Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Holistic Agile Improvement Framework: Embedded with Cost Estimation and Intelligent Requirements Classification Capabilities

Kamran Khan Tatari
Sarhad University of Science and Technology
Shahid Latif
Sarhad University of Science and Technology
Muhammad Ismail Mohmand
Muhammad.mohmand@atlas.edu.tr
Istanbul Atlas University

Article

Keywords: Cost Estimation, Holistic Agile Improvement Framework (HAIF), Machine learning models, Requirement Classification, Software Project Management

Posted Date: April 2nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8916845/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Holistic Agile Improvement Framework: Embedded with Cost Estimation and Intelligent Requirements Classification Capabilities

Kamran Khan Tatari[1], Shahid Latif[2] , Muhammad Ismail Mohmand[3]
[1]Sarhad University of Science and Information Technology, Peshawar, Ring Road Pakistan, Email: kamrankhan.tatari@gmail.com
[2]Department of Computer Science and IT, Sarhad University of Science and Technology, Peshawar, Pakistan, Email: shahid.csit@suit.edu.pk
[3]Department of Computer Engineering at the Faculty of Engineering and Natural Sciences at Istanbul Atlas University, 34408, Turkey, Email: Muhammad.mohmand@atlas.edu.tr

Abstract
The software requirement management is essential and crucial for the success of any software. There are various requirement management tools, framework and artifact used by different software companies based on their nature, context and size. There is no universal agreement among the software companies to adapt to a standardized framework. In this research we have not only proposed a Holistic Agile Improvement Framework (HAIF) but also validated the framework from the industry through survey and expert panel – Delphi Method. The framework is revised and updated by incorporating the recommendations from the agile experts. The framework claimed to be standard light weight customizable version of the available frameworks combining Scrum and Kanban elements align with agile principles. The framework can be used to evaluate and improve the existing process models in the field of requirements engineering and project management. In addition to the framework, serving as light-weight architectural roadmap; We have provided an extension by developing a tool using machine learning approach to address Project Cost scalability- Project Cost Estimation at feature and module level. furthermore, we have classified requirements (User Stories) into Functional and Non-Functional Requirement. The tool with dashboard provides two functional features of accurate cost estimation and requirements classification that are critical to the success of software projects. The findings indicate that framework with added features from machine learning models, combined with intuitive dashboards, can significantly improve software project management, offering a scalable and data-driven approach to cost estimation and requirement classification.
Key Words: Cost Estimation, Holistic Agile Improvement Framework (HAIF), Machine learning models, Requirement Classification, Software Project Management.

1. Introduction

Software project management is an essential and critical domain within the broader field of software engineering that plays a fundamental role in the successful execution of software projects [1]. In order to effectively manage requirements, the establishment of a framework is indispensable. There are many frameworks that are currently in used by various software development companies but its challenging to implement specific framework due to its lack of standardization and inadequate architecture [2]. Software development companies adapt to different frameworks and tools that are aligned with the nature , context and size of the company hence its challenging to adopt a standardised framework and architecture for the development of software [3-5] . Furthermore there are issues of lack of commitment and lack of effective project management methodology with existing frameworks that leads to governance challenges [6]. The

researchers are focusing on standardizatoin for effective requiremetns management to ensure the quality and success of the software projects [7]. Besides standardized framework that can play an effective role in managing requirements; central to the discipline of requirement management there are two key processes that needs careful attention; accurate cost estimation and precise requirement classification**.** These processes plays significant role by ensuring that projects are delivered on time, within budget, and in accordance with expectations of the client needs [8]. The Cost estimation provides financial information about projects that helps in efficient resource investment resulting in successful completion of project [9]. The cost estimation includes forecasting the amount of labor involved, development tools used, and testing of the resources that are needed for the entire project lifecycle. While in requirement classification the software requirements are classified into functional and nonfunctional categories that helps and guides the development, testing, and maintenance phases of the software project [10]. The functional requirements specify the expected services that are accomplished by the system, while non-functional requirements refers to quality characteristics of the software that address quality constraints such as usability , reliability , performance, security, and scalability [11].

The inaccuracies in cost estimation and requirements classification can lead to severe consequences for software projects. Eduardo Carbonera, et al. [12] stated that cost misestimation can lead to significant budget overruns, resource misallocation, and extended timelines that has ultimate impact on project delivery and client satisfaction. The improper and inaccurate classification of requirements have adverse effects on the project stakeholders resulting the development team focusing on the wrong priorities, causing deviations from client expectations, increased technical debt, and poor software performance [13]. Niazi, et al. [14] argue that to provide quality software, the implementation of reliable and efficient methods for accurate precisions of cost estimation and reliable classifications of requirements is essential.

In traditional software engineering the cost estimation relied on models such as Constructive Cost Model (COCOMO) [15] and Function Point Analysis (FPA) [16]. Where COCOMO is a parametric model that estimates project costs based on variables like software size and complexity, while as FPA measures size of the software through functional components that include inputs, outputs, and user interactions [17]. Both models' despite of their popularity and widespread depends heavily on expert judgment and historical data that leads to variability in their accuracy. Furthermore, the human biasness adds potential errors that can compromise the reliability of the project estimates [18]. Moreover, due to the changing and dynamic nature of the software projects, these models often fail to counter the dynamic nature of modern software projects as requirements and project scope may change during development [19].

Traditionally, the requirements classification is usually done manually which is a tedious and time consuming task [20]. In the process of requirements development (specification), requirements are documented typically in natural language, that prone to introduce ambiguities and variability in interpretation of the requirements [21]. These ambiguities make it challenging for stakeholders to accurately distinguish between functional and nonfunctional requirements where functional requirements describes the core functionality of the software while nonfunctional requirements refers to the quality attributes of the software like performance, security, usability, and scalability [22]. The imbalance and misclassifications in addressing both of these requirements can lead to resource misallocation and can potentially adverse consequences for project outcomes both in terms of functionality and quality [23].

The significance and impact on software projects has increased the demand for the use of automated, machine learning-driven approaches for the improvement of cost estimation and

requirement classification [24]. The use of Machine learning models have significantly enhanced the accuracy of these processes by learning from a large datasets and uncovering complex patterns in the data that may not be visible and obvious through traditional ways and methods [25]. These automated approaches not only reduce human error but also provide a scalable solution to the evolving needs of modern software projects [26].

In this study, we have proposed a standardized framework – Holistic Improvement Agile Framework (HAIF) that will served as serve as standard light weight version of the available frameworks align with agile principles. The Framework contains the governance layer mitigating the lack of committeemen issue and enhancing the project management methodology. We have validated the proposed framework through Delphi- Method [27]- expert opinion from local agile software companies that validated the appropriateness of the framework in agile software development environment. In second phase we have incorporated the recommendations of the agile expert and add some additional features. As functioning tool; we incorporated the application of advanced deep learning techniques—specifically, Convolutional Neural Networks (CNNs) for cost estimation and Long Short-Term Memory (LSTM) networks for the classification of functional and non-functional requirements. We opted for CNNs, because of its ability to extract hierarchical pattern from data- well-suited for analyzing complex project features such as lines of code (LOC), development time, and team structure, in order to predict the associated costs [28]. Meanwhile, LSTMs, which are a variant of Recurrent Neural Networks (RNNs), that is capable of handling sequential data, making it ideal for processing textual descriptions of software requirements and accurately classify it into functional or non-functional categories based on their semantic content [28].

This study aims to address challenges of lack of standardized agile frame work by proposing a standardized framework keeping the agile principles intact and also resolving the challenges of Project Cost Scalability – project cost estimation and Neglected non-functional Requirements by classifications of requirements using automation techniques through machine learning models. We have used a Convolutional Neural Network (CNN) regression model to estimate the software development cost based on various project features, such as lines of code, development time, and team composition. CNNs are known for its ability to capture spatial hierarchies in data, making them effective for regression tasks in complex, feature-rich datasets.

For the classification of Functional vs. Non-Functional Requirements (FNR), an LSTM-based model is proposed. Long Short-Term Memory (LSTM) networks which is a type of recurrent neural network (RNN) that are particularly used for handling sequential data, making it well-suited for analyzing textual software requirements [29]. The model process software requirements in natural language and classify them into functional and non-functional categories based on learned features.

The significant contributions of this research include:

- Proposed Agile architectural framework
- Validated the framework through empirical industrial evidence
- Integrated deep learning models (CNN and LSTM) within process architecture
- Introduced two domain specific datasets for Cost estimation and requirements c
- The introduction of two novel datasets specifically designed for cost estimation and requirements classification.
- Introduced Dashboards for better visualization and understandings.

The paper is organized into further sections briefly outlines as follow:

Section 2 reviews related to Frameworks in agile software development environment, software cost estimation and requirements classification. Section 3 proposed Agile framework (HAIF) and its validation and presents the datasets created for Cost Estimation and Requirements Classification, detailing their collection, preprocessing, and features. Section 4 outlines the methodology, including the architectures of the CNN regression model for cost estimation and the LSTM-based model for requirement classification. Section 5 reports the experimental results and performance evaluations of both models. Section 6 discusses the insights gained from the results, the challenges encountered, and the potential applications of this research. The Final Section 7 concludes the study and offers directions for future work.

