Digital well-being also concerns the consequences of technology use itself and the user's capacity to maintain beneficial patterns of interaction. Almoallim and Sas (2022) reviewed digital well-being applications and identified recurring functions such as monitoring, limit awareness, friction, attention support, and motivational support. This literature suggests that an effective software design should connect measurement with an actionable self-regulation loop rather than present isolated statistics.

Behavioural theory also informs the software structure. Fogg's behaviour model states that behaviour is most likely when motivation, ability, and a prompt converge (Fogg, 2009). A digital system can operationalise these components through progress feedback, reduced data-entry effort, reminders, and action commitments. Just-in-time adaptive intervention research adds a more formal vocabulary: decision points, tailoring variables, intervention options, and decision rules (Nahum-Shani et al., 2018). From a software perspective, this is directly relevant to personalisation because it encourages explicit representation of which variables are observed, when a decision is made, and how an intervention is selected.

LLMs change the implementation of the decision layer but do not remove the need for explicit system boundaries. A cloud model can combine multiple behavioural variables and produce natural language that is more flexible than fixed templates. At the same time, the model introduces external operational dependencies and less deterministic output. Wang et al. (2024) identify reliability, privacy, safety, and consistency as recurring concerns for conversational LLM use in health care. Although Hypnara is a wellness application rather than a clinical system, the architectural lesson is similar: an LLM request should not be allowed to become an unexamined single point of failure for basic support.

Dependability engineering provides the theoretical basis for containing that failure. Avizienis et al. (2004) define fault tolerance as a means of avoiding service failure in the presence of faults. Graceful degradation applies this principle by preserving a reduced but valid service when a higher-level component cannot operate. For an AI-enabled application, the relevant question is not whether every response remains equally rich, but whether the system continues to meet a defined minimum service contract. In Hypnara, that contract is a valid, actionable recommendation response derived from the user's current behavioural state, recent history, and goals.

Fault injection is an established method for testing whether such claims hold under adverse conditions. Hsueh et al. (1997) describe fault injection as deliberate introduction of faults to study system behaviour, while Natella et al. (2016) emphasise software fault injection as a practical means of assessing dependability mechanisms. This is particularly relevant to API-dependent AI systems because many important faults are simple to reproduce at the service boundary: HTTP errors, rate limits, connection failures, excessive delay, and malformed output. A resilient architecture should demonstrate not only that an exception handler exists, but that injected faults are converted into controlled state transitions and valid service responses.

In this study, hybrid AI has a narrow engineering meaning: the cloud LLM supplies generative flexibility, whereas the local engine supplies deterministic continuity. The two paths are not assumed to be equivalent in expressive quality. Their value lies in complementary non-functional properties: richer contextual synthesis versus repeatability, inspectability, local execution, and bounded behaviour (Sáez et al., 2024; Taimoor & Rehman, 2022).

The local rule engine also addresses a criticism commonly raised against fallback systems: a fallback that is not explicitly specified cannot be evaluated. A rule-based component should therefore expose its decision structure. Rather than representing the local engine as Rule(C) without definition, the revised design models each rule as a predicate, priority, action identifier, and

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

rendering template. Activated rules are ranked, de-duplicated, and limited to a small number of recommendations. This provides a deterministic mapping from behavioural state to output and creates a testable lower bound on service quality.

Privacy introduces another reason to bound personalisation context. Behavioural histories, mood notes, goals, and schedules can reveal sensitive patterns even when an application is not clinical. Sending an unrestricted database record to a third-party model is therefore unnecessary and difficult to justify. The architecture should minimise transmission by selecting only fields required for the decision. Wang et al. (2024) note privacy concerns around conversational AI in health contexts; the same concern supports an allow-list projection in wellness software. Bounded history also reduces the amount of personal information exposed per request and stabilises prompt size.

