**Survey instrument**

**Evaluation of Amazon Q Generated Documentation (SARSB)**

**Title (form):** Evaluation of Amazon Q Automated Documentation — SARSB Project (ANA)

**Short description (intro text visible to respondent):**  
You are invited to participate in a research survey that evaluates automatically generated documentation produced by Amazon Q for Java methods in the SARSB system. The survey measures three dimensions — **Completeness**, **Helpfulness**, and **Truthfulness** (CHT) — and should take about **10–25 minutes** depending on the number of methods assigned for review. Participation is voluntary and responses will be used for academic research only.

**CONSENT (First page)**

**Please read and accept before continuing**

**Purpose:** Evaluate the quality of documentation generated automatically by Amazon Q for Java methods of the SARSB system. Results will be used for academic research in software engineering.

**Confidentiality:** Your responses will be anonymized. No code or confidential data will be published.

**Voluntary:** Participation is voluntary and you may withdraw at any time.

**Contact:** murillo.ed1402@gmail.com (Murillo Carvalho, Univ. of Brasília).

**Consent question (required):**

I have read the information and I consent to participate. ☐ Yes ☐ No (If No → end survey)

**SECTION A — Participant profile (one page)**

1. **Participant ID** (text) — use a unique code (e.g., ANA\_DEV\_01).
2. **Role** (single choice): Analyst / Developer / Tester / Architect / Other (text).
3. **Years of Java experience** (single choice): <2 / 2–5 / 5–10 / >10.
4. **Familiarity with SARSB** (single choice): None / Low / Medium / High.
5. **Have you used AI tools for code or documentation before?** (single choice) Yes / No.

If **Yes**: *Which tools?* (text)

1. **Average daily time spent maintaining code (hours)** (number).

**ADMINISTRATION INSTRUCTIONS (internal, not shown to respondents)**

**Assignment:** Each participant evaluates up to **10 methods**. Methods are sampled using *proportional stratified sampling* across layers (Config, Model, Repository, Resource, Service, Util, etc.). Randomly assign methods to participants ensuring coverage across layers.

**Block repetition:** The next block (Section B) repeats for each assigned method. Limit total methods per participant to 8–10 to avoid fatigue.

**File attachments:** Provide the code snippet (method signature + body), original comments (if any), and the generated documentation (Javadoc and/or Markdown) to the participant in the form itself (or via a link per method). Each method block must include those elements.

**SECTION B — Method-level evaluation (repeat per assigned method)**

**Method metadata (display only):**

Method ID: M\_METHODID

File path: br.gov.ana.sarsb/...

Layer: Service/Repository/...

Method signature: (show signature)

Method body: (show code snippet)

Generated documentation: (show Javadoc text and Markdown output)

**Instructions for this block:** For each presented method, evaluate the generated documentation according to the items below. Use the 5-point Likert scale where applicable (1 = Strongly disagree / Very poor, 5 = Strongly agree / Excellent). Answer honestly and refer to the code snippet while judging.

**A — COMPLETENESS (structural checks)**

Use Yes/No where indicated or Likert where specified.

B1. **Does the documentation include a clear one-line summary of the method purpose?** (Likert 1–5)  
B2. **Are all method parameters documented in @param entries (if any)?** (Options: Yes / Partially / No / Not applicable)  
B3. **Is the @return documented correctly when the method returns a value?** (Yes / Partially / No / Not applicable)  
B4. **Are thrown exceptions documented in @throws when the code declares or may throw exceptions?** (Yes / Partially / No / Not applicable)  
B5. **Is the overall Javadoc structure and required sections present (summary, params, returns, throws, examples if relevant)?** (Likert 1–5)  
B6. **Completeness quick score (for your internal use):** Indicate the number of required fields present (integer) — optional, form can compute automatically.

**Optional short comment (text):** Completeness remarks.

**B — HELPFULNESS (practical usability)**

C1. **The documentation is clear and easy to read.** (Likert 1–5)  
C2. **The documentation helps me understand the method’s purpose quickly.** (Likert 1–5)  
C3. **The documentation provides enough context to use or maintain the method.** (Likert 1–5)  
C4. **The language and conventions match project standards (tone, names, code examples).** (Likert 1–5)  
C5. **Overall helpfulness score (subjective 1–5).** (Likert 1–5)

**Optional short comment (text):** Helpfulness remarks.

**C — TRUTHFULNESS (factual consistency)**

D1. **All named entities (variables, classes, exceptions, referenced methods) exist in the presented code.** (Yes / Partially / No)  
D2. **The documentation does not state behaviors that do not appear in the code (no hallucinations).** (Likert 1–5)  
D3. **References to external APIs/libraries are accurate (if present).** (Yes / No / Not applicable)  
D4. **Overall factual consistency (Existence Ratio estimate between 0 and 1):** (numeric 0.00–1.00) — optional if you prefer to rate.

**Optional short comment (text):** Truthfulness remarks.

**D — OVERALL & OPERATIONAL QUESTIONS**

E1. **Overall quality of the generated documentation for this method** (single choice / Likert 1–5).  
E2. **Would you accept this documentation into the main branch without edits?** (Yes / No / Only minor edits)  
E3. **Time spent reviewing this documentation (minutes)** (number).  
E4. **Suggested action:** (single choice) Accept as-is / Accept with minor edits / Reject (regenerate) / Manual rewrite required.  
E5. **Free text: Suggest improvements or note any critical issues** (paragraph).

