

Claude development & evaluation plan

Yamma · Development test cases

Product context

Yamma is an unreleased Android personal health-record prototype. Current implementation covers manual symptom entry, user confirmation, encrypted on-device storage and a diary. Visit-summary, medicine and appointment workflows are planned.

Claude is proposed for development support and a separate evaluation workflow. This document is a plan; it contains no measured model results.

How we plan to use Claude

Workstream	Intended use	Review before acceptance
Software development	Coding, schema review, tests and documentation	Developer review and repository checks
Structured extraction	Convert authored symptom notes into a record schema	Validate fields against source facts
Factual visit-summary drafts	Organize supplied records into a reviewable draft	Verify every factual statement

Development test cases are authored scenarios, not records from actual users. An isolated workflow will use these cases without real health records, identifiers or application secrets. Outputs remain test artifacts; API responses do not automatically become saved user records.

A bounded evaluation set

Prepare 100 cases with reference facts, uncertainty labels and acceptable outputs. Use 20 in each group: single or multiple symptoms; occurrence versus recording date; missing or uncertain information; negation, corrections or conflicts; irrelevant, malformed or out-of-scope input.

Keep prompt-development examples separate. Run each case three times for each workflow: extraction and summary drafting. This gives **600 planned requests per model/prompt configuration**, including recorded failures and refusals.

Output rules

- Separate occurrence and recording dates. Resolve relative dates only with a supplied reference date.
- Preserve unknown values. Missing intensity is not zero; uncertainty is not confirmation.
- Link fields and claims to source passages or record IDs. Preserve negation and conflicts.
- Add no diagnoses, triage decisions, medication changes or facts absent from the input.

Evaluate structured outputs as a formatting aid, alongside application validation and factual review. Refusals, truncated output and API errors must produce a visible review or failure state.

Measurement & acceptance

Proposed criteria · Results to be published after execution

These are proposed development gates, not achieved scores or clinical standards. Publish counts, denominators and failures.

Measure	Definition	Proposed gate
Schema validity	Responses passing schema and application rules / all planned requests	≥99%; include errors
Factual consistency	Source-supported claims / all factual claims	Zero unsupported claims or critical errors observed
Missing information	Omitted required facts / all required facts across planned attempts	≤2%; zero critical omissions observed
Unknown preservation	Correctly retained unknown/conflicting fields / all such reference fields	100% on this set
Error handling	Correctly blocked/review-routed failures / 20 separate simulated scenarios	20/20; no silent save/success
Latency and cost	End-to-end time and billed usage, including retries	Median/p95 time; cost per attempt and accepted output

Report by workflow, case group and run. Failed attempts count as omitted required facts. Report **N/A for zero denominators** and list responses without claims separately. Critical errors include changed dates, medicines or negation. Zero observed errors does not establish clinical reliability.

Review and decision

A human reviewer compares outputs with reference facts using a written rubric; a second reviewer resolves disputes. Unsupported claims, critical errors or unsafe failure paths block acceptance. Correct and re-evaluate, retaining earlier reports. Passing permits the next development step, not clinical validation or release authorization.

Execution and budget assumptions

Use a separate 20-request pilot to check logging and estimate usage. Assuming 1,000 input and 300 output tokens per request, the 600-request evaluation would use approximately **600,000 input and 180,000 output tokens**, before retries or additional billable usage. This is an estimate.

Estimated cost = input tokens × input rate per million / 1,000,000 + output tokens × output rate per million / 1,000,000 + other billable charges. Use published rates at execution time and set a spending cap. Record actual usage, retries, model/settings, prompt/schema/case versions and execution date. Track development spending separately.

After execution, publish case definitions, methods, failures, limitations and the next decision. API credits and technical guidance would support this work. Model and operating-architecture choices remain separate decisions.

Official documentation consulted

These references inform the proposed method; they do not endorse Yamma or validate its outputs. Checked 10 October 2026.

Anthropic, Structured outputs: <https://platform.claude.com/docs/en/build-with-claude/structured-outputs>
Anthropic, Define success criteria and build evaluations: <https://platform.claude.com/docs/en/test-and-evaluate/develop-tests>
Anthropic, Pricing: <https://platform.claude.com/docs/en/about-claude/pricing>