1.1 Research Gap

Agile methodologies despite of its widespread adoption such as Scrum, there remains a significant lack of a standardized architectural framework that holistically integrates requirement management and scalable cost estimation within Agile software development environments. Furthermore, existing approaches largely focus on process practices while neglecting structured architectural integration, leading to fragmented implementations. Moreover, non-functional requirements are frequently neglected and overlooked due to the absence of automated classification mechanisms, resulting in incomplete requirement coverage. The traditional estimation models such as COCOMO primarily operate at the project level and do not adequately address feature- or module-level cost scalability required in iterative Agile environment. Additionally, the application of deep learning techniques for requirement classification and cost estimation remains isolated rather than embedded within a unified framework, and there is a shortage of domain-specific datasets tailored to these tasks. Finally, many proposed Agile improvement frameworks lack empirical validation from industry experts, highlighting the need for a standardized, AI-integrated, and industry-validated Holistic Agile Improvement Framework (HAIF).

1.2 Research Questions

The following research questions are intended to address during this research

RQ1. How can a Holistic Agile Improvement Framework (HAIF) be designed as standardized architectural framework integrating the requirement management and scalable cost estimation?

RQ2. How effectively an LSTM based model automatically classify software requirement into functional and non-functional requirements?

RQ3. How accurately can a CNN have based regression model estimate software development cost at feature and module level using project specific attributes?

RQ4. How do the newly developed datasets improved model reliability and predictive performance?

2. Related Work

Agile paradigm is dominating from the last two decades; in spite of its popularity and wide spread adoptions there are still many challenges [30]. Among many other challenges; some challenges include lack of standardized framework, Inadequate architecture, overall project cost scalability,

and neglected nonfunctional requirements [31, 32]. The use of machine learning approaches in the software engineering brought further efficiency in requirements management[33].

2.1. Agile Framework

The software development approach has turned into new approach where plan driven heavyweight processes transformed into to light weight agile approach [34]. Unlike traditional approaches like waterfall, agile has high successful completion rate as 40 % compared to waterfall model which is 15% [35] . There are many methods used under the agility nature that includes XP, Scrum, DSDM and FDD but scrum is the most trendy and popular component of agile [36-38]. The software projects for its better management and quality deliverance usually completed under certain process models and methodology that provides exclusive predefined rules and steps where framework provides an overall guideline providing flexibilities for managing the projects. A framework according to Cambridge Dictionary (2025) refers to 'a system of rules, ideas or beliefs that is used to plan or decide or a framework is a supporting structure around which something can be build. The frameworks provide an additional support for software development team to effectively deliver the software. Frameworks can be used to improve the existing requirement engineering models with value added delivery to the customer [39]. Goodpasture [40] in his book 'Project Management the Agile Way' mentioned that Scrum is a management framework and among other methods it's the simplest and most popular one. While Adam and Maria [41] called Agile as de-facto Framework for the software development. Researchers adopted frameworks to improve the process of software engineering models and its management. one such effort is done by Schön [39] by proposing framework for modeling agile requirement engineering through user centric approach. Agile despite of its popularity still faces various challenges including lack of standardized framework, Inadequate architecture, neglected nonfunctional requirements and overall project cost scalability [42-45]. To the extent of our studies we have depicted the Agile way of methodology as in Feg.1; where agile principles serve as guidelines and facilitate the development process

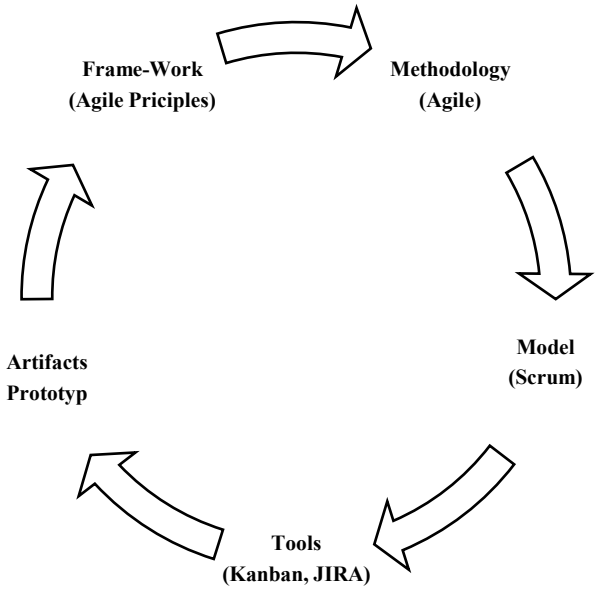

Fig. 1. Agile based Development Process

In this research we have attempted to improve the agile scrum methodology by proposing a framework (HAIF) that will serve as meta model providing guidelines for standardized architecture. The framework is further supported by the development of tools that can mitigate the agile challenge of neglecting non-functional requirements through its classification and prioritization and also resolving cost scalability issue by proposing cost estimation tool.

2.2. Cost Estimation Models in Software Engineering

One of the critical part of software development is its cost estimation because it determines whether the software has surpassed certain threshold- line of codes, time and budgets leading to acceptance or abandoned of software [46]. Over the years, various approaches have been developed for cost estimation in software engineering, each attempting to provide more accurate predictions for resource allocation and financial planning in software projects [47] but there is no single strategy that works well for effective cost estimation [48]. Early methods, such as the Constructive Cost Model (COCOMO) [49] and Function Point Analysis (FPA), have been widely used [50]. These parametric models rely on expert judgment and predefined metrics, making them accessible but often prone to inaccuracies, especially in the case of evolving project scopes or ambiguous initial requirements [51]. Despiteful of the usefulness of the traditional models; face significant limitations due to its dependency on historical data and human interpretation [52].

In order to tackle these limitations, machine learning techniques have been introduced to automate and enhance the cost estimation process. There are many models for cost estimations where Regression-based models are one of the popular choices for cost estimation that uses historical project data to predict future project costs. One of the pioneer model was linear regression models that were first applied to this problem that has provided an initial step toward automating the estimation process [53]. Although, linear models tend to oversimplify the complexities of software projects, but are influenced by many non-linear factors such as team dynamics, changing requirements, and evolving technologies in today's modern era [54].

There are many other sophisticated models that can be utilized to address these complexities, such as decision trees, random forests, and support vector machines. These models are capable to capture non-linear relationships between variables by providing more nuanced estimates [55]. Despite of the fact that these approaches improve over simpler models, they are still limited in handling very large datasets or capturing deep hierarchical structures that are inherent in the project attributes [56].

There are different machine learning models for cost estimations. The Convolutional Neural Networks (CNNs), have gained distinction and popularity in cost estimation tasks [52]. In order to process complex and feature-rich datasets, the CNNs is used due to its ability to detect hierarchical patterns [57]. The application of CNNs in cost estimation allow the models to learn from a variety of input features such as lines of code (LOC), development time, team size, and complexity [58]. These models have shown clear distinction in improving accuracy over traditional machine learning methods. The comparative studies between CNNs to traditional models have demonstrated significant performance improvements, especially in projects with large and diverse datasets giving edge to CNNs over traditional models [59].

There are other recurrent models especially for those projects that have evolving requirements, include Long Short-Term Memory (LSTM) networks that explore cost estimation over time, particularly for projects having evolving requirements [29]. The LSTMs are designed to manage and handle sequential data effectively in capturing temporal dependencies, making them useful in

projects where cost estimations need to be adjusted for the entire Lifecyle of the project [60]. The combination of both CNNs and LSTMs has also been investigated, with the former handling spatial features and the latter managing temporal aspects, leading to more comprehensive cost estimation frameworks for the projects [52] .

2.3. Requirements Classification Approaches

Requirements analysis and Classification is indispensable for the assessment and successful deployment of the project. The Software requirements are usually split into two classes that is functional and non- functional requirement software. The functional requirements refer to specific behavior or function of the system while Non-functional requirements refers to the system characteristics such as performance, usability, scalability and reliability. The nonfunctional requirements are usually neglected as compare to functional requirements but plays a significant role in the quality and acceptance of Software [61-63]. Classifications of requirements into functional and non-functional categories is a key task in project planning, and a variety of approaches have been developed to automate this process [64]

According to Saleem, et al. [65] the early techniques involved rule-based systems, where predefined rules were applied to requirement documents to differentiate between functional and non-functional requirements. While rule-based systems are straightforward and interpretable, they suffer from limited scalability and flexibility, especially when dealing with large, unstructured requirement documents [66].

It is observed that with the growth of natural language processing (NLP), more advanced techniques have been developed to improve the accuracy and scalability of requirement classification [67]. One of the earliest applications of machine learning to this task involved using Naive Bayes classifiers to process text-based requirement documents [68]. In order to distinguished between functional and nonfunctional requirements the classifiers were trained on labeled data, learning to distinguish based on word frequencies and patterns [20]. The Naive Bayes models despite of its simplicity provides a significant improvement over rule-based systems in handling particularly diverse requirement documents [69].

Keeping the challenges in requirements classifications, more sophisticated NLP techniques such as support vector machines (SVMs) and decision trees were applied to classify software requirements [70]. These models are specialized in handling a broader range of features, including semantic structures, providing better accuracy compared to Naive Bayes models [52]. Despite of the performance of these models, their performance is often limited by the quality of the feature engineering process, which requires domain expertise and manual effort to analyze and extract relevant features from the given text [71].

The researchers have diverted their attentions towards learning approaches, particularly recurrent neural networks (RNNs) and their variants, for exploring in the domain of requirement classification. More specifically the RNNs and LSTMs both, are particularly suited for requirement classifications due to their ability to capture sequential patterns in textual data [72]. The research studies revealed that LSTMs outperform traditional machine learning models in classifying software requirements, as they can better capture the context and dependencies within the textual data [73].