A ten-day window has no established clinical status and is not treated as optimal. It is an engineering budget: seven days capture one weekly cycle, while ten days retain that cycle plus three recent days without expanding context as aggressively as fourteen or thirty days. Payload-size sensitivity can test this engineering choice; only a user study could determine whether different windows change recommendation relevance.

The study follows design science research because the research object is an implemented architecture evaluated against explicit technical requirements (Hevner et al., 2004; Peffers et al., 2007). The gap is therefore integrative: behavioural context, LLM-based personalisation, and fault-tolerance principles need to be combined into a reproducible architecture with measurable failure behaviour.

3. METHODOLOGY

The DSR process comprised five stages: problem identification, design requirements, artefact construction, technical verification, and controlled fault evaluation. The central requirements were R1: preserve a valid recommendation when the cloud AI is unavailable; R2: keep the fallback deterministic and inspectable; R3: isolate calendar and data-integrity logic from external AI; R4: bound the historical context supplied to personalisation; and R5: minimise the behavioural information transmitted to the cloud provider. This study used design science research to construct and evaluate Hypnara as a software-architecture artefact. The implementation uses Next.js 14 with TypeScript, React client components, server-side route handlers, PostgreSQL 16, and Docker-based deployment. The application contains modules for authentication, daily habit records, profile and reminder management, wellness scoring, commitments, motivation, data export, conversational support, and personalised suggestions. The evaluation concerns architecture-level correctness and resilience; no clinical or behavioural-effectiveness inference is made.

Personalisation is represented by three input classes. Let X_t be the current-day behavioural state, containing sleep duration, screen time, game time, exercise minutes, pre-sleep phone-disengagement minutes, phone pickups, and mood score. Let H_t^w be the previous w days of available behavioural history and G_t the user's active goals. The internal decision context is:

$$C_t^w = (X_t, H[t-1:t-w], G_t) \quad (1)$$

The production specification fixes w = 10. This value is treated as a bounded engineering choice, not an empirically optimal behavioural horizon. To examine the sensitivity of this choice, the benchmark constructed equivalent context objects for $w \in \{7, 10, 14, 30\}$ and measured serialized payload size and mean serialization time over 10,000 repeated operations. The test addresses computational and privacy exposure, not behavioural effectiveness.

The revised cloud boundary applies a prompt projection π before transmission. The projection uses an allow-list: the seven behavioural measures, relative historical ordering, and selected goal labels may be included, while account identifiers, database keys, session information, timestamps not required for ordering, and free-text notes are excluded. The cloud context is therefore:

$$C_{cloud} = \pi(C_t^{10}), \text{ where } \pi(C) \subset C \quad (2)$$

This projection gives data minimisation a concrete software location. The application can retain richer records locally while transmitting only the fields required for the recommendation request. It also reduces the chance that implementation changes to the database schema silently enlarge the prompt.

The recommendation service uses a common response contract across cloud and local execution. Let q_t be the observed external-service state for a request, derived from configuration availability, HTTP status, parse validity, and a client timeout budget τ. The recommendation function is:

$$A_t(C_t, q_t) = \{ \text{LLM}(\pi(C_t)), \text{ if } q_t = \text{usable}; \ \text{Rule}(C_t), \text{ otherwise } \} \quad (3)$$

A cloud response is usable only when the request finishes before τ, returns an accepted HTTP status, and produces a parseable response containing the required recommendation field. Any other state is converted into degraded local execution. The reference implementation uses exception-triggered routing, but Equation (3) defines the architectural contract: the exception mechanism is an implementation detail, whereas the requirement is that all recognised cloud failures terminate in the local path rather than at the user interface.

