Prioritizing Non-Functional Requirements and Influencing Factors for API Quality Framework: An Industry Approach

Aumir Shabbir[1], Aziz Deraman[2]*, Mohamad Nor Bin Hassan[3], Kamal Uddin Sarker[4], Shahid Kamal[5]*
Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu, Kuala Nerus Terengganu, 21030, Malaysia[1, 2, 3]
Department of Business and Management Studies, Gulf College, Al Mabaila - Muscat 133, Oman[1]
Department of Computer Science and MIS, Oman College of Management and Technology, Al Barka, 320, Oman[4]
Faculty of Computing & Informatics, Multimedia University, Cyberjaya, Selangor, 63100, Malaysia[5]

***Abstract*—Application Programming Interface (API) management is currently a trending research area; however, APIs require careful attention to Non-Functional Requirements (NFRs) to ensure system performance, maintainability, security, and resiliency. The software industry struggles to maintain API quality, especially NFRs, due to a focus on functional aspects in standards like the OpenAPI Specification (OAS). Similarly, standards, such as ISO/IEC 25010:2023, evaluate the quality of general software but offer limited guidance on addressing API challenges. Based on the industry perspective, this paper prioritizes the most critical quality attributes and their influencing factors for APIs, supporting the development of a Non-Functional Requirement Quality Framework for APIs (NFRQF-API). We adopted ISO/IEC 25010 as our reference standard and surveyed industry experts. Eleven NFRs are added in the survey, including nine from ISO/IEC 25010 and two additional attributes, Observability and Resiliency, identified through the literature review. A structured survey tool has been validated, pilot-tested, and distributed to 38 API practitioners, with data analyzed through IBM Statistical Package for the Social Sciences (IBM SPSS). SPSS demonstrates strong internal consistency ($\alpha > 0.7$) across items within each group. Additionally, Maintainability (4.29) and Resiliency (4.20) have been identified as core NFRs, while Interaction Capability (3.18), Flexibility (3.18), and Safety (2.93) have lower scores based on their mean calculation. The remaining six NFRs are moderately significant, highlighting their ongoing importance. These findings, based on NFR classification, establish a solid foundation for developing a Quality Framework for APIs aligned with modern Software engineering requirements. The article supports researchers and practitioners to build a strong understanding towards NFR prioritization, which is a crucial step for API quality management.**

Keywords*—Non-Functional Requirements (NFRs); Application Programming Interface (API); software development practices; API quality; Non-Functional Requirement Quality Framework for APIs (NFRQF-API); ISO/IEC 25010

I. Introduction

In the modern landscape, APIs have become the core elements of digital ecosystems, such as cloud computing, mobile platforms, and microservices architectures. APIs cover both technical and business aspects. Technically, solutions are provided by the APIs based on the problems faced by business organizations and establish a set of requirements that determine how the application interacts and supports in the exchange of data. On the other hand, from a business perspective. APIs serve the organizations in utilizing their resources more efficiently to generate value, both across various departments and in collaboration with external partners [1]. As APIs gain significant popularity in the modern software ecosystems, their quality, explicitly focusing on NFRs, is crucial for their success [2]. However, despite their importance, organizations mainly face challenges in managing API quality, particularly regarding scalability, security, maintainability, and performance [3], [4].

NFRs are referred to as constraints and quality attributes [5], however, neglecting them can result in expensive post-development rework in software applications, which is usually a significant risk in the success of any project [6], [7]. Similarly, Observability and Resiliency have emerged as vital quality attributes in managing the complexity and reliability of modern APIs and microservices. As the behaviour of an API differs from that of traditional software, Observability as a quality attribute not only specifies the quality of the API's behaviour but also the tools, logs, and monitoring systems that provide transparency into its real-time operations [8], [9]. Likewise, Resiliency supports the graceful handling of failures without any disruptions in API services [10], [11]. Recent studies have demonstrated that integrating degradation and recovery through automated requirement-driven adaptations significantly improves overall system resilience and minimizes downtime [12].

The OpenAPI Specification (OAS) is one of the popular existing RESTful APIs standards [13], but it primarily focuses on functional aspects [14], [15], and limited to address the NFRs [16], [17]. Additionally, international quality standards, such as ISO/IEC 25010:2023, outline nine quality attributes, including Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, Flexibility, and Safety [18] but it mainly focuses on maintaining the quality of general software, and is limited in addressing API-specific challenges [19]. Therefore, the lack of comprehensive API guidance in these standards may lead to poor API management, which can impact quality and reliability [20]. Hence, improving API quality through a comprehensive framework that considers the broader range of quality attributes, including both design-time and runtime aspects, remains a

*Corresponding author.

significant unexplored domain in software engineering [21], [22], "in press" [23].

This study identifies and prioritizes key NFRs along with their influencing factors that affect API quality, based on industry perception. The industry feedback has been collected through a structured, validated questionnaire that underwent expert reviews, pilot testing, and statistical validation, contributing a refined classification of NFRs relevant to real-world software development. The findings based on the classification of NFRs support the development of an NFR Quality Framework for APIs, which builds a strong understanding towards the quality management of APIs for both researchers and industry.

This paper is organized into the following sections: Section II presents the literature review, Section III describes the methodology, including the comprehensive NFR prioritization process, while Section IV covers the results and a brief discussion. Similarly, Section V discusses practical and theoretical implications, whereas Section VI concludes by summarizing the findings and proposing directions for future research.

II. Literature Review

A. Significance of Application Programming Interface (API)

The API standard was first developed in the 1940s as a modular software library for EDSAC [24]. The term "Application Program Interface" was first introduced in the 1960s [25] to denote the data structures and functions associated with computer graphics on remote systems [26]. In the 2000s, as the Internet became famous, the Representational State Transfer (REST) architecture was formalized with the name of a network-based API, where the REST and similar APIs have been collected and are called "Web API" [27]. The growing significance of APIs in today's dynamic digital environment has accelerated software development [28], [29]. APIs serve as intermediaries, facilitating communication between various systems and enhancing integration, which promotes efficiency, innovation, and collaboration [30].

1) API development life cycle: APIs are not just a component that a developer can set and forget [31]. Developers need to plan, create, deploy, and monitor APIs to keep them aligned with business needs and know when to retire them. In his article, Stephen J. Bigelow [32] explains the five stages of an API lifecycle: Plan, Develop, Test, Deploy and Retire, as shown in Fig. 1:

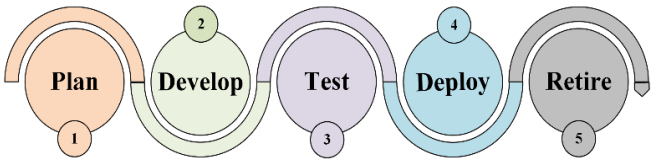

Fig. 1. API development life cycle.

B. Managing Non-Functional Requirements (NFRs)

One of the significant challenges in the dynamic world of software development is managing Non-Functional Requirements (NFRs) because of the deployment and frequent changes they face during the development lifecycle [33], [34]. NFR, as a term, was first used by Roman, G-C., in 1985 [35]. Furthermore, this term is also called quality attributes [5], goals, and non-behavioural requirements, respectively [35]. Krishnamurthy and Saran (2007) use the term "auxiliary requirements" to expand the interpretation of NFR [36]. Some NFRs are crucial and required to be addressed while designing the APIs, such as reliability, security, performance, maintainability, compatibility, and usability [37]. NFRs impact software systems' overall quality and success in a growing software development environment, where a poor approach to NFR management may be the reason for project failure [38].