There are pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers), used to enhance the requirements by classification into functional and non-functional requirements [74]. In order to be fine-tuned on domain specific datasets with limited labeled data such models provide leverage in transfer learning. According to Kaur and Kaur [75]

the applications of such models have significantly improved the classification accuracy, particularly in cases where requirement documents are ambiguous or where functional and non-functional requirements overlap each other. Numerous studies have focused on combining both NLP and deep learning models as hybrid approach. Kaur and Sharma [76] applied and proposed such hybrid approach for accurate classification use feature extraction techniques TF-IDF (Term Frequency-Inverse Document Frequency) alongside LSTMs or transformers. Khan, et al. [77] added that these models have significant edge from the interpretability of traditional NLP features while leveraging the strength of machine learning for sequence processing.

2.4. Machine Learning and Deep Learning in Software Project Management

The Machine Learning (ML) is widely used in different disciplines and domains ranging from laboratories to frameworks, tools and other development system [78]. The researchers have diverted their attention in integrating machine learning models into software project management. There is a particular realization of using machine learning in agile methodologies [79]. A vast number of domains have applied machine learning especially to various aspects of project management that includes risk assessment, effort estimation, and task scheduling etc. Traditional approaches in this area have often relied on expert judgment and historical data, leading to variability in predictions [80]. While machine learning in managing and engineering requirements is very useful compared to other manual methods [81].

All requirements are differently analyzed and different methods are used but they are time consuming compared to machine learning [81]. Regression models is one of the model that has been commonly used for effort estimation, particularly linear regression, support vector regression, and decision tree regression [82]. These models have been successful in capturing linear relationships between project attributes, such as team size and complexity, and project outcomes [72]. However, they tend to struggle with the non-linear and hierarchical nature of software projects, leading to suboptimal predictions in complex projects.

To address these limitations, deep learning models have been explored as an alternative. The CNNs, model has the ability to capture hierarchical patterns, furthermore it has been successfully applied for effort estimation tasks, especially in those projects that have large datasets. It is evident from the studies that CNNs outperform traditional machine learning models, particularly in capturing non-linear relationships between project attributes [83].

Similarly; the recurrent models such as LSTMs in particularly, have also been applied to tasks like risk assessment and task scheduling, where temporal dependencies play a critical role. While predicting the risks of the projects, LSTMs have been shown to perform well in predicting the evolution of project risks over time, providing project managers with valuable insights into potential future challenges [84].

The researchers have diverted their focus to attention-based models, such as transformers, have been explored in software project management. Whereas these models, originally developed for NLP tasks; have shown excellent performance in capturing dependencies between different aspects of a project, such as requirements, timelines, and risks. Studies applying transformers to software project management have shown promising results, particularly in requirement classification and effort estimation [85].

Transfer learning is one of the a techniques to improve the performance of machine learning models in software project management [86]. The Pre-trained models, particularly those trained on large and general datasets, can be fine-tuned on domain-specific datasets, providing significant improvements in accuracy without the need for extensive labeled data [87].

Overall, the application of machine learning to software project management has shown considerable promise of providing automated solutions to tasks that will traditionally be reliant on expert judgment. However, challenges remain, particularly in terms of interpretability and scalability. The black-box nature of many machine learning models poses challenges for project managers who are in need to understand the rationale behind predictions to make informed decisions.

3. Methodology

This study adapted the constructive research paradigm where we have analyzed and interpreted the existing literature to identify patterns and themes pertaining to the role and importance of framework in software engineering. We have narrow down our study to explicitly analyze the role of frameworks, its needs and importance in agile based requirement management. Based on our qualitative study followed by inductive approach; we have proposed a framework based on specific observations (literature review). Through literature the following three gaps were identified and provided based for motivation of this research.

a) Lack of standardized framework and Inadequate architecture
b) Nonfunctional requirements are neglected and lack of its automatic classifications
c) Less work done on Project Cost scalability – Cost estimation at feature and module level

Through inductive approach, we have proposed Holistic Agile Improvement Framework as shown in Fig.2. We have validated the framework for its appropriateness through Delphi Method (Expert Opinion); The model was presented to two group panel of agile experts that were comprised of Product Owner, Scrum Master and development team. The two agile companies Exceed Solutions (Peshawar, Pakistan) and Tech-Arete (Islamabad, Pakistan) participated in the validation process. The Validation was administered through survey approach where both closed/open ended questionnaire was distributed along with interviews with the agile teams of both companies. The agile companies were engaged for validation of the agile framework and data collections for design and development of Data-set for cost estimation and classifications of requirements. The survey was categorized into two phases. In the first phase the agile framework was presented to the group of agile experts and through Interview pre-planned questions the appropriateness of the proposed framework was validated. The questions asked for instance ‘what framework you have followed or following for your software development? ‘Do you consider the framework is mandatory or optional?’ ‘Do you think that following a framework can make development more efficient? ‘Have you ever experienced any difficulty before or after following a framework?’ ‘is the presented framework is adhered to the agile principles?’ and so forth. In the second phase the proposed framework was revised based on the recommendations and added new functionalities. The revised framework was added with two functionalities; that is cost estimation and classifications of requirements. Additionally, visual dashboards were developed for better clarity and understandability. The dataset for cost estimation and requirements classifications was collected from the selected agile companies on the basis of which these two functions were added. This addition of functionalities has provided edge to the proposed framework from theoretical architecture to quantifiable solutions for cost estimation and classification of requirements. The details are given in further sections and sub sections.

3.1. Proposed Holistic Agile Improvement Framework

We proposed framework through a structured approach to analyze, design, and implement solutions tailored to agile environment specially to overcome the issue of lack of commitment and effective project management. Though agile works arrounds its principles and Manifesto; many studies indicates the importance of the people in agile environment that leads to success or failiure of the projects [88-90]. In the context of One of the core and important aspect of agile is people therefore sharing responsibilities is critical and important for the success of a project; since agile environment work is done through a constant collaborative team work; there are chances of confusion amond agile team to make someone responsible and accountable. To enhance the effectivity and productivity of agile team with assigned roles, we have adapted the project management matrix called RACI (Responsible, Accountable, Consulted and Informed) [91] and incorporated RACI as governing layer in the agile framework. Reseachers are of the opionion that it always better to go for hybrid appraoch where agile methodology is combine with other frameworks like Scrum components combined with Kanban [92-94]. The propostion of the theortical framework is essential that can contexutalize the solutions that can map the information flow and focus on usablity parameters [95]. We have contexutualzie the agile issues of neglected non funcitonal with more managebile tasks we merged RACI matrix and Kanban into our proposed framework for its better goverance and visualiztion respectively. The framework not only served as meta-model that will served as architectural raodmap for imprivements in the current and future framewokrs. Additionally the proposed framework. provides two functioning tool for cost estimation and requiremetns classification which are the most common issues in the agile based development.

We have added the follwoign features in our proposed agile framework (HAIF).

- RACI Metrix for better governance – addressing the issue of lack of commitment and effective project management
- Combining Scrum and Kanban- Adapted Hybrid approach that brings further flexibility and effeciency.
- Automatic Classification of Requirements into functional and non-functional requirements – resovling issue of neglected non functional requirements; enhancing productivity and effeciency of the development team
- Auto Cost estimation at feature and module level- resovling the cost scalaiblity issue with effeciency and better cost estimatin management.

The Holistic Agile Improvement Framework (HAIF) will serve as Adaptable- to all team members, Scalable- from small to medium level deployment, Measurable – with clear metrics and improvement cycle, Actionable- with data-driven retrospectives and governance feedback (ASMA), Fig. 2. Depicts the proposed framework (HAIF)- Pre-validated Version.

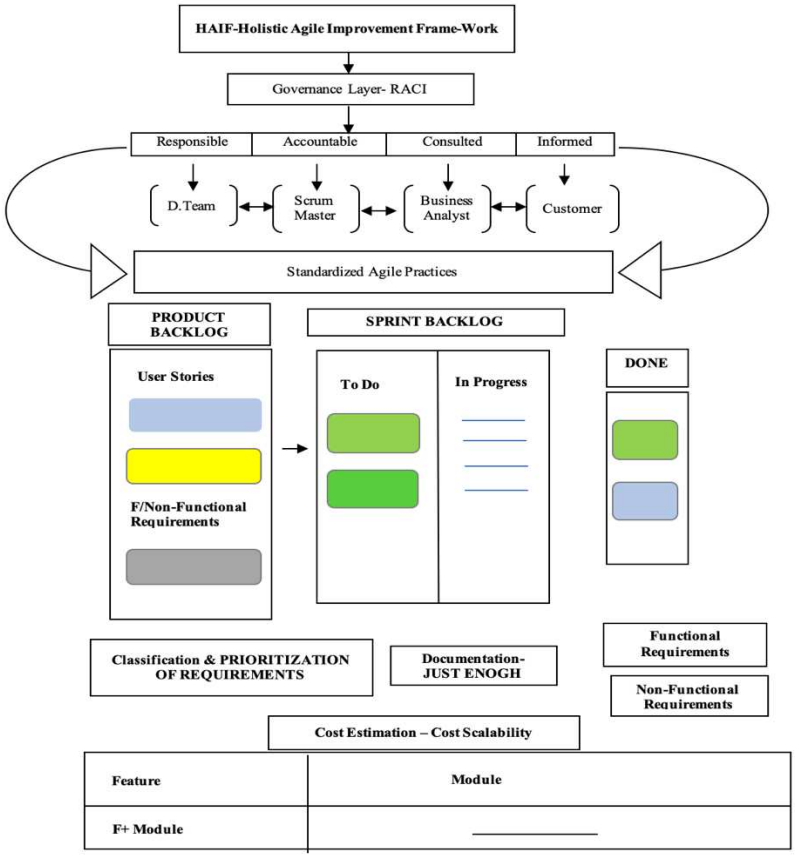

Fig. 2 Proposed Holistic Agile Improvement Framework - HAIF

The integrations of project management aspects such as RACI according to the modern directions for digital innovation in project management [96] that can be related to the context of our project management framework will help the individuals (agile stakeholder) help in developing their soft skills such as collaboration, flexibility integrations and making boundaries for spanning team work

3.1.1 Validation of HAIF

To validate our proposed frame-work we have adapted the Delphi method. This method is one of the suitable data gathering tools for collecting requirements for a new system used for improving new processes in software engineering. [27]. We have performed the validation process in the following steps

1. Development of questionnaire for presenting our proposed framework to the experts and gathered the data on evaluation points (role specific and general). Further supported by online survey using google-form
2. Performed 2–3 iterative rounds of anonymous expert reviews using Likert-scale + open feedback.
3. Refine the framework after each round, sharing aggregated feedback
4. Track consensus levels and use statistical convergence to finalize the framework's appropriateness in the context of agile and tabulated the results.