The orchestration code can be reduced to the following TypeScript pattern:

```
export async function getAiSuggestion(current, history, profile) {
  const context = buildContext(current, history, profile);
  try {
    const cloudContext = projectForCloud(context);
    const suggestion = await callCloudAI(cloudContext, { timeoutMs: 1000 });
    return { suggestion, mode: 'cloud' };
  } catch (error) {
```

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

```
    return {
      suggestion: generateRuleBasedSuggestion(context),
      mode: 'local-fallback'
    };
  }
}
```

The local engine is formalised as a finite set of rule objects. For rule r_j, $P_j(C_t)$ is its predicate, ω_j its priority weight, a_j its action identifier, and T_j its text template. Rule activation and priority score are:

$$z_j = I[Pj(C_t) = true], \quad q_j = \omega_j z_j \tag{4}$$

Activated rules are ordered by q_j, duplicate action families are removed, and the top K = 3 actions are returned. The rendered recommendation uses slot filling from the same context rather than open-ended generation:

$$Rule(C_t) = Render(TopK(\{r_j : z_j = 1\}), Slots(C_t,G_t)), \quad K = 3 \tag{5}$$

The design uses seven rule families: sleep duration, screen exposure, gaming, exercise, pre-sleep phone disengagement, phone pickups, and mood. The technical specification provides explicit conditions for sleep, screen time, gaming, disengagement, and exercise; pickups and mood are contextual modifiers rather than medical thresholds. Rule priorities are versioned configuration values so the fallback remains testable when thresholds change.

```
type WellnessRule = {
  id: string;
  priority: number;
  when: (c: Context) => boolean;
  action: string;
  template: (c: Context) => string;
};
const selected = rules
  .filter(r => r.when(context))
  .sort((a, b) => b.priority - a.priority)
  .filter(deduplicateActionFamily)
  .slice(0, 3);
```

Temporal correctness is independent of AI. Because the target deployment uses Asia/Ho_Chi_Minh while cloud servers commonly use UTC, naive UTC conversion can classify an early-morning local entry as the previous day. Hypnara derives the civil date explicitly and performs calendar arithmetic through UTC-midnight values. If D is the recorded-date set and d_0 is today when recorded, otherwise yesterday, the streak is:

$$S_t = \max\{k \geq 0 : \{d_0, d_1, ..., d_{(k-1)}\} \subseteq D\} \tag{6}$$

where each $d_{(i+1)}$ is exactly one calendar day before d_i. This design keeps streaks, 'yesterday' copying, reminders, and commitment dates consistent around midnight and month/year boundaries.

The wellness score is likewise kept local. It is a transparent behavioural indicator calculated from sleep, screen time, gaming, phone disengagement, and exercise. In compact form:

$$W = clip_[30,99](100 - P_{sleep} - P_{screen} - P_{game} - P_{cutoff} - P_{exercise}) \tag{7}$$

The score is not computed when there is no sleep observation in the evaluation window:

$$N_{sleep} = 0 \Rightarrow W = null \tag{8}$$

This avoids conflating missing data with a poor wellness state. Commitment integrity is enforced by date constraintsby a commitment c with target date T_c and the current date T_0:

$$T_c > T_0 \Rightarrow checkIn(c) = rejected \tag{9}$$

The application also validates input ranges and prevents reversal of an already completed commitment through the normal check-in route. These controls are evaluated separately from AI because their correctness must not depend on the external model.

Technical verification used the supplied unit-test suites for date handling, validation, wellness scoring, commitments, fallback behaviour, and specification compliance. Because the project documentation reports all suites as passing but contains an inconsistency between the aggregate test count and the per-suite enumeration, this paper reports the verified all-pass suite status without deriving a potentially misleading total from those two descriptions.

The quantitative evaluation used Node.js 22.16.0 on a container with five AMD EPYC 9V74 virtual CPUs and 5.8 GiB memory. A local HTTP service emulated the external AI boundary, allowing exact control of status codes and delays. The benchmark measures orchestration and recovery behaviour, not the production latency of DeepSeek or another commercial LLM.

Three configurations were compared: rule-only, cloud-only, and hybrid. The fault-rate campaign used a fixed 250 ms provider delay, forty requests per condition, and concurrency of ten. HTTP 500 failures were injected at 0%, 10%, 30%, and 50%. Additional campaigns injected HTTP 429 for all requests, malformed JSON responses, and provider delays of 500 ms, 1,500 ms, and 5,000 ms. The client timeout τ was fixed at 1,000 ms. The HTTP 429 and malformed-response campaigns used thirty requests per configuration; delay scenarios used twenty. A warm-up run preceded measurement.

