1 LLM Prompts

This section outlines the various prompting techniques employed in the study.

Here, *TitleOfTheArticle* of the article is a string variable and is updated each time based on the actual title of the article. The highlighted portion (italic) in the prompt indicates the emotional stimuli. Likewise, this emotional stimuli text was added at the bottom of each prompt.

The employed prompts are as follows:

1.1 Zero Shot

You have been entrusted with the complete article titled *TitleOfTheArticle* Your objective is to meticulously extract actionable recommendations from this article. These recommendations should be aimed at enhancing the sustainability of open-source software projects.

Your attention to detail in this task is crucial, as its successful completion holds significant importance for my career advancement.

1.2 Chain-of-Thought

You have been given the full text of the article titled *TitleOfTheArticle*. Your task is to extract actionable recommendations from the article. An actionable recommendation is a practical, evidence-based suggestion that provides specific, clear steps or instructions which, when implemented, are expected to produce tangible and positive results. These recommendations should be practical, making them easy to implement in real-world scenarios, and evidence-based, supported by data or findings from the article. They must be specific, providing clearly defined steps or instructions, and result-oriented, aimed at producing tangible and positive outcomes. These recommendations should offer concrete guidance that can be directly put into practice to achieve desired results. Adopting these actionable recommendations can help make open-source software projects more sustainable.

Let's break down the problem into the following steps:

Step 1: Carefully read each sentence of the article and identify recommendations or suggestions. Look for imperative sentences or phrases that give commands, make requests, or offer instructions to direct or persuade someone to perform a specific action.

Step 2: Extract the practical suggestions from the identified sentences and present them as concise, unambiguous statements in a list format.

Step 3: Review the list of recommendations and remove any duplicate items to ensure clarity and avoid redundancy.

Step 4: For each recommendation in the final list, assign a confidence level on a scale of 0 to 1, indicating how confident you are in the effectiveness of the recommendation for making open-source projects sustainable. Provide a brief explanation for your confidence level.

Your attention to detail in this task is crucial, as its successful completion holds significant importance for my career advancement.

1.3 Reason + Action

Thought 1: I have been given the article "On the abandonment and survival of open source projects: An empirical investigation." My task is to extract actionable recommendations from the article. An actionable recommendation is a practical, evidence-based suggestion that provides specific, clear steps or instructions which, when implemented, are expected to produce tangible and positive results. These recommendations should be practical, making them easy to implement in real-world scenarios, and evidence-based, supported by data or findings from the article. They must be specific, providing clearly defined steps or instructions, and result-oriented, aimed at producing tangible and positive outcomes. These recommendations should offer concrete guidance that can be directly put into practice to achieve desired results. Adopting these actionable recommendations can help make open-source software projects more sustainable. Action 1: Carefully read each sentence of the article, focusing on identifying imperative sentences or phrases that give commands, make requests, or offer instructions to direct or persuade someone to perform a specific action. Observation 1: A list of potential actionable recommendations has been identified from the article. Thought 2: I need to extract the practical suggestions from the identified sentences and present them as concise, unambiguous statements. Action 2: Convert the identified sentences into clear, actionable statements and compile them into a list. Observation 2: A list of actionable recommendations has been compiled from the article. Thought 3: To ensure clarity and avoid redundancy, I should review the list and remove any duplicate items. Action 3: Review the list of recommendations and remove any duplicate or highly similar items. Observation 3: The list of recommendations has been refined, removing duplicates and ensuring clarity. Thought 4: I need to assess the confidence level for each recommendation to indicate how effective it might be in making open-source projects sustainable. Action 4: For each recommendation in the final list, assign a confidence level on a scale of 0 to 1, indicating how confident I am in the effectiveness of the recommendation for making open-source projects sustainable. Provide a brief explanation for each assigned confidence level. Observation 4: Confidence scores have been assigned to each recommendation in the final list, along with brief explanations.

Your attention to detail in this task is crucial, as its successful completion holds significant importance for my career advancement.