Observability is one of the key attributes that support enhancing the system's reliability and performance [39]. It covers most of the post-deployment aspects of APIs, such as logging, monitoring, and metrics collection [40], [41]. Similarly, the observability also supports efficient debugging and cost management in APIs [42], [43]. Organizations can ensure that their APIs are robust, efficient, and cost-effective by following the three main components of observability, such as logging completeness, real-time monitoring, and distributed tracing [43], [44]. Similarly, the Resiliency covers fault tolerance, graceful degradation, automatic recovery, and self-healing, which are the crucial aspects in maintaining continuous service and reliability in distributed systems [45]. Fault tolerance is a crucial aspect in APIs that ensures that systems can continue their services even in the failures [12]. It provides the reliability of APIs during failures, which is essential for maintaining high availability and performance. Automatic recovery mechanisms are integral to self-healing systems, which aim to restore normal operations after a failure [46]. Kumar et al. in 2022 [47] highlight the critical importance of NFRs during software development by emphasizing that usability, security, and reliability are often overlooked or inefficiently addressed. While various existing development methodologies, such as Agile, Scrum, Spiral, DevOps, etc., mainly focus on the Functional areas [48], however, the developers ignore the crucial aspect of NFRs when designing the APIs [49], even though the quality of an application is primarily based on the NFRs [50].

Like traditional software, the quality of an API is crucial [51] that can be analyzed by evaluating multiple aspects, such as functional and non-functional [52]. The functional quality is mainly maintained by referring to the requirements that have been collected and documented. However, the Non-Functional Requirements are partially addressed, which can affect the quality of the software [53] as well as APIs [15]. Hence, software quality is essential in a software development environment and needs to be maintained frequently [54].

C. Review of Quality Factors Addressed

ISO/IEC25010 is the latest product quality model, updated in 2023 as the second edition. This model belongs to the Systems and Software Quality Requirements and Evaluation (SQuaRE) family, which applies to Information and Communication Technology (ICT) and software products. To assess the quality of the products, this model comprises nine quality characteristics [20], such as, Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, Flexibility, and Safety [20], as shown in Fig. 2.

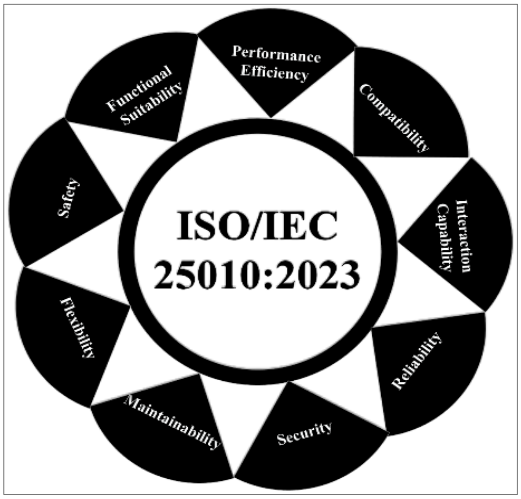

Fig. 2. A Diagrammatic model of ISO/IEC 25010:2023.

In Table I, we have analyzed few studies based on quality attributes addressed such as, Functional Suitability (FS), Performance Efficiency (PE), Compatibility (CY), Interaction Capability (IC), Reliability (RL), Security (SE), Maintainability (MY), Flexibility (FY), Safety (SA), along with the two emerging quality attributes, Observability (OY) and, Resiliency (RS).

By summarizing the review of the above-mentioned articles in terms of the comparison of quality factors addressed, Table I shows that the majority of articles only addressed Performance efficiency, Security, Maintainability, and to some extent, Functional suitability and Reliability as quality attributes in their research. However, their focus on the remaining NFRs mentioned in ISO/IEC 25010:2023 is limited. The two emerging quality attributes, Observability and Resiliency, identified in the literature review, are also addressed in a limited manner. However, the NFRQF-API effectively addresses all key non-functional quality attributes in alignment with ISO/IEC 25010.

TABLE I. LITERATURE REVIEW ON QUALITY FACTORS ADDRESSED

References	List of Quality Attributes										
	FS	***PE***	***CY***	***IC***	***RL***	***SE***	***MY***	***FY***	***SA***	***OY***	***RS***
[46]	✓	✓	□	□	✓	✓	✓	✓	□	✓	✓
[12]	✓	✓	✗	✗	✓	✓	✓	□	✗	✓	✓
[55]	✓	✓	□	✗	✓	✓	✓	□	□	✗	✗
[56]	✗	✓	✗	□	✗	✗	✗	✗	✗	✗	✗
[41]	✓	✓	✗	✗	✓	✓	✓	□	✗	✓	□
[47]	✓	✓	□	□	✓	✓	✓	□	□	✗	□
[57]	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
[58]	✗	✓	✗	□	✗	✓	✗	✗	✗	✗	✗
[59]	✗	✓	✗	□	✓	✓	✓	✗	✓	✗	✗
[60]	✗	✓	✗	□	✓	✓	✓	✓	✗	✗	✗
[61]	✗	✓	✗	□	✗	✓	✓	✗	✗	✗	✗
[62]	✗	✓	✗	□	✓	✓	✓	✗	✗	✗	✗
[63]	✗	✓	✓	□	✓	✓	✓	✗	✗	✗	✗
[64]	✓	✓	✗	✗	✓	✓	✓	✓	✗	□	□
[65]	✓	✓	□	✗	✓	✓	✓	□	✗	✗	□
[66]	✓	✓	✗	✗	✓	✓	✓	□	✗	✗	□
[67]	✓	✓	✗	✗	✓	✓	✓	□	✗	✗	□
[68]	✓	✓	□	□	✓	✓	✓	✓	✓	✗	□
[69]	✓	✓	✗	✗	✓	✓	✓	□	✗	✗	□
[70]	✓	✓	✗	✗	✓	✓	✓	□	✗	□	□
NFRQF- API	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Legend: ✓ =Fully Addressed, ✗=Not Addressed, □= Partially Addressed											

III. METHODOLOGY

This study adopts a quantitative research methodology to identify and prioritize the key NFRs that affect the quality of APIs. The quantitative approach has been selected due to its suitability in collecting expert perceptions. Fig. 3 shows a systematic structure of the Methodology adopted in this study. The structure of the Methodology consists of five phases, including Survey design, Expert Validation Process, Survey Distribution (covering both Pilot and Final Survey distribution),

Cronbach's alpha reliability test using IBM SPSS Statistics, and Data analysis (utilizing descriptive statistical tools such as mean and Standard Deviation).

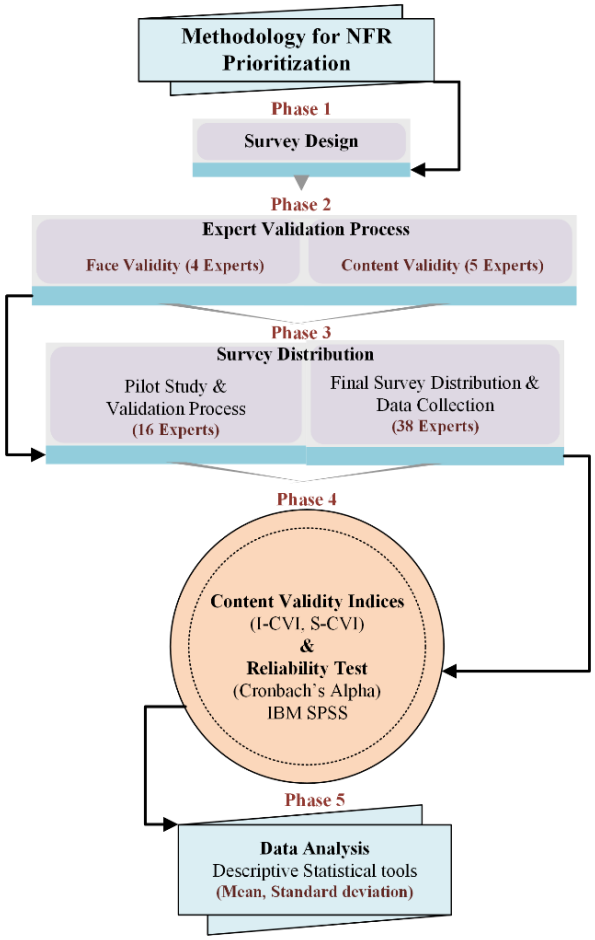

Fig. 3. A Structure of a research methodology structure.

