**🧠 LAB 4 — AI PROMPT DESIGN FOR RELIABILITY CALCULATION**

**📊 Context**

This lab demonstrates how Generative AI (GAI) can generate synthetic repository data and calculate a **Repository Reliability Index (RRI)** from four key process metrics extracted from GitHub events.  
The RRI is calculated as the **average of normalized metric scores**, representing a simplified AI reasoning approach.

I designed structured prompts to simulate repository performance metrics (X variables) and compute a Repository Reliability Index (Y). The AI model applied a simple averaging formula to quantify repository reliability and assigned qualitative labels. This mirrors how predictive reliability scoring could work in a real system where data is collected automatically from GitHub API events.

There are 3 prompt sections, 1st one represents a zero shot approach, 2nd one represents a few shot approach, 3rd one represents a more detailed approach.

Contents

[**📊 Context** 1](#_Toc212797176)

[**Section 1: Zero Shot Approach** 2](#_Toc212797177)

[**Goal: Give the AI a set of new observations `X` and ask it to compute `RRI` and a reliability label (no examples provided).** 2](#_Toc212797178)

[**🧮 Results Computed by AI (Y Output)** 4](#_Toc212797179)

[**Section 2: Few Shot Approach** 5](#_Toc212797180)

[**Goal: Provide one known (X, Y) pair to the AI, then ask it to apply the same mapping to new inputs.** 5](#_Toc212797181)

[**🧮 Results Computed by AI (Y Output)** 7](#_Toc212797182)

[**Section 3: Detailed Approach** 8](#_Toc212797183)

[**⚙️ Step 1 — Prompt: Generate Mock Metrics (X Variables)** 8](#_Toc212797184)

[**🧮 Step 2 — Prompt: Calculate Reliability Index (Y Output)** 8](#_Toc212797185)

[**📈 Step 3 — Prompt: Summarize Insights** 9](#_Toc212797186)

[**🧠 Step 4 — Prompt: Final AI-driven Interpretation** 9](#_Toc212797187)

**Section 1: Zero Shot Approach**

**Goal: Give the AI a set of new observations `X` and ask it to compute `RRI` and a reliability label (no examples provided).**

**Prompt:**

You are an AI agent for repository reliability assessment.  
Calculate the Repository Reliability Index (RRI) and reliability classification for each repository using the formula:

Use the following reliability scale:

* 0–40 → Poor
* 41–60 → Moderate
* 61–80 → Good
* 81–100 → Excellent

Return results both as a **table** and as **JSON** with fields: `repository`, `RRI` (rounded to 1 decimal), and `reliability\_label`.

Here is the input data:

[

{

"repository": "repo\_1",

"review\_rigor\_score": 45.6,

"pr\_merge\_ratio": 96.1,

"contributor\_diversity\_index": 97.9,

"issue\_resolution\_rate": 86.4

},

{

"repository": "repo\_2",

"review\_rigor\_score": 28.7,

"pr\_merge\_ratio": 49.3,

"contributor\_diversity\_index": 10.5,

"issue\_resolution\_rate": 44.7

},

{

"repository": "repo\_3",

"review\_rigor\_score": 59.1,

"pr\_merge\_ratio": 41.1,

"contributor\_diversity\_index": 28.9,

"issue\_resolution\_rate": 39.2

},

{

"repository": "repo\_4",

"review\_rigor\_score": 58.7,

"pr\_merge\_ratio": 44.8,

"contributor\_diversity\_index": 44.2,

"issue\_resolution\_rate": 42.4

},

{

"repository": "repo\_5",

"review\_rigor\_score": 36.2,

"pr\_merge\_ratio": 41.0,

"contributor\_diversity\_index": 74.5,

"issue\_resolution\_rate": 33.0

}

]

**🧮 Results Computed by AI (Y Output)**

**Table Output:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Repository** | **Review Rigor** | **PR Merge** | **Diversity** | **Issue Resolution** | **RRI** | **Reliability** |
| repo\_1 | 45.6 | 96.1 | 97.9 | 86.4 | **81.5** | **Excellent** |
| repo\_2 | 28.7 | 49.3 | 10.5 | 44.7 | **33.3** | **Poor** |
| repo\_3 | 59.1 | 41.1 | 28.9 | 39.2 | **42.1** | **Moderate** |
| repo\_4 | 58.7 | 44.8 | 44.2 | 42.4 | **47.5** | **Moderate** |
| repo\_5 | 36.2 | 41.0 | 74.5 | 33.0 | **46.2** | **Moderate** |