2 Definition of ReACT Categories

2.1 New Contributor Onboarding and Involvement

Definition. This category focuses on ensuring that new contributors can easily join, understand, and meaningfully contribute to the project.

Criteria for Assignment: *a)* Actionable facilitates the integration of new contributors by providing mentorship, onboarding materials, or simplifying the contribution process; *b)* Actionable relates to improving project documentation or offering better support mechanisms for first-time contributors; *c)* Actionable helps build a welcoming, inclusive, and open culture for new participants.

2.2 Code Standards and Maintainability

Definition. This category deals with ensuring that the codebase adheres to established standards, making it easier to maintain and scale. It includes efforts to ensure code readability, modularity, and compliance with coding best practices.

Criteria for Assignment. *a)* Actionable relates to improving the quality, readability, or structure of the codebase; *b)* Actionable includes efforts to enforce coding guidelines, refactor code for better maintainability, or reduce technical debt; Actionable includes the use of linters, formatters, or static code analysis tools.

2.3 Automated Testing and Quality Assurance

Definition. This category focuses on ensuring the project's robustness and reliability through automated testing practices, such as unit, integration, and end-to-end tests. It also includes broader quality assurance activities.

Criteria for Assignment. a) Actionable involves the implementation or improvement of automated testing frameworks and testing strategies; *b)* Actionable includes practices that ensure the detection of bugs early in the development cycle and ensure high-quality releases;

2.4 Community Collaboration and Engagement

Definition. This category deals with activities that foster collaboration, communication, and engagement within the OSS community. It includes practices for keeping the community active and involved.

Criteria for Assignment. a) Actionable aims to improve communication between contributors, maintainers, and users; Actionable involves organizing community-driven events, discussions, or collaborations, as well as platforms to enhance transparency and teamwork; Actionable relates to tools and processes for better community governance and decision-making.

2.5 Documentation Practices

Definition. This category focuses on ensuring that the project's documentation is thorough, up-to-date, and easily accessible. Documentation practices are crucial for both current and future contributors.

Criteria for Assignment. a) Actionable focuses on improving the quality, clarity, or accessibility of project documentation, such as user guides, API references, or contributor guides; *b)* Actionable includes practices for keeping documentation synchronized with the codebase and ensuring it meets the needs of different stakeholders; *c)* Actionable involves translation efforts or making documentation more accessible to non-expert audiences.

2.6 Project Management and Governance

Definition. This category deals with the governance structure and project management practices that keep the project organized, transparent, and sustainable over the long term.

Criteria for Assignment. a) Actionable enhances the governance model, clarifies roles and responsibilities, or improves the decision-making process; *b)* Actionable involves defining or refining processes for issue triaging, release management, or conflict resolution; *c)* Actionable includes efforts to improve the transparency of project goals, progress, and decision-making.

2.7 Security Best Practices and Legal Compliance

Definition. This category addresses efforts to secure the project and ensure compliance with relevant legal standards, such as licenses, data privacy laws, and security protocols.

Criteria for Assignment. *a)* Actionable focuses on improving the security posture of the project by following best practices, addressing vulnerabilities, or conducting audits; *b)* Actionable involves ensuring compliance with open-source licenses, setting up contributor license agreements (CLAs), or aligning with data privacy regulations; *c)* Actionable includes security measures such as dependency management, security audits, and secure coding practices.

2.8 CI/CD and DevOps Automation

Definition. This category deals with continuous integration and continuous deployment (CI/CD) processes that automate building, testing, and deployment pipelines. It also includes broader DevOps automation tasks.

Criteria for Assignment. *a)* Actionable involves the setup or enhancement of CI/CD pipelines to ensure faster, reliable, and automated releases; *b)* Actionable relates to automating infrastructure provisioning, containerization, or deployment to cloud environments; *c)* Actionable includes the integration of DevOps practices that ensure smooth, automated, and repeatable processes for software development, testing, and deployment.

3 Server Configuration

We conducted all the experiments for our study on a dedicated local server with a well-defined configuration to ensure consistency and reliability in our results. The configuration of the server is presented in table 1.