In the validation process via Delphi method, two panels of expert (15 experts in each panel from two reputable agile companies) participated in the validation of the framework. The panel of experts were first briefed about the purpose of the survey. It was ensured that participants participating in the survey are Expert – having more than 10 years of experience in software development particularly in agile/extreme programming. The experts were shown the proposed framework and were asked about any ambiguity of understanding framework as whole and regarding the concepts. The validation process is performed in three rounds.
Round 1. The full questionnaire s hared with panel of 15 agile experts from each company.
Round 2. Share anonymized aggregated results from Round 1 and ask experts to revise their opinions.
Round 3. Analyzed and extracted the results to reach final consensus.
Questionnaire Design: The questionnaire was designed and classified into following sections:

- Section A: General Evaluation
- Section B: Product Owner
- Section C: Scrum Master
- Section D: Development Team
- Section E: QA / Tester
- Section F: Final Feedback

For each question we have used Linear Scale (1–5 Likert scale) questions followed by short answer or paragraph for the "Comments" as open-ended questions. General and Final feedback section were mandatory that were to be answered by all the participant (experts) while for section B-E. The experts were given options to jump directly to the questions related to their area of expertise. The experts were allowed to give their expert opinion in other sections in case he/she has multiple expertise. Parallel to the Interviews; an online survey was also conducted where more than 30 experts were selected and contacted out of which 15 experts responded and participated in the survey. The online survey parallel to interviews were conducted to get more broad and generalized view about the validations to incorporate in final convergence round to reach to final consensus. We analyzed and classified expert opinion separately depending on their role (section B-E). The results of the expert are summarized in the Table 1 and Table2.

Table 1. Delphi Evaluation Expert Opinion Results

Expert	HAIF	Remarks	Suggestions
Product Owner	RACI	Recommended as it clearly defines stakeholders' roles (Responsible, Accountable, Consulted, Informed), which aids in stakeholder alignment and mitigates ambiguity in decision-making.	The framework could include an impact mapping technique or value-based prioritization model to better align features with business value. Consider adding a feedback loop from the customer directly into backlog refinement.
	Backlog Management	Visualization of user stories and requirements enhances backlog grooming.	
	Auto Classification of Requirements	Helps ensure non-functional requirements (often overlooked) are integrated into user stories—boosts product quality and delivery consistency	

Scrum Master	HAIF over all	Promotes process consistency across teams	Add metrics dashboards (velocity, throughput, cycle time) to help Scrum Masters track team health and process bottlenecks. Specify how retrospective insights feed back into the framework Mention how technical debt and spikes (R&D tasks) are handled within the backlog. Include support for DevOps or CI/CD practices to better integrate development and delivery
	Hybrid Agile (Scrum + Kanban):	Flexible delivery—Scrum for planning and ceremonies; Kanban for visual workflow tracking	
	Governance Layer	Helps Scrum Masters ensure team roles and responsibilities are adhered to, supporting scaled agile practices	
Development Team	Sprint Backlog View	Clear status (To Do → In Progress → Done).	
	Classification of Requirements:	Enables developers to understand critical backend (non-functional) concerns early—performance, security, etc	
	Auto Cost Estimation	Helps team foresee effort and complexity before committing to work.	
Tester/QA	Functional /Nonfunctional Requirements	Emphasis on functional and non-functional requirements directly supports Test Case Design.	It's unclear how test automation, acceptance criteria, and Definition of Done are enforced—these could be layered under Standardized Agile Practices. Include a feedback loop for regression issues and production bugs to reinforce learning
	Auto Classification of Requirements	Automated classification helps testers create test plans early in the cycle.	
	Inclusion in the sprint cycle ensures continuous validation		

Table 2. Summarized Consolidated Expert Views:

Evaluation Criteria	Verdict
RACI Governance Layer	Strong alignment with Agile scaling and accountability
Hybrid Scrum-Kanban Approach	Practical for most Agile teams enhance flexibility
Auto Requirement Classification	Great addition for productivity; often missing in traditional Scrum
Auto Cost Estimation (Feature/Module)	Helps manage scope creep and supports Lean Budgeting resolving the cost scalability issue

Recommendations	
Dashboard	Add Functionable Cost Estimation Dashboard and Classification Dash Boards for better visualizations and understanding
Prioritization	Include impact mapping technique or value-based prioritization model to better align features with business value
Customer Feedback Integration	Needs more explicit detail
Retrospective Feedback Mechanism	Recommend a closed-loop learning process
DevOps Integration	Not clearly visible – a gap to address for modern Agile practices
Test Automation & QA Practices	Light mention – can be improved

The HAIF framework is validated from its appropriateness and suitability in the context of agile environment based on the Delphi method hence one of the objectives of this research is achieved. The following features are added in the revised proposed model. The proposed model HAIF validated by the agile based companies through expert opinion which was one of the objectives of this research. In addition to the appropriateness of the model in the agile environment; the HAIF has been revised with certain new features added based on the recommendation- expert opinion. The new added features are as following:

1. Added value-based prioritization model -MoScoW
2. Developed Dash-Boards for Project Cost Estimation and Requirement Classification.

The proposed framework is revised by adding functionality of Cost estimation and requirements classifications. The details of the added functionality and dashboards are explained in the further sections.

4. Cost Estimation

Cost estimation and cost scalability is one of the significant concerns for the software project. The well estimated software project brings various significant including effective resource management. Unlike traditional methods, such as the Constructive Cost Model (COCOMO) and Function Point Analysis (FPA) that often rely on expert judgment and historical data, leading to variability in accuracy. To add functionality to our proposed model HAIF, we have used advanced deep learning models that is Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for improving the precision of software cost estimation. To estimate the project cost based on key features such as lines of code, development time and team size, we employed CNN model that has achieved an R-squared value of 0.85. while an LSTM is used for classification due to its model suitability for sequential data and classification of requirements. For cost estimation as its next step, we required to develop the data by adding features and attributes. The details are given in the subsections.

4.1 Data Set Collection Process

We have collected the data for software cost estimation from various software houses located in *Peshawar and Islamabad City, Pakistan.* The companies were engaged for prior consent regarding the relevant data and information. After the consent, the companies have provided detailed project information spanning both past and ongoing software projects with varying size, complexity and

domain, offering a diverse range of data points for our analysis. The collected datasets contained key project attributes that are significant and critical for predicting the development costs. The primary sources our collected data were included project management tools, time-tracking software, interviews with project managers and team leads. The dataset collected for cost estimation were based on various software projects domains including mobile applications, web applications, and large enterprise level systems. The model inputs contained the following project variables extracted as input for our model. The total sample of the data set for cost estimation is 12 hundred instances where as 960 samples are used as model training data while 240 data sample were used for model validations. The dataset is consisting of nine attributes where as eight features are used as model inputs while the last one cost estimation is the output of the model. Below are the details of the input/output features.

- **Lines of Code (LOC):** The Total number of lines of code written for each project.
- **Development Time:** The duration of the project, measured in months.
- **Team Size:** The total number of developers, testers, and management personnel involved in the project.
- **Project Complexity:** At a scale of 1-10 a subjective rating provided by the project team, indicating the perceived complexity of the project.
- **Customer Involvement:** The Frequency and depth of customer interaction during the development process.
- **Project Type:** The category of the project that includes mobile application, web applications, and enterprise level systems.
- **Risk Level:** The anticipated risk, assessed by the project teams, classified as low, medium, or high.
- **Tools and Technologies Used:** The specific technologies, frameworks, and tools applied in the development process.
- **Final Project Cost (USD):** The actual monetary cost required to complete the project.

4.1.1 Features and Attributes

We have extracted the key features and attributes for cost estimation based on relevance and impact on project cost. The details of the key features, its description and type are summarized in table 3.

Table 3: Key features and attributes in the cost estimation dataset collected from software companies.

Feature	Description	Type
Lines of Code (LOC)	Total number of lines of code for project	Numeric
Development Time	Time taken to complete the project (in months)	Numeric
Team Size	Number of developers, testers, and managers involved	Numeric
Project Complexity	Subjective rating of complexity (scale of 1 to 10)	Ordinal
Customer Involvement	Frequency of customer communication and feedback	Categorical
Project Type	Type of project (web, mobile, enterprise, etc.)	Categorical
Risk Level	Estimated risk of the project (low, medium, high)	Ordinal
Tools and Technologies	Technologies and development tools used	Categorical
Final Project Cost (USD)	Actual cost of the project in US dollars	Numeric

The Project Cost Estimation Dashboard as Figure 3. streamlines cost prediction by allowing users to input critical project variables such as LOC, team size, and complexity. The system then generates an estimated project cost, enhancing project planning efficiency.