**JSON Output:**

[

{"repository": "repo\_1", "RRI": 81.5, "reliability\_label": "Excellent"},

{"repository": "repo\_2", "RRI": 33.3, "reliability\_label": "Poor"},

{"repository": "repo\_3", "RRI": 42.1, "reliability\_label": "Moderate"},

{"repository": "repo\_4", "RRI": 47.5, "reliability\_label": "Moderate"},

{"repository": "repo\_5", "RRI": 46.2, "reliability\_label": "Moderate"}

]

**Section 2: Few Shot Approach**

**Goal: Provide one known (X, Y) pair to the AI, then ask it to apply the same mapping to new inputs.**

**Prompt:**

You are an AI agent trained to assess repository reliability.  
Learn from the example below showing how input metrics map to an RRI and reliability classification. Then apply the same logic to the new repositories provided.

**Formula:**

**Scale:**

* 0–40 → Poor
* 41–60 → Moderate
* 61–80 → Good
* 81–100 → Excellent

**Known Example (X → Y):**

* Review Rigor Score: 76.7
* PR Merge Ratio: 68.4
* Contributor Diversity Index: 42.1
* Issue Resolution Rate: 95.4
* RRI: 70.6 → Good

**New Input Data:**

[

{

"repository": "repo\_A",

"review\_rigor\_score": 82.3,

"pr\_merge\_ratio": 90.4,

"contributor\_diversity\_index": 87.9,

"issue\_resolution\_rate": 91.0

},

{

"repository": "repo\_B",

"review\_rigor\_score": 61.1,

"pr\_merge\_ratio": 72.5,

"contributor\_diversity\_index": 69.3,

"issue\_resolution\_rate": 58.8

},

{

"repository": "repo\_C",

"review\_rigor\_score": 48.2,

"pr\_merge\_ratio": 51.4,

"contributor\_diversity\_index": 55.0,

"issue\_resolution\_rate": 45.9

}

]

Return results both as a **table** and as **JSON** with fields: `repository`, `RRI` (rounded to 1 decimal), and `reliability\_label`.

**🧮 Results Computed by AI (Y Output)**

**Table Output:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Repository** | **Review Rigor** | **PR Merge** | **Diversity** | **Issue Resolution** | **RRI** | **Reliability** |
| repo\_A | 82.3 | 90.4 | 87.9 | 91.0 | **87.9** | **Excellent** |
| repo\_B | 61.1 | 72.5 | 69.3 | 58.8 | **65.4** | **Good** |
| repo\_C | 48.2 | 51.4 | 55.0 | 45.9 | **50.1** | **Moderate** |

**JSON Output:**

[

{"repository": "repo\_A", "RRI": 87.9, "reliability\_label": "Excellent"},

{"repository": "repo\_B", "RRI": 65.4, "reliability\_label": "Good"},

{"repository": "repo\_C", "RRI": 50.1, "reliability\_label": "Moderate"}

]

# **Section 3: Detailed Approach**

**⚙️** **Step 1 — Prompt: Generate Mock Metrics (X Variables)**

**Prompt:**

You are an AI assistant analyzing software repository reliability.  
Generate mock values for the following four metrics for 5 GitHub repositories.  
Each metric is on a scale from 0 to 100, where higher is better.  
Metrics:

* review\_rigor\_score (depth and frequency of code reviews)
* pr\_merge\_ratio (percentage of successfully merged pull requests)
* contributor\_diversity\_index (number and activity diversity of contributors)
* issue\_resolution\_rate (speed and efficiency of resolving issues)

Provide results as a JSON array with 5 repositories showing mixed performance levels.

**🧮 Step 2 — Prompt: Calculate Reliability Index (Y Output)**

**Prompt:**

Based on the following repository metrics, calculate the *Repository Reliability Index (RRI)* for each repository.

Formula:

Then classify the reliability based on this scale:

* 0–40 → **Poor**
* 41–60 → **Moderate**
* 61–80 → **Good**
* 81–100 → **Excellent**

Return an Excel and JSON output with each repository’s calculated RRI (rounded to 1 decimal) and reliability label.

**📈 Step 3 — Prompt: Summarize Insights**

**Prompt:**

Summarize insights from the computed RRI values.  
Identify which repositories are highly reliable, which are risky, and provide one short suggestion per repository on how to improve reliability (e.g., increase review activity, resolve issues faster, etc.).

**🧠 Step 4 — Prompt: Final AI-driven Interpretation**

**Prompt:**

Based on the RRI results, generate a short paragraph (4–5 sentences) that an AI analyst could include in a reliability assessment report.  
Mention which repositories show strong software process maturity and which indicate risk areas.