Table 1: Local server configuration

Component	**Configuration**
Primary Memory (RAM)	64 GB
Secondary Memory (Hard Disk)	512 GB SSD
Processor	*Name*: Intel(R) Xeon(R) W-2135 CPU @ 3.70GHz
	Architecture: x86_64
	CPU(s): 12
	Thread(s) per core: 2
	Core(s) per socket: 6
	Max Clock Speed: 4500.00 MHz
	Min Clock Speed: 1200.00 MHz
	PCI Express Lanes: 48
	Integrated Graphics: None
	TDP (Thermal Design Power): 140 W
Cache	*L1d cache*: 192 KiB (6 instances)
	L1i cache: 192 KiB (6 instances)
	L2 cache: 6 MiB (6 instances)
	L3 cache: 8.3 MiB (1 instance)
GPU	*Name*: GV100 [TITAN V]
	Memory: 12 GB HBM2
	Memory Interface Width: 3072 bits
	Memory Bandwidth: 652.8 GB/s
	Base Clock Speed: 1200 MHz
	Boost Clock Speed: 1455 MHz
	Memory Clock Speed: 850 MHz (1700 MHz effective)
	CUDA Cores: 5120
	Tensor Cores: 640
	Thermal Design Power (TDP): 250 W

4 ReACT-Gold Annotation

4.1 Article 1

"*On the abandonment and survival of open source projects: An empirical investigation*" by Avelino et al. [1].

Derived Actionables

i Project maintainers should strive to increase the number of Truck Factor (TF) developers to reduce the risk of project abandonment.

ii Projects should seek alternative backing, such as company-based support, to prevent or reduce the chances of TF developer detachments (TFDDs).

iii Open source communities should foster a friendly and active environment to attract and retain new contributors.

iv Project owners should ensure that their repositories are easily accessible.

v Project maintainers should implement and adhere to well-known software engineering principles and practices to make it easier for new developers to contribute.

vi Open source projects should consider using popular programming languages to attract a wider pool of potential contributors.

vii Project maintainers should be aware of and mitigate common barriers faced by new contributors, particularly the lack of time and experience.

viii Open source communities should promote and share successful cases of projects overcoming TFDDs to motivate developers to actively contribute to projects at risk.

ix Project maintainers should regularly assess the risk of abandonment by TF developers and take proactive measures to ensure project continuity.

x Projects should implement continuous integration practices to facilitate contributions from new developers.

xi Project maintainers should strive to keep the codebase clean and well-designed to make it easier for new contributors to understand and work with the project.

xii Projects should consider implementing a code review process to maintain code quality

4.2 Article 2

"What attracts newcomers to onboard on OSS projects? tl; dr: Popularity" by Felipe et al. [2].

Derived Actionables

i Reduce the time taken to review and merge pull requests

ii Use multiple programming languages in the project iii Maintain the project over a longer period of time to increase its age

iv Have a larger number of integrators (contributors with rights to merge pull requests)

v Choose a popular main programming language for the project.

vi Select an appropriate software application domain for the project.

4.3 Article 3

"Difficulties of newcomers joining software projects already in execution" by Matturro et al. [3].

Derived Actionables

i Assign an experienced team member to coach and guide the newcomer.

ii Follow up with newcomers on both success and failure.

iii Holding training sessions for newcomers.

iv Maintain concise, updated, accessible documentation.

v Grant the newcomer freedom: Encourage and allow them to express opinions, propose changes, and share personal viewpoints to foster a comfortable environment.

vi Establish a personalized integration plan: Outline a gradual assignment of tasks and responsibilities for seamless incorporation.

4.4 Article 4

"Developer turnover in global, industrial open source projects: Insights from applying survival analysis" by Lin et al. [4].

Derived Actionables

i Provide better onboarding assistance for newcomers to help them become more engaged in the project.

ii Ensure developers maintain both code developed by others and their own code.

iii Assign more code maintenance tasks to developers who primarily write new code.

iv Give coding tasks to developers who mainly work on documentation to increase their chances of staying in the project.

v Encourage early contribution to the project, as developers who join earlier tend to stay longer.

vi Implement strategies to manage growing code complexity, as it can form an obstacle for new developers' contributions.