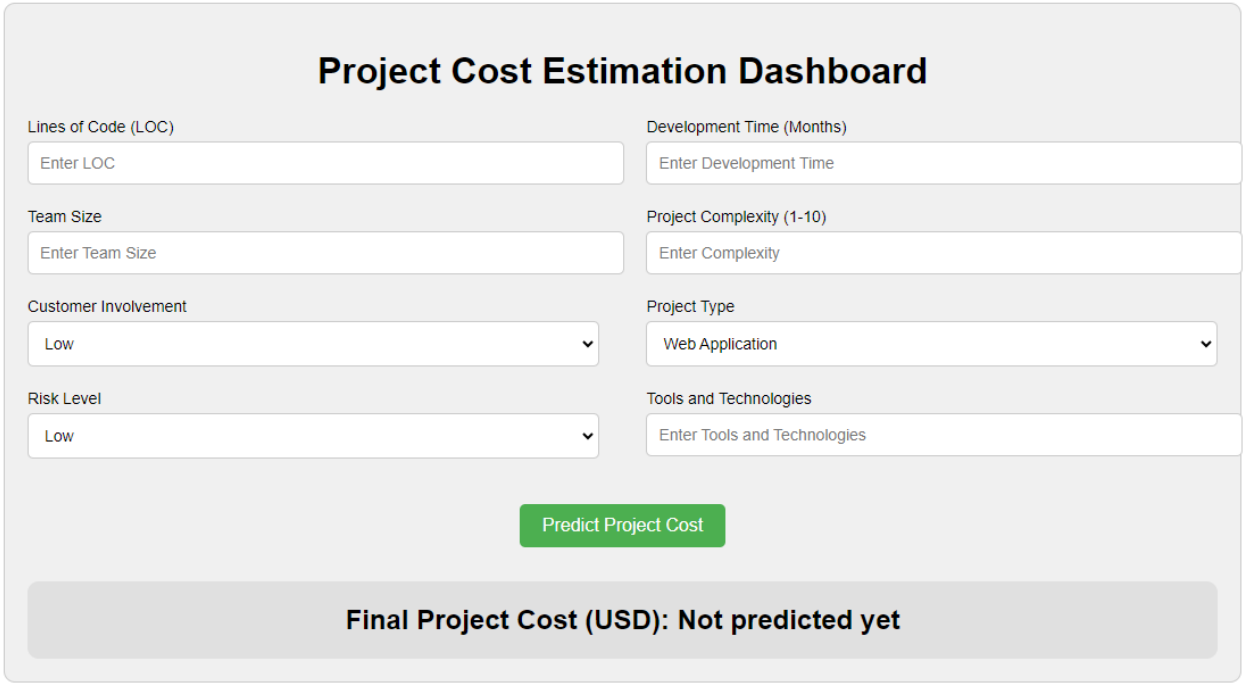

Figure 3: Project Cost Estimation Dashboard allowing users to input project parameters to predict final project costs in USD.

4.1.2 Data Preprocessing and Preparation

Once the data was collected, several preprocessing steps were performed to ensure it was ready for training the CNN regression model. The table 4 describes the details about preprocessing along with its examples

Table 4: Overview of preprocessing steps including missing data handling, normalization, encoding, outlier detection, and dataset splitting.

Preprocessing Step	Description	Example
Handling Missing Data	Missing or incomplete data handled by imputing missing values (median for numerical fields, most frequent for categorical) or excluding projects with substantial missing data.	If the team size for a project is missing, the median team size across other projects is used.
Normalization	Numeric variables (e.g., LOC, development time, team size) were normalized using min-max scaling to ensure all features contribute equally during CNN model training.	LOC (ranging from 1,000 to 50,000 lines) normalized to a scale between 0 and 1.
Categorical Encoding	Categorical variables, such as project type, customer involvement, and tools used, were converted to binary vectors using one-hot	A project type such as "web" becomes (1) and "mobile" becomes (0, 1, 0).

	encoding for compatibility with the CNN model.	
Outlier Detection	Outliers in the dataset identified and removed based on interquartile range (IQR), particularly extreme values in terms of development time or project cost.	Projects with a cost above $1,0000, when the typical range is between $10,00 and $100,00.
Dataset Splitting	The dataset was divided into 80% training and 20% testing sets, ensuring a balanced distribution of project types, complexity, and risk levels.	100 projects are split into 80 for training and 20 for testing, maintaining proportional complexity.

5. Requirements Classification

There are various types of requirements from the perspective of different stakeholders in the project. The requirements can be of project, system, business or user requirements and so on. In order to design solutions, the developers use the functional and nonfunctional requirements to design solutions that implement the necessary functionality in the developing software under the given constraints. In the context of this research; while developing requirements, the user requirements (user stories) from verbal to text and then text into classification into functional and non-functional requirements brings efficiency and innovation. The details are given in the following sections. The total samples for the requirement classifications are 600 which are split at a ratio of 80:20. The 80% data sample were used for training while 20% of the remaining are used for model validation.

5.1 Requirement Collection and Text Conversion

The second dataset focuses on the classification of requirements into functional and non-functional categories. The collection of requirements was carried out by extracting requirement documents from several software projects developed by the companies in Peshawar and Islamabad. These requirement documents included a variety of user stories, technical specifications, and design documents that describe the functionality and performance criteria of the software being developed. The visual illustration of the requirement classification is depicted in Figure 4.

The requirements were collected in their raw form, typically written in natural language. To prepare the dataset for classification, these raw requirements were first converted into machine-readable text format. To ensure that the dataset contains clean and consistent textual data for further processing we performed conversions. The process of conversion involved removing any formatting, special characters, or irrelevant details like version numbers or dates.

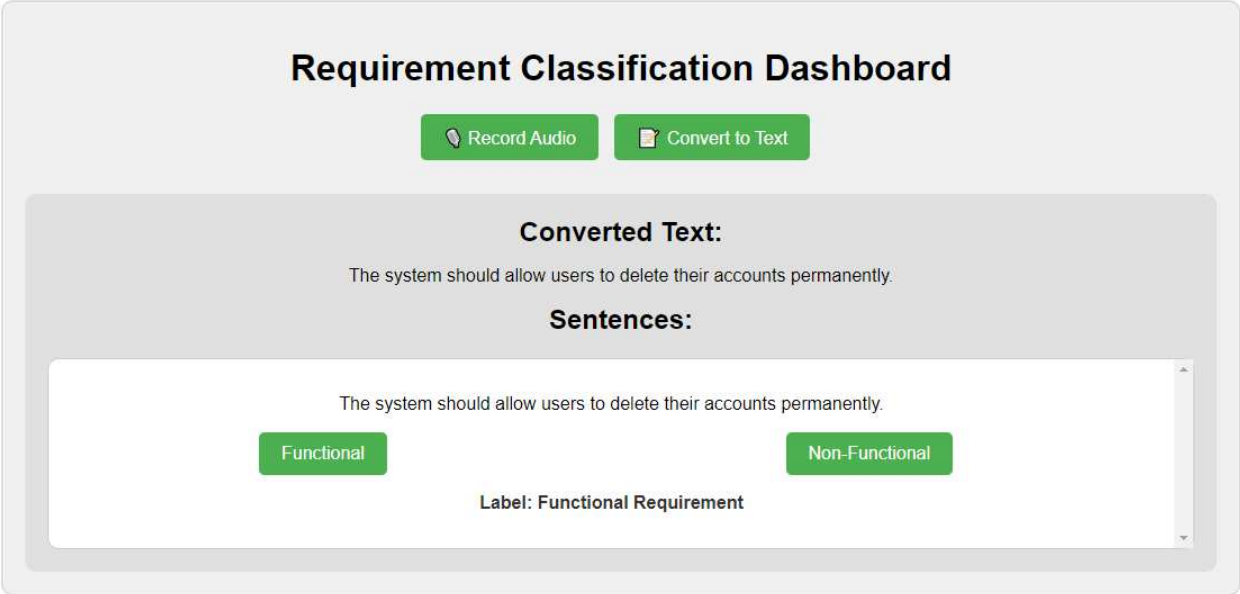

Figure 4: Requirement Classification Labelling Dashboard

5.1.1 Sentence Segmentation and Keyword Extraction

The segmentation of the document into individual sentence is performed after conversion of the requirements into clean text. We perform sentence segmentation to isolate each functional or non-functional requirement, as many of these documents contained multiple requirements within a single paragraph.

After segmentation the key word extraction was applied to each sentence to identify key terms related to functionality (e.g., "login", "submit", "process") or non-functional attributes (e.g., "performance", "security", "scalability"). The key extraction was done using a combination of Term Frequency-Inverse Document Frequency (TF-IDF) and Named Entity Recognition (NER) techniques, where the selection of keywords was based on their relevance to the functional or non-functional nature of the requirement. The table 5 and table 6 shows the segmentations and extraction of the keywords and its labeling respectively.

Table 5: Example of Sentence Segmentation and Keyword Extraction.

Original Text	Segmented Sentence	Keywords Extracted
The system must allow users to log in and submit their orders securely.	The system must allow users to log in.	login, users
	The system must allow users to submit their orders securely.	submit, orders, securely
The system must allow users to submit their orders securely.	The application should perform efficiently under high user load.	perform, efficiently, high user load
	The application should be scalable.	scalable

5.1.2 Labeling of Functional and Non-Functional Requirements

After this systematic and structured approach to requirement collection, segmentation, keyword extraction, and labeling, the dataset became ready for training and evaluating the models for automated classification requirements into functional and non-functional requirements. See table 6 for labeling of requirements with examples.