A. Implementation of Methodology

1) Phase 1: Survey Design. The questionnaire for the pilot study comprises two primary sections: demographic data to contextualize participant responses, and targeted questions assessing the perceived importance of quality attributes mentioned in ISO/IEC 25010:2023. The first section includes 12 closed-ended and open-ended questions to collect detailed demographic data, focusing on participants' professional roles, industry sectors, and experience with API management. The second section includes 47 closed-ended questions linked to 11 quality attributes and the factors influencing them. Each quality attribute grid includes a mandatory 5-point Likert scale to assess respondent feedback, ensuring accurate and consistent responses.

2) Phase 2: Expert Validation Process. To ensure the accuracy, clarity, and appropriateness of the questionnaire, both face validity and content validity [71] assessments have been conducted prior to distribution, as mentioned below:

a) Face validity. The face validity [71], [72] of the questionnaire has been assessed through direct consultations with four academic experts possessing relevant backgrounds in research methodology, software engineering, and computer science. Their feedback has been pursued to ensure that the questionnaire is not only visually appealing and structurally sound but also suitable for effective data collection and analysis. Their feedback addressed:

- The overall clarity and understandability of the questions.
- The formatting of Likert scale responses for consistency and ease of use.
- Vocabulary adjustments to simplify technical jargon for broader respondent comprehension.
- Structural suggestions include expanding the number of questions per variable to enhance statistical reliability.

b) Content validity. Content validity [71], [73] is assessed by five domain experts in API development and software architecture, who evaluated the questionnaire for:

- Relevance and representativeness of the selected NFRs.
- Alignment with real-world industry practices, such as development environments, tools, and common development challenges.
- Addition of two novel NFRs and improvements to demographic and technical sections.

Overall, other industry experts have provided positive and encouraging feedback on the questionnaire. They also recognized that the inclusion of two emerging NFRs from the literature, along with attributes mentioned in the ISO/IEC 25010 standard, enhances the questionnaire's comprehensiveness and practicality. This feedback shows the applicability and usefulness of the questionnaire for both academic and professional domains.

3) Phase 3: Survey distribution

a) Pilot study and validation process. To ensure the instrument's quality and reliability, a pilot study [74], [75] with 16 respondents across various countries has been conducted. By following the guidelines of M. Tavakol and R. Dennick (2011) [76], a Cronbach's Alpha has been employed, which has supported identifying issues related to low reliability, such as redundancy, ambiguity, and imbalance among items. Based on the pilot results presented in Table II, overlapping or unclear questions have been refined, and the questionnaire structure has been adjusted to assign an equal number of 4 items, resulting in a total number of 44 questions. This adjustment helps to avoid potential biases in the calculation of mean scores.

b) Final survey distribution. The questionnaire-based survey methodology employed a non-probabilistic convenience sampling approach, following the expert recommendations of Wohlin, Fowler Jr, and Kasunic [77], [78], [79] to gather data from industry experts specialized in API domains across multiple countries. Convenience sampling involves selecting participants based on their accessibility and availability, with a particular focus on their expertise and relevance to API management.

c) Data collection. Data collection has occurred through an online survey platform (Google Forms), chosen for its accessibility and effectiveness in managing responses. The survey has been accessible to respondents for over four weeks, allowing industry experts sufficient flexibility to participate and complete the questionnaire conveniently. A total of 38 relevant and valid responses have been collected during the four-week period.

4) Phase 4: Content Validity Indices (I-CVI and S-CVI) and Reliability Test

a) Item-level content validity index and scale-level content validity index. Table III provides a detailed summary of the I-CVI (Item-level Content Validity Index) values for individual items and the overall S-CVI (Scale-level Content Validity Index) score, reflecting the content validity of the questionnaire.

TABLE II. GROUP-WISE CRONBACH'S ALPHA RELIABILITY FOR NFR ITEMS (PILOT STUDY)

Sr.#	NFRs	Number of Items	Cronbach's Alpha (Pilot Study)
1.	FS	5	0.61
2.	PE	4	0.57
3.	CY	4	0.41
4.	IC	5	0.72
5.	RL	3	0.48
6.	SE	4	0.75
7.	MY	5	0.54
8.	FY	3	0.52
9.	SA	5	0.50
10.	OY	4	0.80
11.	RS	5	0.77

TABLE III. I-CVI (ITEM-LEVEL CVI) AND S-CVI (SCALE-LEVEL CVI)

NFR	Factors	Experts rating 3 or 4	I-CVI
FS	FS1: It is important that the API provides all the required functions for users.	5	1.00
	FS2: It is important that the API covers all tasks and goals thoroughly.	4	0.80
	FS3: It is important that the API provides accurate and reliable results.	5	1.00
	FS4: It is essential that Functional Suitability supports API quality by allowing tasks to be completed without unnecessary steps.	5	1.00
PE	PE1: It is important that the API provides a quick response time under expected load conditions.	4	0.80
	PE2: It is important that the API ensures minimal resource usage during operations.	5	1.00
	PE3: It is important that the API efficiently handles high traffic, ensuring performance under peak conditions.	5	1.00
	PE4: It is essential that Performance Efficiency enhances API speed, responsiveness, and resources used to ensure quality.	4	0.80
CY	CY1: It is important that the API integrates seamlessly with other systems, regardless of platform.	4	0.80
	CY2: It is important that the API supports multiple platforms (e.g., mobile, web, IoT) and environments.	5	1.00
	CY3: It is important that the API ensures backward compatibility with previous versions.	4	0.80
	CY4: It is essential that Compatibility supports API quality by ensuring seamless integration with multiple platforms and systems.	5	1.00
IC	IC1: It is important that the API's interface is intuitive and easy for users to operate and control.	4	0.80
	IC2: It is important that the API provides meaningful error messages for troubleshooting.	4	0.80
	IC3: It is important that users can quickly learn to use the API.	4	0.80
	IC4: It is essential that an API's quality depends on strong Interaction Capability, including clear documentation, a user-friendly interface, and easy integration.	4	0.80
RL	RL1: It is important that the API consistently performs its intended functions without failure, even under load.	5	1.00
	RL2: It is important that the API ensures high availability with minimal downtime to maintain service continuity.	5	1.00
	RL3: It is important that the API remains operational even in the event of an error.	5	1.00
	RL4: It is essential that Reliability supports API quality by ensuring consistent performance, high availability, and minimal downtime.	4	0.80
SE	SE1: It is important that the API provides strong authentication and authorization to secure access.	5	1.00
	SE2: It is important that the API uses data encryption during transmission and storage to protect sensitive information.	5	1.00
	SE3: It is important that the API is resilient against common security threats, such as injection and cross-site scripting attacks.	4	0.80
	SE4: It is essential that Security supports API quality by protecting data, preventing unauthorized access, and meeting security standards.	4	0.80
MY	MY1: It is important that the API has a clear modular architecture to support easy maintenance and upgrades.	5	1.00
	MY2: It is important that the API allows seamless updates and versioning without disrupting service or breaking backward compatibility.	5	1.00
	MY3: It is important that well-structured and clear documentation helps maintain and troubleshoot the API efficiently.	5	1.00