4.5 Article 5

"The role of mentoring and project characteristics for onboarding in open source software projects" by Fagerholm et al. [5].

<u>*Derived Actionables*</u>

i 1. Provide onboarding support and help newcomers to make their first contribution

4.6 Article 6

"The more the merrier? The effect of size of core team subgroups on success of open source projects" by Przybilla et al. [6].

<u>*Derived Actionables*</u>

i 1. Projects should implement mechanisms to motivate and support long-term, consistent participation from key developers.

ii Encourage experienced developers to guide and support newcomers, particularly through issue-related activities like commenting on and resolving issues.

iii Implement strategies to increase the visibility and recognition of high-reputation contributors.

iv Create opportunities for core members to focus on issue-related activities.

v Be cautious about having too many extensively contributing core members. Balance the core team to avoid creating the impression of a closed circle that might deter new contributors.

vi Address common barriers faced by newcomers, such as identifying where to start contributing.

vii Analyze and potentially steer the configuration of subgroups within the project based on factors like reputation, issue focus, contribution extent, and persistence.

viii Consider implementing formal collaborator status or other metrics to make reputation more visible and meaningful to outsiders.

ix Focus on creating a supportive environment that aligns with core OSS values, such as learning opportunities and the ability to make meaningful contributions.

x Implement mechanisms to track and analyze the content and quality of contributions

xi Develop strategies to signal ongoing project activity and future maintenance to attract and retain contributors.

4.7 Article 7

"Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions" by Greene & Fischer [7].

Derived Actionables
[No actionable recommendation could be derived from the article]

4.8 Article 8

Studying the use of developer IRC meetings in open source projects by Shihab et al. [8].

Derived Actionables
[No actionable recommendation could be derived from the article]

4.9 Article 9

"More common than you think: An in-depth study of casual contributors" by Pinto et al. [9].

Derived Actionables
[No actionable recommendation could be derived from the article]

4.10 Article 10

"Evolution of the core team of developers in libre software projects" by Robles et al. [10].

Derived Actionables

[No actionable recommendation could be derived from the article]

References

[1] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. 2019. On the abandonment and survival of open source projects: An empirical investigation. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, 1–12.

[2] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. 2019. What attracts newcomers to onboard on oss projects? tl; dr: Popularity. In Open Source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26–27, 2019, Proceedings 15. Springer, 91–103."

[3] "Gerardo Matturro, Karina Barrella, and Patricia Benitez. 2017. Difficulties of newcomers joining software projects already in execution. In 2017 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, 993–998.

[4] Bin Lin, Gregorio Robles, and Alexander Serebrenik. 2017. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In 2017 IEEE 12th International Conference on Global Software Engineering (ICGSE). IEEE, 66–75.

[5] Fabian Fagerholm, Alejandro S Guinea, Jürgen Münch, and Jay Borenstein. 2014. The role of mentoring and project characteristics for onboarding in open source software projects. In Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement. 1–10

[6] Leonard Przybilla, Maximilian Rahn, Manuel Wiesche, and Helmut Krcmar. 2019. The more the merrier? The effect of size of core team subgroups on success of open source projects. (2019)."

[7] Gillian J Greene and Bernd Fischer. 2016. Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions. In Proceedings of the 31st IEEE/ACM international conference on automated software engineering. 804–809.

[8] Emad Shihab, Zhen Ming Jiang, and Ahmed E Hassan. 2009. Studying the use of developer IRC meetings in open source projects. In 2009 IEEE International Conference on Software Maintenance. IEEE, 147–156.

[9] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. 2016. More common than you think: An in-depth study of casual contributors. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (SANER), Vol. 1. IEEE, 112–123.

[10] Gregorio Robles, Jesus M Gonzalez-Barahona, and Israel Herraiz. 2009. Evolution of the core team of developers in libre software projects. In 2009 6th IEEE international working conference on mining software repositories. IEEE, 167–170.