The main availability metric was valid-response rate (VRR):

$$VRR = N_{valid} / N_{requests} \tag{10}$$

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

Fallback coverage (FC) measured whether detected cloud failures were converted into valid local results:

$$FC = N_{\text{valid fallback}} / N_{\text{detected cloud failure}} \quad (11)$$

Latency was measured from recommendation-call entry to the returned application-level result. Percentiles were calculated over the observed request latencies:

$$L_p = \text{Quantile}_p(\{L_1, L_2, ..., L_n\}), \; p \in \{0.50, 0.95, 0.99\} \quad (12)$$

The benchmark also recorded the serialized byte size of the prompt-eligible context for each history window. Because the provider was emulated locally, no monetary API cost is reported. Likewise, the experiment does not infer user-perceived quality from response validity. A valid fallback means that the architecture preserved the minimum response contract; it does not mean that users would judge a rule-based message equivalent to a generative response.

4. RESULTS

The evaluation shows that Hypnara's resilience is measurable at the service boundary rather than only visible in source-code structure. The supplied verification suites passed for temporal logic, input validation, wellness-score behaviour, commitment constraints, local fallback behaviour, and specification compliance. These tests establish that the high-risk domain rules can be executed independently of the cloud model. The fault-injection campaign then examined what happens when the external dependency becomes unreliable.

Under normal operation with a fixed 250 ms provider delay, the cloud-only and hybrid configurations both returned valid responses for all requests. Their latency distributions were also similar: p95 was 262.2 ms for cloud-only and 258.3 ms for hybrid. The difference is small relative to the injected provider delay and indicates that the hybrid routing layer adds no material penalty while the cloud path succeeds.

When HTTP 500 failures were introduced, valid-response rates diverged as expected. At a 10% external fault rate, the cloud-only configuration returned valid responses for 90% of requests, while the hybrid configuration returned valid responses for 100% and activated fallback for exactly 10%. At 30% faults, cloud-only validity decreased to 70%, whereas hybrid validity remained 100% with a 30% fallback rate. At 50% faults, cloud-only validity decreased to 50%, while hybrid again maintained 100% valid responses. Across these conditions, hybrid p95 latency remained between 256.2 and 258.3 ms because the injected HTTP failures were returned after the same 250 ms service delay. The local rule computation itself was negligible compared with the external wait time, with warmed median execution below 0.01 ms in this microbenchmark.

The same pattern appeared for complete hard-failure conditions. When every request returned HTTP 429, the cloud-only path produced no valid recommendation responses, while the hybrid path produced valid fallback responses for all thirty requests. The hybrid p95 latency was 257.8 ms. When every response contained malformed JSON, cloud-only validity again fell to zero, whereas hybrid validity remained 100%; hybrid p95 was 258.5 ms. These cases show that the recovery path is triggered not only by explicit service errors but also by contract violations at the parsing boundary.

Delay injection demonstrates a different trade-off. At a 500 ms provider delay, which remained below the 1,000 ms client timeout, both cloud-only and hybrid configurations achieved 100% valid responses, with p95 latency of 506.4 ms and 506.6 ms, respectively. When the provider delay increased to 1,500 ms, the cloud-only configuration returned no valid responses because all requests crossed the timeout. The hybrid configuration still returned a valid result for every request, but p95 latency increased to 1,005.7 ms because fallback could not begin until the timeout condition had been detected. A 5,000 ms provider delay produced the same availability pattern: 0% valid responses for cloud-only and 100% for hybrid, with hybrid p95 of 1,006.4 ms. The result makes the architectural trade-off explicit: fallback preserves availability, but the timeout budget sets an upper bound on how long a user may wait before degraded service begins.