	MY4: It is essential that Maintainability enables easy updates, bug fixes, and adaptation to new requirements over time.	5	1.00
FY	FY1: It is important that the API supports multiple deployment models (e.g., cloud, on-premises).	4	0.80
	FY2: It is important that the API can be installed easily.	4	0.80
	FY3: It is important to design APIs that are scalable to support flexibility.	4	0.80
	FY4: It is essential that the API stays flexible so it can adapt to new business needs and changing technologies.	5	1.00
SA	SA1: It is important that the API does not cause harm to people, property, or the environment.	4	0.80
	SA2: It is important that the API works within safe limits to prevent unsafe actions.	4	0.80
	SA3: It is important that the API can help find risks that could cause harm.	4	0.80
	SA4: It is essential that Safety helps ensure API quality by preventing unsafe actions and switching to a safe mode if something goes wrong.	4	0.80
OY	OY1: It is important that the API captures and retains logs for debugging, auditing, and monitoring performance.	5	1.00
	OY2: It is important that the API provides real-time monitoring and alerts to notify of any failures.	4	0.80
	OY3: It is important that the API uses distributed tracing to track requests across services.	5	1.00
	OY4: It is essential that Observability supports API quality by enabling monitoring, logging, and troubleshooting to maintain optimal performance.	5	1.00
RS	RS1: It is important that the API handles failures smoothly without disrupting service.	5	1.00
	RS2: It is important that the API keeps partial functionality available instead of failing completely.	5	1.00
	RS3: It is important that the API can recover automatically from failures without requiring manual intervention.	5	1.00
	RS4: It is essential that Resiliency supports API quality by effectively managing failures, recovering quickly, and maintaining uninterrupted operations.	5	1.00
S-CVI (Average of I-CVIs)			**0.91**

N= Number of Experts = 5

M= Number of Items = 44

A_i= Number of experts who rated item (i) as 3 or 4

$$I-CVI_i = \frac{Number\ of\ experts\ rating\ 3\ or\ 4\ (A_i)}{Total\ number\ of\ Experts}$$

IF, (A_i= 5 and N= 5) $I-CVI_i\ (for\ 24\ items) = \frac{5}{5},\ I-CVI_i = 1.00$

IF, (A_i= 4 and N= 5)

$$I-CVI_i\ (for\ 20\ items) = \frac{4}{5}, \qquad I-CVI_i = 0.80$$

$$S-CVI_{Ave} = \frac{\sum (I-CVI\ for\ all\ items)}{Total\ number\ of\ Items\ (M)}$$

Hence,

$$S-CVI_{Ave} = \frac{40}{44}, \qquad S-CVI_{Ave} = 0.91$$

To ensure the relevance and appropriateness of the questionnaire items before the final distribution, a Content Validity Index (CVI) has been calculated based on ratings from five domain experts. Each expert has rated all 44 items on a 4-point scale (1 = Not Relevant to 4 = Highly Relevant). An item-level CVI(I-CVI) has been computed by dividing the number of experts who rated an item as 3 or 4 by the total number of experts. All 44 items have achieved an I-CVI ≥ 0.78, indicating acceptable content validity based on the standard threshold recommended by Lynn (1986) [80] and supported by the recent studies of Almohanna et al., (2022) [81]. Thus, no items have been eliminated at this stage. By following the guidelines of Polit, D. F., & Beck, C. T. (2006), the Scale-level Content Validity Index (S-CVI) has been calculated as the average of all I-CVI values across the instrument, which shows 0.91, indicating excellent overall content validity [73].

b) Reliability test (Cronbach's alpha). By following the guidelines of M. Tavakol and R. Dennick (2011) [76], a Cronbach's Alpha is employed to assess the reliability of the questionnaire shown in Table IV, followed by Fig. 4, which ensures that the set of questionnaire items is reliable.

The above-mentioned Table IV and Fig. 4 demonstrate the acceptable reliability ($\alpha > 0.7$) [76], [82] of the items mentioned against each NFR. Reliability coefficients across all NFRs in the final study exceeded the acceptable threshold of 0.70, with Compatibility (CY), Security (SE), and Observability (OY) achieving the highest reliability, above 0.80. However, Functional Suitability (FS), Performance Efficiency (PE), Interaction Capability (IC), Reliability (RL), Maintainability (MY), Flexibility (FY), Safety (SA), and Resiliency (RS) remained between 0.70 and 0.80, indicating good reliability coefficients.

5) Phase 5: Data Analysis. The collected data have been prepared for analysis through coding and categorization based on participant responses to Likert-scale items. By applying IBM SPSS Statistics, descriptive statistical tools such as mean and standard deviation are used to compute the ranking of NFRs for identification purposes. Table V, together with Fig. 5, below, shows the mean and standard deviation range to compute the ranking of NFRs. By following the guidelines of "Boone & Boone, 2012" [83], an interpretation based on the mean and standard deviation has also been included in the table.

TABLE IV. GROUP-WISE CRONBACH'S ALPHA RELIABILITY FOR NFR ITEMS (FINAL STUDY)

Sr.#	NFRs	Final Study	
		Number of Items	*Cronbach's Alpha*
1.	FS	4	0.72
2.	PE	4	0.74
3.	CY	4	0.81
4.	IC	4	0.73
5.	RL	4	0.76
6.	SE	4	0.82
7.	MY	4	0.75
8.	FY	4	0.76
9.	SA	4	0.76
10.	OY	4	0.82
11.	RS	4	0.79

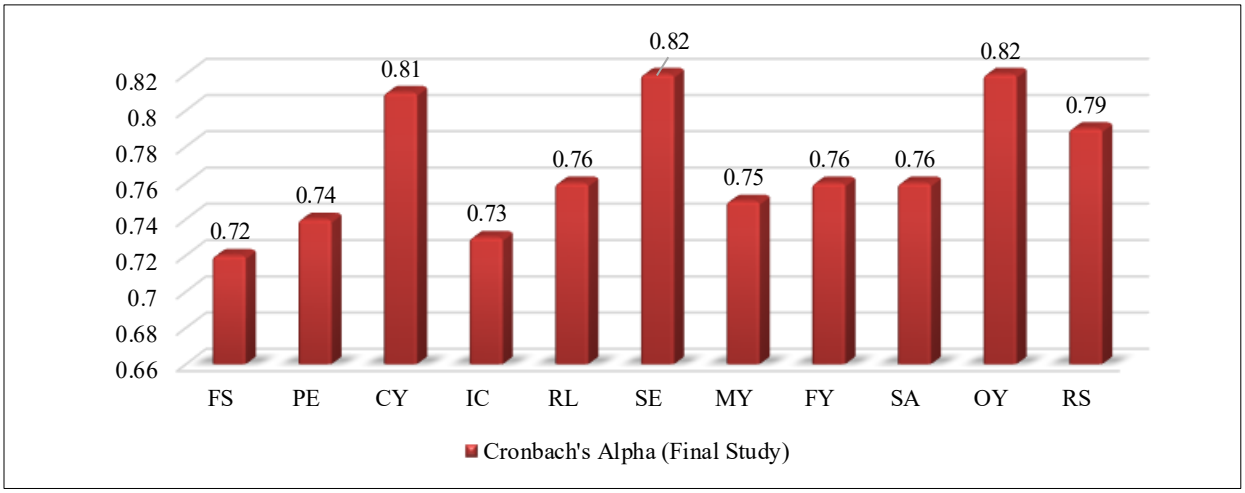

Fig. 4. Group-wise Cronbach's alpha reliability for NFR items (final study).