Table 6: Example of Labeled Functional and Non-Functional Requirements

Segmented Sentence	Keywords Extracted	Label
"The system must allow users to log in."	login, users	Functional Requirement
"The system must allow users to submit their orders securely."	submit, orders, securely	Functional Requirement
"The application should perform efficiently under high user load."	perform, efficiently, high user load	Non-Functional Requirement
"The application should be scalable."	scalable	Non-Functional Requirement

6. CNN Regression Model Architecture

As for as learning from complex patterns from a variety of projects; the Convolutional Neural Network (CNN) regression model used for cost estimation is designed to learn complex patterns from attributes such as lines of code (LOC), development time, team size, and project complexity. The CNNs are well suited to accurately predict project costs due to its ability to extract spatial hierarchies and patterns from the input data, helping in accurate project cost prediction. The details of the layers its shapes and descriptions are given in table 7 as model layer overview.

Input Layer

The matrix of project features served as input to the CNN model where each project is represented by a vector of numerical and categorical attributes. The project type and customer involvement as Categorical variables are one-hot encoded, and numerical features are normalized to ensure all input features are on a similar scale. The input layer processes this matrix of shape as (batch_size, num_features).

Convolutional Layers

In convolutional layer to learn meaningful relationship between attributes, the architecture employs two 1D convolutional layers that apply filters over the project feature vectors

Layer 1 (Conv1D):

- **Filter size:** 64
- **Kernel size:** 3
- **Activation:** ReLU (Rectified Linear Unit)
- **Stride:** 1
- **Padding:** Same
- **Output Shape:** (batch_size, num_features, 64)
- **Layer 2 (Conv1D)**:
    - **Filter size:** 128
    - **Kernel size:** 3
    - **Activation:** ReLU
    - **Stride:** 1
    - **Padding:** Same
    - **Output Shape:** (batch_size, num_features, 128)

Pooling Layer

In the pooling layer, we have applied max-pooling layer to reduce the size of the feature maps and to extract the most important information from the convolutional layers.

- **Max Pooling**:
    - **Pool size:** 2
    - **Stride:** 2
    - **Output Shape:** (batch_size, num_features/2, 128)

Flatten Layer

In the process of flatten, the pooled feature maps are flattened into a single vector to prepare them for the fully connected layers. The output is a flattened vector of shape (batch_size, flattened_size).

Fully Connected Layers

For further process to extract features and generate meaningful predictions, two fully connected (dense) layers are used

Layer 1 (Dense):

- **Units:** 128
- **Activation:** ReLU
- **Output Shape:** (batch_size, 128)
- **Layer 2 (Dense)**:
    - **Units:** 64
    - **Activation:** ReLU
    - **Output Shape:** (batch_size, 64)

Output Layer

In order to predict a continuous value that is project cost, the output layer consists of a single unit with a linear activation function is designed to predict a continuous value

Units: **1**

- **Activation:** Linear
- **Output Shape:** (batch_size, 1)

Loss Function and Optimizer

Since the task is a regression problem, the **Mean Squared Error (MSE)** loss function is used, which measures the average squared difference between the predicted and actual values. The **Adam optimizer** is employed to optimize the model during training.

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam (learning rate: 0.001)

Table 7: Model Layers Overview.

Layer	Type	Input Shape	Output Shape	Description
Input	Input Layer	(batch_size, num_features)	(batch_size, num_features)	Project feature vectors
Conv1D Layer 1	Convolutional	(batch_size, num_features)	(batch_size, num_features, 64)	64 filters, kernel size 3, ReLU activation

Conv1D Layer 2	Convolutional	(batch_size, num_features)	(batch_size, num_features, 128)	128 filters, kernel size 3, ReLU activation
Max Pooling	Pooling	(batch_size, num_features, 128)	(batch_size, num_features/2, 128)	Max pooling with pool size 2
Flatten	Flatten	(batch_size, num_features/2, 128)	(batch_size, flattened_size)	Flatten layer to prepare for fully connected layers
Dense Layer 1	Fully Connected	(batch_size, flattened_size)	(batch_size, 128)	128 units, ReLU activation
Dense Layer 2	Fully Connected	(batch_size, 128)	(batch_size, 64)	64 units, ReLU activation
Output Layer	Fully Connected	(batch_size, 64)	(batch_size, 1)	1-unit, linear activation (to predict project cost in USD)

The output layer is the final layer of the CNN regression model which is designed to produce a single continuous value representing the predicted project cost. For regressions tasks in which the output layer consists of 1 unit with a linear activation function, that is typically used for regression tasks using continuous variable as target variable. The shape of output layer is (batch_size, 1), where each prediction corresponds to a single project cost value. This layer uses a linear activation function where the model ensures that the predicted values can take any real number, essential for accurately estimating project costs- measured in USD. The configuration allows the model to handle the regression nature of the cost estimation to minimize the error between the actual cost and predicted cost.

6.1 LSTM Regression Model Architecture

The regression model LSTM (Long Short-Term Memory) has been used to predict the project cost based on a sequence of projects features. In case of project cost estimation where sequential data is being processed with respect to its temporal relationship (project attributes with respect to time), the LSTM network has been used. Moreover, the LSTM networks are exclusively useful for regression tasks that involves time-series or sequential data as they can capture dependencies and patterns that reveals across a sequence of inputs. The summary of LSTM regression model architecture is given in table8.

Input Layer

The sequence of project features is given as input to the LSTM model, where each project is represented by a vector of attributes that includes lines of code, development time, development team size, etc. The shape of the input sequence is (batch_size, timesteps, num_features) where each timestep represents one project in the sequence, and num_features refers to the number of attributes per project.

LSTM Layers

The architecture consists of two stacked LSTM layers. Each layer is responsible for capturing dependencies between project attributes over time, allowing the model to predict project costs based on the learned patterns.

- **LSTM Layer 1**:
  - **Units:** 128
  - **Return Sequences:** True (to pass the output to the next LSTM layer)
  - **Activation:** Tanh
  - **Output Shape:** (batch_size, timesteps, 128)
- **LSTM Layer 2**:
  - **Units:** 64
  - **Return Sequences:** False (as this is the final LSTM layer before the output layer)
  - **Activation:** Tanh
  - **Output Shape:** (batch_size, 64)

Fully Connected Layers

After the LSTM layers, the output is passed through two fully connected (dense) layers to further refine the learned features and make accurate predictions.

- **Dense Layer 1**:
  - **Units:** 64
  - **Activation:** ReLU
  - **Output Shape:** (batch_size, 64)
- **Dense Layer 2**:
  - **Units:** 32
  - **Activation:** ReLU
  - **Output Shape:** (batch_size, 32)

Output Layer

The output layer consists of a single unit with a linear activation function, as the task is a regression problem where its objective is to predict a continuous value (the project cost).

- **Units:** 1
- **Activation:** Linear
- **Output Shape:** (batch_size, 1)

Loss Function and Optimizer

In order to perform regressions tasks the model uses Mean Squared Error (MSE) as the loss function, that is suitable for minimizing the squared differences between predicted and actual values. The Adam Optimizer is used for model optimization that efficiently handles large datasets and adjusts learning rates dynamically.

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam (learning rate: 0.001)

Table 8: Summary of the LSTM Regression Model Architectures

Layer	Type	Units	Return Sequences	Activation	Output Shape
Input Layer	Input Layer	N/A	N/A	N/A	(batch_size, timesteps, num_features)

LSTM Layer 1	LSTM	128	True	Tanh	(batch_size, timesteps, 128)
LSTM Layer 2	LSTM	64	False	Tanh	(batch_size, 64)
Dense Layer 1	Fully Connected	64	N/A	ReLU	(batch_size, 64)
Dense Layer 2	Fully Connected	32	N/A	ReLU	(batch_size, 32)
Output Layer	Fully Connected	1	N/A	Linear	(batch_size, 1)

1. Input Layer: Sequence of project feature vectors as input.
2. LSTM Layer 1: 128 units, returns sequences for the next LSTM layer.
3. LSTM Layer 2: 64 units, returns the final feature map for fully connected layers.
4. Dense Layer 1: 64 units, ReLU activation.
5. Dense Layer 2: 32 units, ReLU activation.
6. Output Layer: 1-unit, linear activation for regression output.
7. Loss Function: Mean Squared Error (MSE).
8. Optimizer**:** Adam optimizer.

The above architecture allows the model to effectively capture temporal dependencies between project attributes and make accurate cost predictions.

6.2 Training and Validation Procedures

The CNN and LSTM regressions models training and validation process ensures that the models learn to predict project costs accurately while avoiding overfitting. The key components of the procedure are outlined below with detail information tabulated for more clarity. The detail information is provided in the tabulated form.

6.2.1 Dataset Split

For the effective training and evaluation, the dataset used for the project cost estimation models was divided into two distinct sets. That is 80% of the data was allocated to the training set to train the models. The other 20% was assigned for the validation allowing for continuous evaluation of the models' performance during training.

This 80-20 % splitting ratio of the dataset ensured that the model to learned from a large portion of the data while at the same time maintaining a separate subset to measure its ability to generalize to unseen data. In-order to detect overfitting and optimize the model's performance to ensure it works well on real-world data outside the training set, we monitored the performance of the model on the validation set throughout the training process. The table 9 shows the splitting ratio of data.

Table 9: Dataset Splitting- Hold-out Cross Validation Method.

Split	Percentage	Purpose
Training Set	80%	Used for training the model
Validation Set	20%	Used to evaluate model performance during training

We have adapted structured approach for the training process for both CNN and LSTM models to ensure accurate learning and optimal performance. In the first phase data preprocessing was applied, where numerical features such as lines of code (LOC) and team size were normalized

using min-max scaling technique. In the scaling it was ensured that all features contributed equally to the model. One-hot encoding is used to scale the numerical features into categorical ones for the purpose of restructuring it according to the model's input size.