Fallback coverage was 100% for the detected failure conditions in the benchmark. In the 10%, 30%, and 50% HTTP-500 campaigns, the proportion of hybrid requests entering fallback matched the injected external failure rate. Under all-429, malformed-response, and over-timeout conditions, every hybrid request entered fallback and returned a valid local response. This is stronger evidence of fault containment than the presence of an exception handler alone because it connects injected failures to observable service outcomes.

Table 1. Valid-response rate and latency under injected HTTP 500 faults

Fault rate	Cloud-only VRR	Hybrid VRR	Hybrid fallback	Cloud p95 (ms)	Hybrid p95 (ms)
0%	100%	100%	0%	262.2	258.3
10%	90%	100%	10%	262.6	258.3
30%	70%	100%	30%	257.3	256.2
50%	50%	100%	50%	256.8	256.3

The context-window sensitivity analysis also clarifies the ten-day design choice. The seven-day context occupied 1,252 bytes in the benchmark payload. Extending the history to ten days increased the payload to 1,696 bytes, a 35.5% increase over seven days. Fourteen days required 2,295 bytes, while thirty days required 4,677 bytes, approximately 2.76 times the ten-day payload. Mean serialization time remained very small in all cases, rising from 5.4 microseconds at seven days to 20.3 microseconds at thirty days. Thus, serialization cost is not a practical constraint at these sizes; the more relevant effects of a longer window are the larger prompt, greater transmission of behavioural history, and additional material made available to the external model.

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

The ten-day window is therefore supported as a bounded design compromise rather than an optimum. It retains one complete seven-day cycle plus three recent days while keeping the transmitted context substantially smaller than a thirty-day window. The benchmark does not show that ten days produces better behavioural advice than seven or fourteen days. It shows that the design has an explicit and measurable information budget, which can be altered and evaluated without changing the rest of the architecture.

The rule-engine formalisation also resolves ambiguity about degraded operation. For a fixed context, the same activated predicates, priorities, action identifiers, and templates produce the same ordered recommendations. The cloud path remains free to vary in wording and synthesis, but the minimum service contract does not depend on such variability. The client can additionally expose the execution mode, allowing a user interface to state whether a recommendation was generated through cloud AI or local fallback. This makes degradation visible rather than silently presenting two qualitatively different mechanisms as identical.

Finally, the revised prompt boundary reduces unnecessary disclosure. Only the allow-listed behavioural variables, bounded history, and selected goals need to leave the application process for cloud generation. Direct account identifiers, database keys, session data, and free-text notes are outside the projected cloud context. This architectural separation does not eliminate all privacy risks, but it converts data minimisation from a general policy statement into an enforceable software boundary.

Table 2. Recovery under rate-limit, malformed-response, and timeout conditions

Injected condition	Cloud-only VRR	Hybrid VRR	Hybrid fallback	Hybrid p95 (ms)
HTTP 429; 250 ms	0%	100%	100%	257.8
Malformed JSON; 250 ms	0%	100%	100%	258.5
Provider delay 500 ms	100%	100%	0%	506.6
Provider delay 1,500 ms	0%	100%	100%	1,005.7
Provider delay 5,000 ms	0%	100%	100%	1,006.4

5. DISCUSSION

A try-catch statement is not novel by itself. The contribution is the contract built around it: cloud AI is non-essential; both paths share bounded inputs and an output contract; degraded execution is observable; local rules are formalised; domain integrity remains local; and cloud prompts pass through a privacy-oriented projection. The fault-injection results show that this contract changes observable service behaviour under external failure.

The availability results are straightforward but important. For injected HTTP failures, the cloud-only valid-response rate decreased in direct proportion to the external fault rate, whereas the hybrid configuration remained at 100% valid responses. Under all-429 and malformed-response conditions, the hybrid path also contained every detected fault. This confirms the architectural claim behind RQ1: an external model failure does not have to become a recommendation-service failure when a deterministic local path is part of the service contract.