TABLE V. OVERALL MEAN AND STANDARD DEVIATION FOR 11 NFRS

NFR	Mean	Standard Deviation	Interpretation
MY	4.29	0.58	High Importance, High Consensus
RS	4.20	0.69	High Importance, High Consensus
FS	4.18	0.54	Moderate Importance, High Consensus
OY	4.16	0.61	Moderate Importance, High Consensus
SE	4.14	0.65	Moderate Importance, High Consensus
RL	4.09	0.62	Moderate Importance, High Consensus
PE	4.08	0.65	Moderate Importance, High Consensus
CY	3.97	0.66	Moderate Importance, High Consensus
IC	3.18	0.68	Low to Moderate Importance, High Consensus
FY	3.18	0.55	Low to Moderate Importance, High Consensus
SA	2.93	0.63	Low to Moderate Importance, High Consensus

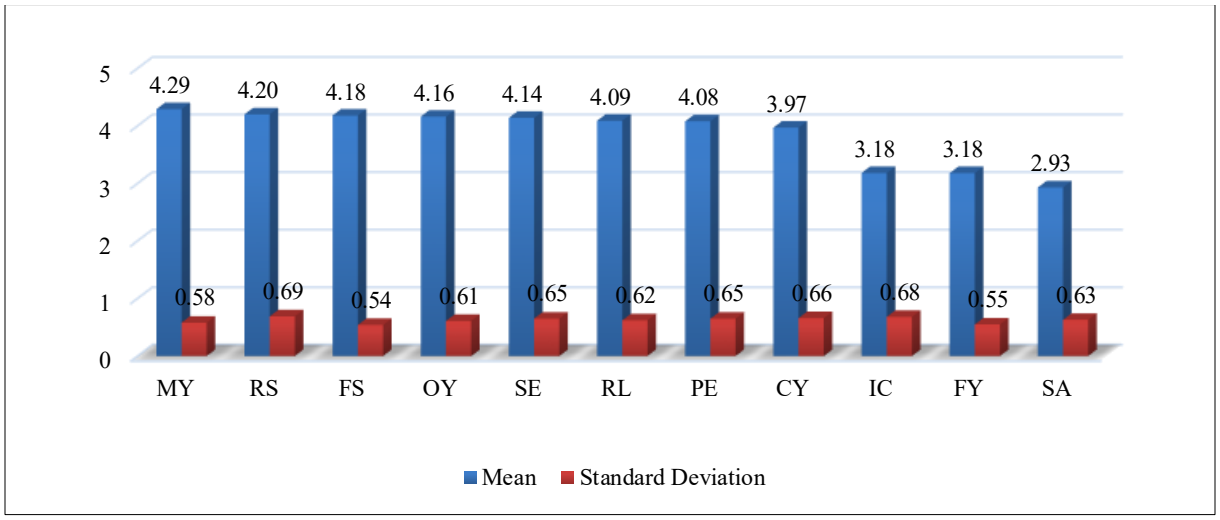

Fig. 5. NFR-wise mean and standard deviation.

The above-mentioned Table V, followed by Fig. 5 shows that Maintainability and Resiliency achieve "High Importance" as Core NFRs, while Functional Suitability (FS), Observability (OY), Security (SE), Reliability (RL), Performance Efficiency (PE), and Compatibility (CY) are considered to have moderate importance and are regarded as Critical NFRs. Conversely, Interaction Capability (IC), Flexibility (FY), and Safety (SA) are assessed as "Low to Moderate Importance" and are interpreted as Contextual NFRs. On the other hand, the standard deviation indicates a high level of variability in participants' responses to each item in the questionnaire.

For a detailed analysis of the factors influencing each NFR, Table VI offers an interpretation of each factor based on the mean and standard deviation.

TABLE VI. INTERPRETATION BASED ON FACTOR-WISE MEAN AND STANDARD DEVIATION

NFR	Factors	Mean	Standard Deviation	Mean Interpretation	Standard Deviation Interpretation
FS	FS1	4.29	0.65	High Importance	High Consensus
	FS2	4.03	0.82	Moderate Importance	Moderate Consensus
	FS3	4.03	0.79	Moderate Importance	High Consensus
	FS4	4.37	0.63	High Importance	High Consensus
PE	PE1	4.18	0.83	Moderate Importance	Moderate Consensus
	PE2	3.87	0.96	Moderate Importance	Moderate Consensus
	PE3	4.03	1.00	Moderate Importance	Moderate Consensus
	PE4	4.24	0.63	High Importance	High Consensus
CY	CY1	3.95	0.87	Moderate Importance	Moderate Consensus
	CY2	4.26	0.72	High Importance	High Consensus
	CY3	3.79	0.87	Moderate Importance	Moderate Consensus
	CY4	3.87	0.84	Moderate Importance	Moderate Consensus
IC	IC1	3.21	0.93	Low to Moderate Importance	Moderate Consensus
	IC2	3.18	0.93	Low to Moderate Importance	Moderate Consensus
	IC3	3.11	0.89	Low to Moderate Importance	Moderate Consensus
	IC4	3.24	0.88	Low to Moderate Importance	Moderate Consensus
RL	RL1	3.87	0.99	Moderate Importance	Moderate Consensus
	RL2	4.05	0.80	Moderate Importance	Moderate Consensus
	RL3	4.05	0.80	Moderate Importance	Moderate Consensus
	RL4	4.37	0.63	High Importance	High Consensus
SE	SE1	4.18	0.8	Moderate Importance	Moderate Consensus
	SE2	4.11	0.86	Moderate Importance	Moderate Consensus

	SE3	3.95	0.87	Moderate Importance	Moderate Consensus
	SE4	4.34	0.71	High Importance	High Consensus
MY	MY1	4.21	0.74	High Importance	High Consensus
	MY2	4.05	0.90	Moderate Importance	Moderate Consensus
	MY3	4.53	0.65	High Importance	High Consensus
	MY4	4.37	0.75	High Importance	High Consensus
FY	FY1	3.11	0.80	Low to Moderate Importance	Moderate Consensus
	FY2	3.16	0.79	Low to Moderate Importance	High Consensus
	FY3	3.24	0.63	Low to Moderate Importance	High Consensus
	FY4	3.21	0.66	Low to Moderate Importance	High Consensus
SA	SA1	2.84	0.82	Low to Moderate Importance	Moderate Consensus
	SA2	2.84	0.89	Low to Moderate Importance	Moderate Consensus
	SA3	3.29	0.69	Low to Moderate Importance	High Consensus
	SA4	2.74	0.86	Low to Moderate Importance	Moderate Consensus
OY	OY1	4.16	0.82	Moderate Importance	Moderate Consensus
	OY2	4.05	0.84	Moderate Importance	Moderate Consensus
	OY3	4.18	0.69	Moderate Importance	High Consensus
	OY4	4.26	0.69	High Importance	High Consensus
RS	RS1	3.97	0.91	Moderate Importance	Moderate Consensus
	RS2	4.32	0.84	High Importance	Moderate Consensus
	RS3	4.32	0.90	High Importance	Moderate Consensus
	RS4	4.21	0.87	High Importance	Moderate Consensus

The above-mentioned Table VI shows an in-depth analysis of the factors associated with each NFR, where each NFR is divided into four items based on the factors affecting it. Table VI presents the Mean, Standard deviation, and interpretation based on the mean and standard deviation of the four items for each NFR. Out of the 11 NFRs (44 items), 8 NFRs (32 items) are categorized as having “High” or “Moderate” importance. However, 3 NFRs (12 items) specifically Interaction Capability (IC), Flexibility (FY), and Safety (SA) are assessed as having "Low to Moderate" importance.

IV. Results and Discussion

A. Comparison of Pilot and Final Studies

Table VII, followed by Fig. 6 below, summarizes the key differences between pilot and final reliability metrics.