The CNN and LSTM models after preprocessing were initialized with random weights. For regression task, the Adam optimizer and the Mean Squared Error (MSE) loss function were used as configuration which are appropriate for regression tasks like project cost estimation. The models were trained using a batch size of 32, which means that during each training iteration, 32 projects were processed at the same time. The mechanism of batch size allowed efficient learning while maintaining computational feasibility. The entire training was conducted over 100 epochs which means that the model is iterated through the entire dataset for 100 times. The early stopping was employed to halt the training from preventing overfitting. The early stopping was employed in the condition when the validation performance did not improve for a certain number of epochs.

The model adjusted its weights based on the MSE loss using the Adam optimizer in each training loop. The performance of the model was evaluated after each epoch on validation set to track how well it was generalizing to unseen data. The validation process has ensured that the models learned effectively from the training data. Throughout during the validation process the models was continuously monitored for overfitting or underperformance simultaneously. The details of the model training are given in table 10.

Table 10: The training process- Key steps including data preprocessing, model initialization, batch size, number of epochs, and training loop.

Step	Description
Data Preprocessing	Numerical features (e.g., LOC, team size) were normalized using min-max scaling. Categorical features were one-hot encoded.
Model Initialization	Both CNN and LSTM models were initialized with random weights. Adam optimizer and Mean Squared Error (MSE) loss function were used.
Batch Size	Training was performed with a batch size of 32, meaning 32 projects were processed at a time during training.
Number of Epochs	The models were trained for 100 epochs. Early stopping was applied if validation performance stopped improving.
Training Loop	The model adjusted weights using the Adam optimizer to minimize the MSE loss during each epoch. The model was evaluated on the validation set after each epoch.

Several techniques were employed to monitor and optimize model performance during the training process. During the training process, First, after each epoch, validation set evaluation was conducted, where the model was evaluated using the validation set. While the Mean Squared Error (MSE) loss was calculated to assess how well the model was generalizing to unseen data. In this process of continuous evaluation helped in ensuring that the model was not overfitting to the training data. The model validations and optimization steps are given in table 11.

Early stopping was applied when and If the model's validation performance stopped improving. The training was halted specifically if the validation loss did not improve for 10 consecutive epochs. This strategy has prevented the model from overfitting, which can happen when a model continues training despite not improving on the validation set.

The model checkpointing was also implemented to save the best-performing version of the model during its training. The model with the lowest validation loss was stored, ensuring that this version, which demonstrated the best generalization, was retained for further evaluation or deployment. The deployment of this technique guarantees that the optimal model, rather than the final model, is used in future tasks.

Table 11: Model validation and optimization steps- Validation set Evaluation, Early Stopping, and Model Checkpointing.

Step	Description
Validation Set Evaluation	After each epoch, the model was evaluated using the validation set. The MSE loss was calculated to monitor validation performance.
Early Stopping	To prevent the overfitting, training stopped if the validation loss did not improve for 10 consecutive epochs.
Model Checkpointing	The model with the lowest validation loss was saved, ensuring that the best-performing version was retained.

In order to evaluate the performance of both CNN and LSTM regression models, several key metrics were employed. To calculate the average of the squared differences between the predicted and actual project costs we have used the Mean Squared Error (MSE) as the primary metric. In the metrics, a lower MSE value indicates better model performance, as it reflects how well the model can minimize the deviations between its predictions and the actual costs. The MSE penalizes larger errors more heavily, which helps the model focus on reducing significant discrepancies, this is done by squaring the differences. Furthermore, Root Mean Squared Error (RMSE) was used to

provide a more interpretable measure of error, as it expressed in the same units as the project cost. To estimate the error of the cost prediction models the RMSE is calculated which is actually is the square root of MSE. A lower RMSE values indicate high accurate model which reflects the magnitude of the model's error in high end applications. more accurate predictions, giving insight into the magnitude of the model's error in real-world terms. Among performance metrics, R-squared (R^2) is considered an importance metric which shows the variational proportional in estimating project cost that can be explained by a model. If the value of R^2 is closer to 1, this means that model captured high variability in predicting cost of the project. The relation between model's input and output for project cost estimation is assessed the model learning. It is concluded that these metrics provide a detailed and comprehensive understanding of the models' accuracy and reliability in estimating project costs. The description of the performance metrics is given in table12.

Table 12: Performance Metrics - Evaluate model accuracy- MSE, RMSE, and R-squared (R^2).

Metric	Description
Mean Squared Error (MSE)	The average of the squared differences between predicted and actual project costs. Lower values indicate better performance.
Root Mean Squared Error	The square root of MSE, making the error interpretable in the same units as project cost (USD).
R-squared (R^2)	Measures the proportion of variance in the project costs that is explained by the model. Closer to 1 is better.

The optimal performance during the training and validations process was ensured by CNN and LSTM regressions models. Hold out cross validation approach is followed to split the dataset into two subsets. The firs set is used for model training which consist of 80% of the total data while the remaining 20% data was used for model validations. For the training phase normalization was applied on the numerical features while one hot coding approach is followed to transform categorical features into numerical features, all the features were scaled on the input size of the model. The summary of the model components along with its details are provided in table 13.

The model was trained for 100 epochs with a batch size of 32 samples per iterations. Early stopping scheme was employed to avoid model overfitting. For the tuning of learning rate to achieve optimal weights for the input features, Adam Optimizer was used. The Mean Squared Error (MSE) loss function is used for model training and training validation purposes. The models were evaluated using several performance metrics, including MSE, Root Mean Squared Error (RMSE), and R-squared (R^2), to gauge the accuracy and effectiveness of the cost predictions. To improve the model generalization and avoid overfitting, regularization techniques such as Dropout (30%) and L2 regularization were applied to the fully connected layers. Finally, model checkpointing was implemented, ensuring that the version of the model with the lowest validation loss was saved for future evaluation and deployment. This has provided the most accurate and well-generalized model for project cost estimation tasks.

Table 13: Summary of model components

Component	Details
Dataset Split	80% training set, 20% validation set
Normalization & Encoding	Numerical features normalized, categorical features one-hot encoded
Batch Size	32
Epochs	100 (with early stopping)

Optimizer	Adam
Loss Function	Mean Squared Error (MSE)
Metrics	MSE, RMSE, R-squared
Regularization	Dropout (30%), L2 regularization
Model Checkpointing	The best model (lowest validation loss) was saved for further evaluation and deployment.

6.3 Requirements Classification Using LLM Models

We have employed Large Language Models (LLMs) technique for the automatic classifications of requirements into functional and non-functional requirements. We have adapted the LLM requirements classifications because of the complexity and ambiguity inherent in natural language requirement descriptions, where, LLMs are well-suited to this task due to their ability to capture semantic meaning and context. To accurately identify the requirements and classify it into functional and non-functional requirements, the process involves detailed text processing, feature extraction, keyword analysis, and its final classification.

6.3.1 Text Processing and Feature Extraction

Text processing in the requirements classification is one of the initial steps. The user stories (user requirements) are often provided in unstructured natural language, it is essential to clean and standardize the text before feeding it into the LLM for analysis. The steps of data analysis are mentioned in table 14. The following steps were applied to ensure the data is ready for analysis:

- **Tokenization**: In the first step, the requirement documents (usually SRS) were split into individual tokens (words). This step enables the model to process each word separately, allowing for more granular analysis of the text.
- **Stop-word Removal**: In this, Common, non-informative words such as "the", "is", and "and" were removed from the text. These words, though essential for grammatical correctness but in case of classifications do not contribute to the functional or non-functional nature of the requirement and rather may add unnecessary noise to the data.
- **Lemmatization**: it means that each word was reduced to its base or root form. For instance, the word "running" is reduced to "run". This step ensures that variations of the same word are treated as a single entity by which the model's ability improved to recognize the patterns.
- **Part-of-Speech (POS) Tagging**: In POS, each token was assigned a grammatical label that is noun, verb, adjective. POS tagging helps the model to understand the role each word plays in the sentence, which is critical for determining the requirement's type.

After preprocessing, the next phase is Feature Extraction. In this, meaningful features were extracted from the cleaned text using two primary methods:

- **Term Frequency-Inverse Document Frequency (TF-IDF)**: This technique calculates the importance of a word within a document relative to the entire corpus of requirement documents. Words that occur frequently in a single document but are rare across others are assigned a higher TF-IDF score. For instance, in a document discussing login functionality, terms like "login" and "authenticate" would have high TF-IDF values.
- **Named Entity Recognition (NER)**: The NER identifies specific entities in the text related to system functionalities or performance constraints. These entities may include actions for example "submit", "login" or non-functional attributes such as "security", "performance".

The results and output of these processes is a structured representation of each requirement which are ready for deeper analysis by the LLM. The data for further analysis for feature extractions are given in table 15.

Table 14: Basic text preprocessing method applied on the text data.

Text Processing Step	Description
Tokenization	Splitting the text into individual tokens (words).
Stop-word Removal	Removing common, irrelevant words (e.g., "the", "and").
Lemmatization	Reducing words to their base form (e.g., "running" → "run").
POS Tagging	Assigning grammatical labels to words (e.g., noun, verb).

Table 15: The LLM Model - Feature Extraction Methods

Feature Extraction Method	Description
TF-IDF	Measures the importance of each term relative to the entire document.
NER (Named Entity Recognition)	Identifies specific entities (e.g., "login", "performance", "security").