**SECTION C — Aggregate & Perception (final pages)**

1. **Across all methods you reviewed, how consistent was the model in delivering required Javadoc sections?** (Likert 1–5)
2. **Did few-shot prompts result in better outputs than zero-shot (if both were shown)?** (Yes / No / Not sure / Not applicable)
3. **Which layer types produced the most reliable documentation?** (multi-select: Config / Model / Repository / Resource / Service / Util / Other)
4. **Would you recommend integrating Amazon Q into the CI pipeline for auto-documentation (with gates)?** (Yes / No / Only for specific layers)
5. **What acceptance threshold would you set for automatic merge without mandatory human review?** (numeric % — e.g., 90%)
6. **General comments on trust, governance and training needed for adoption** (paragraph).

**SECTION D — Demographics & closing (optional)**

Organization / Team (text) — optional.

Would you be available for follow-up (email)? (Yes / No) — if Yes, provide contact (text).

Final remarks (text).

**Scoring rules & mapping to research metrics (for data analysis)**

**Goal:** compute per-method and aggregated C-score, H-score, T-score, QDI.

**C-score (Completeness)** — suggested automated computation:  
C = (# required Javadoc sections present) / ( # required sections applicable )  
*required sections* typically = {summary, @param (if params exist), @return (if return type not void), @throws (if exceptions present)}.  
Score range: 0–1. Multiply by 100 for percentage.

**H-score (Helpfulness)** — average of Likert items under Helpfulness (C1–C5).  
H = mean(C1..C5) mapped to 0–1 scale: (mean - 1) / 4.

**T-score (Truthfulness)** — use a combination of binary/likert and existence ratio:  
Option A: T = (Existence\_Ratio + (D2 mapped to 0–1))/2  
where Existence\_Ratio = manually reported (0–1) or computed as (# entities that exist) / (# entities mentioned).  
Option B: If no numeric existence ratio, map D1 and D2 to 0–1 and average.

**QDI (Quality Deviation Index)** — measures gain relative to original documentation:  
QDI = (C\_generated - C\_original) / max( C\_original, ε )  
where C\_generated and C\_original are C-scores (0–1), ε small constant to avoid divide-by-zero. Values >0 indicate an improvement.

**Per-participant & aggregated analysis:**

Compute method-level C/H/T and then compute means per layer and for the entire dataset.

Compute inter-rater reliability if multiple reviewers per method exist (Fleiss’ Kappa for categorical, ICC for continuous).

Perform correlation analysis (Pearson / Spearman) between code properties (size, cyclomatic complexity) and C/H/T.

**Data export & suggested CSV schema (columns)**

Each reviewed method should map to one CSV row. Suggested column names:

* responder\_id
* responder\_role
* responder\_experience\_years
* method\_id
* method\_layer
* method\_file\_path
* method\_signature
* generated\_doc\_text (or a link to file)
* completeness\_summary (Likert 1–5)
* completeness\_params (Yes/Partially/No/NA)
* completeness\_return (Yes/Partially/No/NA)
* completeness\_throws (Yes/Partially/No/NA)
* completeness\_structure\_score (numeric 0–1)
* helpfulness\_c1 … helpfulness\_c5 (1–5)
* helpfulness\_mean (1–5)
* truth\_d1\_exists (Yes/Partially/No)
* truth\_d2\_hallucination\_likert (1–5)
* truth\_existence\_ratio (0–1 numeric)
* overall\_quality (1–5)
* accept\_without\_edits (Yes/No/MinorEdits)
* time\_spent\_min (number)
* suggested\_action (Accept / Minor edits / Reject / Rewrite)
* free\_text\_comments
* timestamp

**Deployment instructions (quick)**

1. **Create the form** in Google Forms / Qualtrics / LimeSurvey.
2. **Create one repeating block** (Section B) with variables for method fields and the UIs described. Use “Page per method” or repeatable blocks.
3. **Attach code snippets and generated docs** inline or provide links to files stored in the repo (preferably anonymized).
4. **Pilot test** with 3–5 internal participants; confirm that assignment, time per method, and question wording are clear.
5. **Deploy** to sample of participants; keep per-respondent method count between 5–10 to avoid fatigue.
6. **Export CSV** and verify column mapping matches the suggested schema.

**Sampling and required sample size (recommendation)**

If population of reviewers ~ N (internal devs), aim for at least **30–50 reviewers** or **≥3 independent reviewers per method** for inter-rater reliability.

For per-method averages, **3 reviewers per method** is a pragmatic target.

For hypothesis tests (e.g., difference in C-score between layers), compute power analysis based on expected effect size; as a pragmatic default, sample ~200 method-reviews (e.g., 40 reviewers × 5 methods each) to allow moderate-power comparisons.

**Ethics & privacy checklist**

Do not include production-sensitive code or credentials.

Mask or remove any personal data from code comments.

Keep responses anonymized; store only participant IDs.

Ensure that use of Amazon Q / outputs complies with ANA data governance and AWS agreements.

**Optional: Example Google Forms layout (quick copy)**

Page 1: Consent & Participant profile.

Page 2+: Method block repeated (1 page per method).

Final page: Aggregate perception & closing questions.

Use “Required” on critical items (consent, overall quality, accept action).

For multi-method assignment, create a spreadsheet input that pre-loads assigned method details (Qualtrics can pipe-in fields per panel).