TABLE VII. Group-Wise Cronbach’s Alpha Reliability Comparison of Pilot and Final Study

Sr.#	NFR	Pilot Study		Final Study	
		Number of Items	*Cronbach's Alpha*	*Number of Items*	*Cronbach's Alpha*
1.	FS	5	0.61	4	0.72
2.	PE	4	0.57	4	0.74
3.	CY	4	0.41	4	0.81
4.	IC	5	0.72	4	0.73
5.	RL	3	0.48	4	0.76
6.	SE	4	0.75	4	0.82
7.	MY	5	0.54	4	0.75
8.	FY	3	0.52	4	0.76
9.	SA	5	0.50	4	0.76
10.	OY	4	0.80	4	0.82
11.	RS	5	0.77	4	0.79

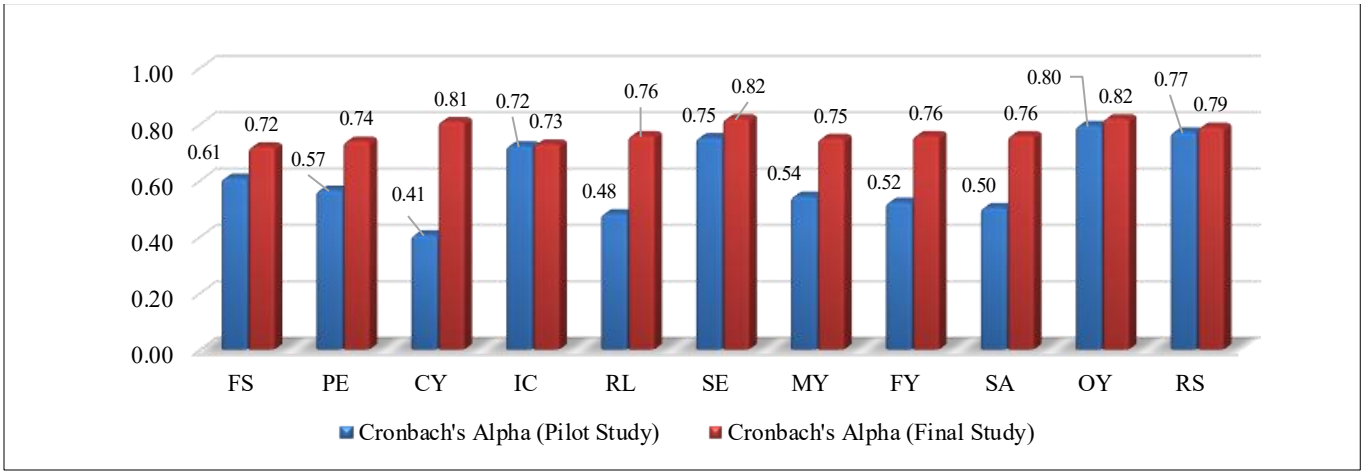

Fig. 6. Group-wise Cronbach's alpha reliability comparison of pilot and final study.

Based on the comparison between the Pilot and Final studies mentioned in the above Table VII, the final survey achieved higher Cronbach's alpha scores and confirmed improved internal consistency, with group-wise reliability for each NFR exceeding the threshold of 0.70, demonstrating strong reliability across all individual dimensions.

The Item-Level Content Validity Index (I-CVI) was calculated as the proportion of experts rating each item as three or four. Twenty-four items were rated three or four by all five experts, resulting in an I-CVI of 1.00. Conversely, 20 items received ratings of three or four from four experts, resulting in an I-CVI of 0.80. Consequently, all items exceeded the recommended minimum of 0.78 for five reviewers. Using the average method, the Scale-Level Content Validity Index (S-CVI_{Ave}) also confirmed the excellent overall content validity of the instrument as 0.91.

B. Prioritization of NFRs

Analysis of expert responses based on the Descriptive analysis of NFRs (mentioned in Tables V and VI) shows:

- Maintainability (MY) and Resiliency (RS) have been rated as highly important (Mean ≥ 4.20) with strong consensus (SD ≤ 0.79) and are classified as Core NFRs.
- Functional Suitability (FS), Performance Efficiency (PE), Compatibility (CY), Reliability (RL), Security (SE), and Observability (OY) demonstrate moderate importance (Mean 3.40 – 4.19), with varying levels of consensus and are categorized as Critical NFRs.
- Interaction Capability (IC), Flexibility (FY), and Safety (SA) are rated as low to moderate significance (Mean 2.60 – 3.39), with standard deviations reflecting considerable variability in responses and are classified as Contextual NFRs.
- The factor-wise analysis also reveals similar trends: the dimensions related to MY and RS were rated as highly important, while those associated with IC, FY, and SA scored comparatively lower.

C. Validation and Related Work

The classification of NFRs into Core, Critical, and Contextual categories was validated through experts' feedback, as demonstrated by the results of descriptive statistics. These results align with recent research that identifies Maintainability [84], [85] and Resiliency [10] as the essential NFRs of API quality and long-term adaptability [86]. Other studies highlight Resiliency as a key attribute for modern microservice and cloud-native APIs, emphasizing fault tolerance and automatic recovery to ensure uninterrupted service [11], [12]. Similar support exists for Functional Suitability, Performance Efficiency, Compatibility, Reliability and Security as primary quality attributes in API design [85], [3], [4]. Likewise, observability as a quality attribute not only defines the behaviour of the API but also covers the tools, logs, and monitoring systems that offer transparency into its real-time operations [8], [9]. By integrating these findings with the ISO/IEC 25010 quality model and expanding it to incorporate emerging factors such as Observability and Resiliency, this study bridges established standards with current industry needs, offering both theoretical foundation and practical relevance.

V. Practical and Theoretical Implications

This research offers both practical and theoretical contributions through the presentation of a validated prioritization of NFRs for API development. These requirements are categorized as Core, Critical, and Contextual, aligning with ISO/IEC 25010 while incorporating emerging NFRs such as Observability and Resiliency. The classification supports industry professionals in early planning of high- and medium-prioritized NFRs and their influencing factors during the development phases of APIs. Conversely, lower-prioritized Contextual NFRs are addressed as context-based on specific project requirements. Furthermore, theoretically, it enhances the literature by integrating emerging API-specific NFRs with an international standard, offering researchers with a structured and empirically supported foundation for future studies and for developing a comprehensive NFR-based API quality framework.

VI. Conclusion and Future Work

This study introduces a practical and validated instrument for capturing industry insights on API-related NFRs. The comprehensive pilot process, expert feedback, and strong internal consistency verify that the instrument is reliable and relevant for guiding future framework development. Moreover, our study successfully prioritize the most critical NFRs that influence API quality, as determined by relevant industry experts. We have gathered feedback from 38 respondents with considerable expertise in the API domain. The key findings emphasize that Maintainability and Resiliency are top priorities as core NFRs, reflecting the need for APIs that are easy to update, troubleshoot, and recover after failures. Meanwhile, moderate importance for critical NFRs highlights that Functional Suitability, Performance Efficiency, Compatibility, Reliability, Security, and Observability remain essential. Considering the low ratings for Interaction Capability, Flexibility, and Safety, our proposed framework should focus on developer-focused usability, in addition to the user-focused usability defined under Interaction Capability in the ISO standard. Furthermore, the framework will also assess whether the installability, as mentioned in the Flexibility and physical safety aspects of Safety, genuinely applies to the API domain.

Based on the preliminary findings, future research should focus on developing a comprehensive Non-Functional Requirement Quality Framework for APIs (NFRQF-API) that aligns with current quality standards. Additionally, these insights can significantly support both academic research and industry practices, offering a solid foundation for developing an API Non-Functional Requirements Quality Framework for APIs that aligns well with modern software engineering approaches and practical implementation needs.

Acknowledgment

We sincerely thank Universiti Malaysia Terengganu (UMT) and the Ministry of Higher Education (MOHE) Malaysia for their support in conducting this research. We also appreciate Gulf College, Muscat, Oman and the Ministry of Higher Education, Research and Innovation - Sultanate of Oman for their vital research assistance.

References

[1] S. Andreo and J. Bosch, "API Management Challenges in Ecosystems," in Eur. Commission, Louxembourg, Louxembourg, UK, Tech. Rep. JRC118082, 2019.

[2] A. Rashwan, "Automated quality assurance of non-functional requirements for testability," 2015.

[3] S. Andreo and J. Bosch, "API Management Challenges in Ecosystems," in Software Business: 10th International Conference, ICSOB 2019, Jyväskylä, Finland, November 18–20, 2019, Proceedings 10, Springer, 2019.

[4] M. Ahmad, J. J. Geewax, A. Macvean, D. Karger, and K.-L. Ma, "API Governance at Scale," in Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice, New York, NY, USA: ACM, Apr. 2024.