Latency qualifies the availability result. Fast failures trigger fallback with little additional cost, whereas slow failures cannot degrade until a timeout is detected. With a one-second timeout, the hybrid path preserved 100% valid responses under 1.5 s and 5 s delays, but p95 latency moved to approximately one second. The timeout is therefore both a reliability and user-experience parameter: shorter values accelerate fallback, whereas longer values preserve more opportunities for cloud generation. Adaptive timeouts or a circuit breaker should be evaluated in future versions.

RQ2 concerned the quality of the fallback specification. Formalising the rule engine as predicates, priorities, action families, and templates creates a clearer basis for verification than an opaque Rule(C) function. The seven rule families mirror the behavioural variables already captured by Hypnara. Importantly, deterministic fallback is not presented as equivalent to the LLM. Its value is that the system can guarantee a minimum level of bounded, inspectable advice. The LLM remains useful for language variation, synthesis across signals, and conversational flexibility. The architecture therefore treats local rules and generative AI as complementary components with different assurance properties.

The context analysis does not claim that ten days is behaviourally optimal. Computational cost was trivial across all tested windows, but the amount of behavioural information exposed to the provider increased with history length. Ten days is therefore a bounded implementation budget: one full weekly cycle plus recent continuation, with substantially less payload than thirty days. Recommendation relevance across seven, ten, and fourteen days remains an HCI question for participant research.

The privacy projection is similarly modest but necessary. Pseudonymisation is often discussed at the database level, yet LLM integration creates a second data boundary at prompt construction. Excluding identifiers and unnecessary free text before transmission reduces exposure and limits accidental schema leakage. This does not make the cloud path anonymous in a strict legal sense, because behavioural sequences themselves can sometimes be identifying. It does, however, satisfy a stronger engineering principle than sending raw application objects to the provider. Production deployment should additionally use transport encryption, secure credential handling, explicit retention policies, access control, and provider-specific data-processing terms.

Temporal correctness and semantic missing-data handling remain secondary but generalisable findings. Daily self-tracking systems operate on civil dates, not only timestamps. Treating UTC as the user's calendar can corrupt streaks, reminders, and due-date logic even when database timestamps are technically valid. Likewise, mapping an absent observation to a low score can create a false negative state. These issues illustrate why AI resilience should not distract from ordinary software correctness. A dependable AI-enabled application requires both: the external intelligence path must fail safely, and the local domain model must remain semantically coherent.

SJIF Impact Factor (2026): 8.863| ISI I.F. Value: 1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 11 | Issue: 10| October 2026 - Peer Reviewed Journal

The limitations are clear. The provider emulator isolates status, parsing, and timeout behaviour, so its latency figures are not measurements of DeepSeek production service. The campaign is not a large-scale load test and does not cover database contention or multi-instance deployment. Local rules were evaluated as software logic, not clinically validated recommendations. Finally, no participant study assessed preference, trust, or engagement when the application switches between cloud and fallback modes.

In conclusion, Hypnara demonstrates a resilient hybrid-AI pattern for personalised digital wellness in which cloud generation enriches the service but does not define its minimum availability. The controlled fault-injection benchmark provides quantitative evidence that the hybrid configuration preserves valid recommendation responses under injected HTTP failures, malformed responses, rate limits, and delays beyond the timeout boundary. The rule engine, bounded context policy, privacy projection, local temporal logic, and behavioural integrity constraints make the architecture more than a single exception handler: together they define what the system must continue to do when AI is degraded. The central engineering implication is that personalisation and resilience should be designed together. An AI-enabled wellness application can benefit from generative language while still preserving a transparent, deterministic, and locally enforceable service when the external model is unavailable.