[5] F.-L. Li et al., "Non-functional requirements as qualities, with a spice of ontology," in 2014 IEEE 22nd International Requirements Engineering Conference (RE), IEEE, Aug. 2014.

[6] X. Zhang and X. Wang, "Tradeoff Analysis for Conflicting Software Non-Functional Requirements," IEEE Access, vol. 7, pp. 156463–156475, 2019.

[7] U. S. Shah, S. J. Patel, and D. C. Jinwala, "Constructing a Knowledge-Based Quality Attributes Relationship Matrix to Identify Conflicts in Non-Functional Requirements," J. Comput. Theor. Nanosci., vol. 17, no. 1, pp. 122–129, Jan. 2020.

[8] M. Cinque, R. Della Corte, and A. Pecchia, "Microservices Monitoring with Event Logs and Black Box Execution Tracing," IEEE Trans. Serv. Comput., vol. 15, pp. 294–307, 2019.

[9] M. Scrocca, R. Tommasini, A. Margara, E. Della Valle, and S. Sakr, "The Kaiju project: enabling event-driven observability," Proc. 14th ACM Int. Conf. Distrib. Event-based Syst., 2020.

[10] J. Zhang, Y. Rong, J. Cao, C. Rong, J. Bian, and W. Wu, "DBFT: A Byzantine Fault Tolerance Protocol With Graceful Performance Degradation," IEEE Trans. Dependable Secur. Comput., vol. 19, pp. 3387–3400, 2021.

[11] Y. Lin, S. Kulkarni, and A. Jhumka, "Automation of fault-tolerant graceful degradation," Distrib. Comput., vol. 32, pp. 1–25, 2019.

[12] P. Rajput and G. Sikka, "Multi-agent architecture for fault recovery in self-healing systems," J. Ambient Intell. Humaniz. Comput., vol. 12, pp. 2849–2866, 2020.

[13] A. Shabbir, N. Tahir, M. Shahzad, and T. D. A. Deraman, "Perceptions on Restful APIS Testing and Evaluation for Digitial AI Deployed Applications-A Literature Review and Systematic Survey," Well Test. J., vol. 33, no. S2, pp. 551–570, 2024.

[14] A. Karavisileiou, N. Mainas, and E. G. M. Petrakis, "Ontology for OpenAPI REST Services Descriptions," in 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, Nov. 2020.

[15] S. Bucaille, J. L. Cánovas Izquierdo, H. Ed-Douibi, and J. Cabot, "An OpenAPI-Based Testing Framework to Monitor Non-functional Properties of REST APIs," in International Conference on Web Engineering, Springer, 2020.

[16] R. Pergl and N. Jíša, "Semantic Analysis of API Blueprint and OpenAPI Specification," in World Conference on Information Systems and Technologies, Springer, 2024.

[17] A. Gamez-Diaz et al., "The role of limitations and SLAs in the API industry," in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 1006–1014.

[18] R. Nacheva, "Conceptual Model of a Software Accessibility Evaluation System".

[19] E. dos Santos and S. Casas, "API Management and SQuaRE: A Comprehensive Overview from the Practitioners' Standpoint," in Argentine Congress of Computer Science, Springer, 2023, pp. 137–150.

[20] ISO, "ISO/IEC 25010:2023," Online Browsing Platform (OBP). [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en

[21] L. Murphy, M. B. Kery, O. Alliyu, A. Macvean, and B. A. Myers, "API designers in the field: Design practices and challenges for creating usable APIs," in 2018 ieee symposium on visual languages and human-centric computing (vl/hcc), IEEE, 2018, pp. 249–258.

[22] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API testing methodologies: Rationale, challenges, and solution directions," Appl. Sci., vol. 12, no. 9, p. 4369, 2022.

[23] A. Shabbir, A. Deraman, M. Nor Bin Hassan, K. U. Sarker, and S. Kamal, "Enhancing API Quality: A Comprehensive Review of Non-Functional Requirements for Quality-centric Framework Development," KSII Trans. INTERNET Inf. Syst. ISSN 1976-7277, 2025.

[24] A. Svensson, "What is the best API from adeveloper's perspective?: Investigation of API development with fintechdevelopers in the spotlight," 2024.

[25] W. Granli, J. Burchell, I. Hammouda, and E. Knauss, "The driving forces of API evolution," in Proceedings of the 14th International Workshop on Principles of Software Evolution, 2015, pp. 28–37.

[26] J. Peddie, "Application Program Interface (API)," in The History of the GPU - Eras and Environment, Cham: Springer International Publishing, 2022.

[27] N. Kratzke, "A brief history of cloud application architectures," Appl. Sci., vol. 8, no. 8, p. 1368, 2018.

[28] T. Shimosawa, "Quality is not an act, it is a habit—Aristotle," Hypertens. Res., vol. 46, no. 5, pp. 1221–1226, May 2023.

[29] J. Ofoeda, R. Boateng, and J. Effah, "Application Programming Interface (API) Research," Int. J. Enterp. Inf. Syst., vol. 15, no. 3, pp. 76–95, Jul. 2019.

[30] M. Raatikainen, E. Kettunen, A. Salonen, M. Komssi, T. Mikkonen, and T. Lehtonen, "State of the Practice in Application Programming Interfaces (APIs): A Case Study," in European Conference on Software Architecture, Springer, 2021.

[31] B. Jin, S. Sahni, and A. Shevat, Designing Web APIs: Building APIs That Developers Love. " O'Reilly Media, Inc.," 2018.

[32] Stephen J. Bigelow, "5 stages of an API lifecycle explained," TechTarget.

[33] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are" non-functional" requirements really non-functional? an investigation of non-functional requirements in practice," in Proceedings of the 38th international conference on software engineering, 2016, pp. 832–842.

[34] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, and X. Franch, "Non-functional requirements documentation in agile software development: challenges and solution proposal," in International conference on product-focused software process improvement, Springer, 2017, pp. 515–522.

[35] J. Alvarado-Uribe, A. Y. Barrera-Animas, M. Gonzalez-Mendoza, A. L. Garcia-Gamboa, and N. Hernandez-Gress, "Towards a Standardized Evaluation of APIs Non-Functional Requirements Focused on Completeness and Soundness Qualities," Comput. y Sist., vol. 27, no. 4, pp. 889–897, 2023.

[36] N. Krishnamurthy and A. Saran, Building software: a practitioner's guide. Auerbach Publications, 2007.

[37] G. J. Singh, "10 Non-functional requirements for API design."

[38] A. C. da Silva Andrade, J. L. Braga, A. L. de Castro Leal, and F. H. Zaidan, "Risk management in software projects: an approach based on non-functional requirements," Sist. Gestão, vol. 14, no. 2, pp. 188–196, 2019.

[39] H. Ahmed, S. Zehra, U. Faseeha, H. J. Syed, and F. Samad, "Observability in Microservices: An In-Depth Exploration of Frameworks, Challenges, and Deployment Paradigms," IEEE Access, vol. 13, pp. 72011–72039, 2025.

[40] I. Gorton, L. Fong-Jones, and A. Larsson, "Observability Q&A," IEEE Softw., vol. 41, pp. 50–54, 2024.

[41] A. Chandrasehar, "The Role of Observability in Modern Software Development Lifecycle," Int. J. Sci. Res., 2021.

[42] B. Balakrishna, "Optimizing Observability: A Deep Dive into AWS Lambda Metrics," J. Artif. Intell. & Cloud Comput., 2022.

[43] N. Kratzke, "Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging - A Multi-Case Study," Futur. Internet, vol. 14, p. 274, 2022.

[44] B. Sharma and D. Nadig, "eBPF-Enhanced Complete Observability Solution for Cloud-native Microservices," ICC 2024 - IEEE Int. Conf. Commun., pp. 1980–1985, 2024.

[45] R. Dhall, "Designing Graceful Degradation in Software Systems," pp. 171–179, 2017.

[46] S. Chu, J. Koe, D. Garlan, and E. Kang, "Integrating Graceful Degradation and Recovery Through Requirement-Driven Adaptation," 2024 IEEE/ACM 19th Symp. Softw. Eng. Adapt. Self-Managing Syst., pp. 122–132, 2024.

[47] D. Kumar, A. Kumar, and L. Singh, "Non-functional requirements elicitation in agile base models," Webology, vol. 19, no. 1, pp. 1992–2018, 2022.

[48] A. A. Khan, J. A. Khan, M. A. Akbar, P. Zhou, and M. Fahmideh, "Insights into software development approaches: mining Q &A repositories," Empir. Softw. Eng., vol. 29, no. 1, p. 8, 2024.

[49] J. Zou, L. Xu, M. Yang, X. Zhang, and D. Yang, "Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis," Inf. Softw. Technol., vol. 84, pp. 19–32, 2017.

[50] I. Gorton, "Software quality attributes," Essent. Softw. Archit., pp. 23–39, 2006.

[51] D. Bermbach and E. Wittern, "Benchmarking web api quality," in Web Engineering: 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings 16, Springer, 2016, pp. 188–206.

[52] M. R. M. Assis and L. F. Bittencourt, "A survey on cloud federation architectures: Identifying functional and non-functional properties," J. Netw. Comput. Appl., vol. 72, pp. 51–71, 2016.

[53] A. Mahmoud, "An information theoretic approach for extracting and tracing non-functional requirements," in 2015 IEEE 23rd International Requirements Engineering Conference (RE), IEEE, 2015, pp. 36–45.

[54] B. De, "Api architecture trends in 2023," in API Management: An Architect's Guide to Developing and Managing APIs for Your Organization, Springer, 2023, pp. 385–411.

[55] L. Viviani, E. Guerra, J. Melegati, and X. Wang, "An empirical study about the instability and uncertainty of non-functional requirements," in International Conference on Agile Software Development, Springer Nature Switzerland Cham, 2023, pp. 77–93.

[56] S. Das, N. Deb, N. Chaki, and A. Cortesi, "Minimising conflicts among run‐time non‐functional requirements within DevOps," Syst. Eng., vol. 27, no. 1, pp. 177‐198, 2024.

[57] S. Santos, T. Pimentel, F. G. Rocha, and M. Soares, "Using Behavior-Driven Development (BDD) for Non-functional Requirements," 2024.

[58] A. Muhammad, A. Siddique, M. Mubasher, A. Aldweesh, and Q. N. Naveed, "Prioritizing non-functional requirements in agile process using multi criteria decision making analysis," IEEE Access, vol. 11, pp. 24631–24654, 2023.

[59] S. Rahy and J. M. Bass, "Managing non‐functional requirements in agile software development," IET Softw., vol. 16, no. 1, pp. 60‐72, 2022.

[60] H. M. A. El Sameaa, N. A. abd el Azim, and N. Ramadan, "Challenges of Non-functional Requirements Extraction in Agile Software Development using Machine Learning," Int. J. Comput. Appl., vol. 183, no. 43, pp. 23–26, 2021.

[61] M. Mathijssen, M. Overeem, and S. Jansen, "Identification of practices and capabilities in API management: a systematic literature review," arXiv Prepr. arXiv2006.10481, 2020.

[62] G. G. Martinez, Á. F. Del Carpio, and L. N. Gómez, "A model for detecting conflicts and dependencies in non-functional requirements using scenarios and use cases," in 2019 XLV Latin American Computing Conference (CLEI), IEEE, 2019, pp. 1–8.

[63] C. F. Castro et al., "Towards a conceptual framework for decomposing non-functional requirements of business process into quality of service attributes," in ICEIS 2019-Proceedings of the 21st International Conference on Enterprise Information Systems, SciTePress, 2019, pp. 481–492.

[64] E. Sherif, W. Helmy, and G. H. Galal-Edeen, "Proposed framework to manage non-functional requirements in agile," IEEE access, vol. 11, pp. 53995–54005, 2023.

[65] S. Pargaonkar, "A comprehensive review of performance testing methodologies and best practices: software quality engineering," Int. J. Sci. Res., vol. 12, no. 8, pp. 2008–2014, 2023.

[66] R. Koçi, X. Franch, P. Jovanovic, and A. Abelló, "Web api evolution patterns: A usage-driven approach," J. Syst. Softw., vol. 198, p. 111609, 2023.

[67] M. Lamothe, Y.-G. Guéhéneuc, and W. Shang, "A systematic review of API evolution literature," ACM Comput. Surv., vol. 54, no. 8, pp. 1–36, 2021.

[68] I. Rauf, E. Troubitsyna, and I. Porres, "A systematic mapping study of API usability evaluation methods," Comput. Sci. Rev., vol. 33, pp. 49–68, 2019.

[69] J. Scheibmeir and Y. Malaiya, "An API development model for digital twins," in 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, 2019, pp. 518–519.

[70] Y. Liu, A. Hamou-Lhadj, J. Li, and Q. Lu, "Observability and Explainability for Software Systems Decision Making," IEEE Softw., vol. 41, no. 1, pp. 45–49, 2023.

[71] M. Saunders, P. Lewis, and A. Thornhill, Research methods for business students. Pearson education, 2009.

[72] W. G. Zikmund, B. J. Babin, J. C. Carr, and M. Griffin, "Business Research Method 8th ed," 2010, Cengage Learning.

[73] D. F. Polit and C. T. Beck, "The content validity index: are you sure you know what's being reported? Critique and recommendations," Res. Nurs. Health, vol. 29, no. 5, pp. 489–497, 2006.

[74] M. Khan, M. A. Akbar, and J. Kasurinen, "Integrating Large Language Models in Software Engineering Education: A Pilot Study through GitHub Repositories Mining," arXiv Prepr. arXiv2509.04877, 2025.

[75] M. A. Hertzog, "Considerations in determining sample size for pilot studies," Res. Nurs. Health, vol. 31, no. 2, pp. 180–191, 2008.

[76] M. Tavakol and R. Dennick, "Making sense of Cronbach's alpha," Int. J. Med. Educ., vol. 2, p. 53, 2011.

[77] F. J. Fowler Jr, Survey research methods. Sage publications, 2013.

[78] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in software engineering, vol. 236. Springer, 2012.

[79] M. Kasunic, "Designing an effective survey," 2005, Citeseer.

[80] M. R. Lynn, "Determination and quantification of content validity," Nurs. Res., vol. 35, no. 6, pp. 382–386, 1986.

[81] A. A. S. Almohanna, K. T. Win, S. Meedya, and E. Vlahu-Gjorgievska, "Design and content validation of an instrument measuring user perception of the persuasive design principles in a breastfeeding mHealth app: A modified Delphi study," Int. J. Med. Inform., vol. 164, p. 104789, 2022.

[82] K. C. Pentapati, D. Chenna, V. S. Kumar, and N. Kumar, "Reliability generalization meta-analysis of Cronbach's alpha of the oral impacts on daily performance (OIDP) questionnaire," BMC Oral Health, vol. 25, no. 1, p. 220, 2025.

[83] H. N. Boone Jr and D. A. Boone, "Analyzing likert data," J. Ext., vol. 50, no. 2, p. 48, 2012.

[84] M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari, "Coupling metrics for predicting maintainability in service-oriented designs," in 2007 Australian Software Engineering Conference (ASWEC'07), IEEE, 2007, pp. 329–340.

[85] E. dos Santos and S. Casas, "An API Management Software Quality Metamodel based on Square and GQM," Eng. e soluç oes Ciência e Tecnol. para o Desenvolv. Hum., pp. 249–266, 2025.

[86] N. Dragoni et al., "Microservices: yesterday, today, and tomorrow," Present ulterior Softw. Eng., pp. 195–216, 2017.