REFERENCES

1. *Almoallim, S., & Sas, C. (2022). Toward research-informed design implications for interventions limiting smartphone use: Functionalities review of digital well-being apps. JMIR Formative Research, 6(4), e31730. https://doi.org/10.2196/31730*
2. *Arroyo, A. C., & Zawadzki, M. J. (2022). The implementation of behavior change techniques in mHealth apps for sleep: Systematic review. JMIR mHealth and uHealth, 10(4), e33527. https://doi.org/10.2196/33527*
3. *Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 1(1), 11-33. https://doi.org/10.1109/TDSC.2004.2*
4. *Chu, Y., Oh, Y., Gwon, M., Hwang, S., Jeong, H., Kim, H.-W., Kim, K., & Kim, Y. H. (2023). Dose-response analysis of smartphone usage and self-reported sleep quality: A systematic review and meta-analysis of observational studies. Journal of Clinical Sleep Medicine, 19(3), 621-630. https://doi.org/10.5664/jcsm.10392*
5. *Fogg, B. J. (2009). A behavior model for persuasive design. In Proceedings of the 4th International Conference on Persuasive Technology (Persuasive '09) (Article 40, pp. 1-7). Association for Computing Machinery. https://doi.org/10.1145/1541948.1541999*
6. *Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75-105. https://doi.org/10.2307/25148625*
7. *Hsueh, M.-C., Tsai, T. K., & Iyer, R. K. (1997). Fault injection techniques and tools. Computer, 30(4), 75-82. https://doi.org/10.1109/2.585157*
8. *Monge Roffarello, A., & De Russis, L. (2024). Hey StepByStep! Can you teach me how to use my phone better? International Journal of Human-Computer Studies, 183, 103195. https://doi.org/10.1016/j.ijhcs.2023.103195*
9. *Nahum-Shani, I., Smith, S. N., Spring, B. J., Collins, L. M., Witkiewitz, K., Tewari, A., & Murphy, S. A. (2018). Just-in-time adaptive interventions (JITAIs) in mobile health: Key components and design principles for ongoing health behavior support. Annals of Behavioral Medicine, 52(6), 446-462. https://doi.org/10.1007/s12160-016-9830-8*
10. *Natella, R., Cotroneo, D., & Madeira, H. S. (2016). Assessing dependability with software fault injection: A survey. ACM Computing Surveys, 48(3), Article 44, 1-55. https://doi.org/10.1145/2841425*
11. *Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management Information Systems, 24(3), 45-77. https://doi.org/10.2753/MIS0742-1222240302*
12. *Sáez, C., Ferri, P., & García-Gómez, J. M. (2024). Resilient artificial intelligence in health: Synthesis and research agenda toward next-generation trustworthy clinical decision support. Journal of Medical Internet Research, 26, e50295. https://doi.org/10.2196/50295*
13. *Smits, M., Kim, C. M., van Goor, H., & Ludden, G. D. S. (2022). From digital health to digital well-being: Systematic scoping review. Journal of Medical Internet Research, 24(4), e33787. https://doi.org/10.2196/33787*
14. *Taimoor, N., & Rehman, S. (2022). Reliable and resilient AI and IoT-based personalised healthcare services: A survey. IEEE Access, 10, 535-563. https://doi.org/10.1109/ACCESS.2021.3137364*
15. *Wang, L., Wan, Z., Ni, C., Song, Q., Li, Y., Clayton, E., Malin, B., & Yin, Z. (2024). Applications and concerns of ChatGPT and other conversational large language models in health care: Systematic review. Journal of Medical Internet Research, 26, e22769. https://doi.org/10.2196/22769*
16. *Watson, N. F., Badr, M. S., Belenky, G., Bliwise, D. L., Buxton, O. M., Buysse, D., Dinges, D. F., Gangwisch, J., Grandner, M. A., Kushida, C., Malhotra, R. K., Martin, J. L., Patel, S. R., Quan, S. F., & Tasali, E. (2015). Recommended amount of sleep for a healthy adult: A joint consensus statement of the American Academy of Sleep Medicine and Sleep Research Society. Sleep, 38(6), 843-844. https://doi.org/10.5665/sleep.4716*