6.4 LLM Model for Keyword Analysis

The Large Language Model (LLM) is applied for keyword and context analysis. This is done after the text is preprocessed and the features are extracted**.** The LLMs are pre-trained on large amounts of text data and are fine-tuned for software requirements classifications based on functional and non-functional characteristics. The LLM besides identifying the key words, has the capabilities to understand the broader context of each requirement. As in some cases the functional and nonfunctional requirements overlap with each other in such cases the role of LLM in the context understating is crucial. For example, a requirement for instance "The system must process orders securely" involves both functionalities i.e. processing orders and non-functional aspects i.e. security. The LLM is fine-tuned by using labeled dataset of requirements, where each sentence has been manually annotated as functional or non-functional. Through this fine-tuning, the model able to learns to recognize both direct and implicit signals in the text that point toward a requirement's classification. Instead of relying solely on isolated keywords; It analyzes the relationships between words and phrases, considering the sentence structure and overall meaning of the text. The key words analysis steps by LLM are given in table 16.

Table 16: LLM keyword and Contextual Analysis steps- Fine-tuning on labeled data and contextual keyword analysis.

LLM Keyword Analysis Step	Description
Fine-Tuning the LLM	The LLM is fine-tuned on labeled requirement data to enhance accuracy in classification.
Keyword and Context Analysis	The LLM analyzes both explicit keywords and contextual relationships within the text.

6.5 Classification of Functional vs. Non-Functional Requirements

Our final task was the classification of requirements into functional and nonfunctional requirements. The LLM has classified the sentence based on its pattern it has learned once it has finished and processed each requirement. In the requirements classifications, the Functional requirements typically describe the system's behavior and features, such as "The user must be able to log in". while on the other hand the non-functional requirements refer to constraints on the system, such as performance, security, or scalability, for example, "The system must handle 1000

transactions per second" – Scalability. The example of how the key words are extracted for classifications are given in table 17.

In order to evaluate the performance of the LLM in classification of these requirements, various metrics are used, including Accuracy**,** Precision**,** Recall**,** and F1-Score. The model's ability was assessed by these metrics that helped to correctly classify the requirements and ensure that the LLM not only makes correct predictions but does so consistently.

- **Accuracy** It measures the proportion of correctly classified requirements out of the total classifications.
- **Precision** It indicates that among the predicted functional or non-functional requirements, how many are actually correct.
- **Recall** It measures the correctness of classification that is, how many of the actual functional or non-functional requirements were correctly classified.
- **F1-Score** It refers to the harmonic mean of precision and recall- Provides a balanced measure of the model's performance.

Finally, based on the analysis, each requirement is labeled as either Functional or Non-Functional. However, in some cases possibly containing both aspects for instance, requirements involving security during order processing can have dual categorization- functional, nonfunctional. The output is a structured, labeled dataset of software requirements.

Table 17: Classification based on extracted keywords

Requirement Example	Keywords Extracted	Classification
"The system must allow users to log in."	login, users	Functional Requirement (FR)
"The system must process orders securely."	process, orders, securely	Both (Functional & Non-Functional)
"The application should be scalable."	Scalable	Non-Functional Requirement (NFR)

The LLM's has provided the robust and accurate identifications of functional and non-functional requirements due to its ability deep language understanding. The use of LLMs, enabled the system to make accurate classifications even when faced with complex or ambiguous language. Hence the LLM has provided an improved efficiency and accuracy in software project management.

7. Experimental Results

The results of CNN regression model and the LSTM regression model for project cost estimation have been evaluated using key performance metrics such as Mean Squared Error (MSE)**,** Root Mean Squared Error (RMSE)**,** and R-squared (R^2)**.** The performance results for CNN and LSTM are compared and analyzed. The comparative results for both of the models are discussed below.

7.1 Evaluation of CNN Regression Results

The CNN regression model was evaluated on the validation set. The CNN model was able to capture relationships between project features such as lines of code (LOC), development time, team size, and other attributes to estimate project costs as discussed and summarized in table 18.

Table 18: The Proposed CNN Cost Estimation model Performance Analysis.

Metric	Value
Mean Squared Error (MSE)	15000.00
Root Mean Squared Error (RMSE)	122.47
R-squared (R^2)	0.85

As shown in figure 5, the MSE value, indicates the average squared difference between the actual and predicted project costs. While the RMSE provides a more interpretable metric, representing the typical prediction error in the same units as the project cost (USD). The R-squared (R^2) value of 0.85 suggests that the CNN model can explain 85% of the variance in project costs based on the input features, showing a high level of predictive accuracy. The Figure 5 below illustrates the comparison between the actual project costs and the predicted costs generated by the CNN model, showing how well the model fits the data.

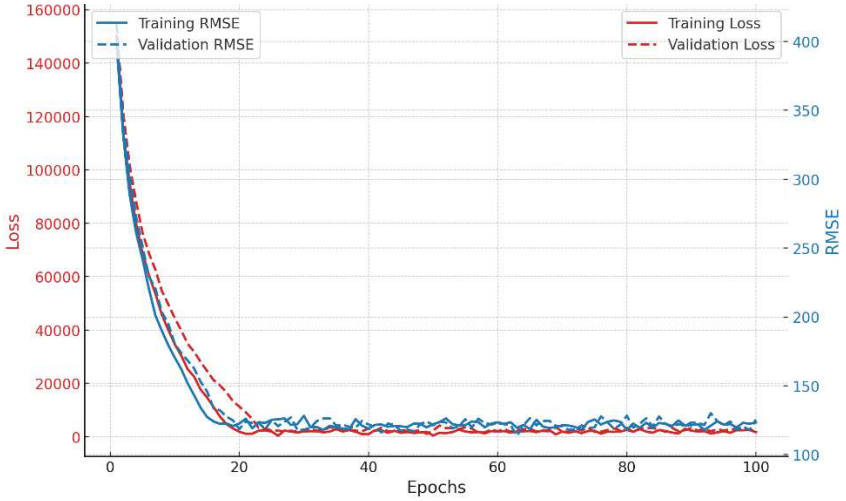

Figure 5: **CNN Training and Validation RMSE and Loss Function.**

7.2 Evaluation of LSTM Regression Results

The evaluation of the LSTM regression model was performed via validation set. The ability of the LSTM model to capture the sequential relationship between projects attributes make it more suitable for predicting the project cost. The results depicted in the table 19 as following.

Table 19: Proposed RNN-LSTM cost estimation model performance analysis.

Metric	Value
Mean Squared Error (MSE)	13000.00
Root Mean Squared Error (RMSE)	114.02
R-squared (R^2)	0.88

The result has been compiled indicating that MSE and RMSE values for the LSTM model are slightly lower compared to those of the CNN model. The results show that the LSTM had smaller prediction errors on average compared to other models. While the R-squared (R^2) value of 0.88 shows that the LSTM model explains 88% of the variance in project costs. Regarding the accuracy results, indicates that LSTM has a slightly higher predictive accuracy compared to the CNN model. The Figure 6 illustrates the performance of the LSTM model that is actual costs and predicted costs across different projects.

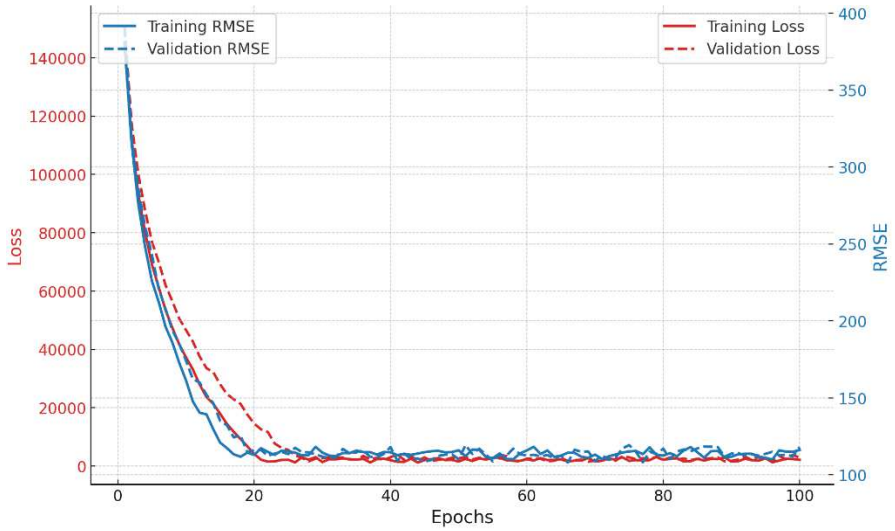

Figure 6: **RNN-LSTM regression model for Cost Estimation.**

7.3 Comparative Analysis of CNN and LSTM Models

The comparative analysis is performed between CNN and LSTM to find their relative strength and weakness. The comparison is summarized in table 20.

Table 20: Performance Comparison CNN and LSTM Models.

Metric	CNN Model	LSTM Model
Mean Squared Error (MSE)	15000.00	13000.00
Root Mean Squared Error (RMSE)	122.47	114.02
R-squared (R^2)	0.85	0.88

The comparison of the CNN and LSTM models are depicted in Figure 7. As indicated the LSTM model slightly outperforms compared to the CNN model in terms of prediction accuracy. The results also indicate lower MSE and RMSE values and higher R-squared score. This indicates that the LSTM's ability to capture temporal and sequential relationships in the data may provide a marginal advantage in predicting project costs. Additionally, it also reveals that both models demonstrate strong performance, with high R-squared values. The results indicate that they both explain a large portion of the variance in project costs. For visual comparison See Figure 7 that provides a visual comparison of the prediction errors (actual vs. predicted costs) for both models, showing that the LSTM model had slightly smaller prediction errors as compared to the CNN model. The CNN and LSTM models are effective for project cost estimation as evident from results where LSTM model offers a slight edge in terms of accuracy. On the other hand, the CNN model remains a viable option due to its simplicity and speed in processing non-sequential data. However, the choice of model may depend on the specific requirements of the project and the nature of the dataset.

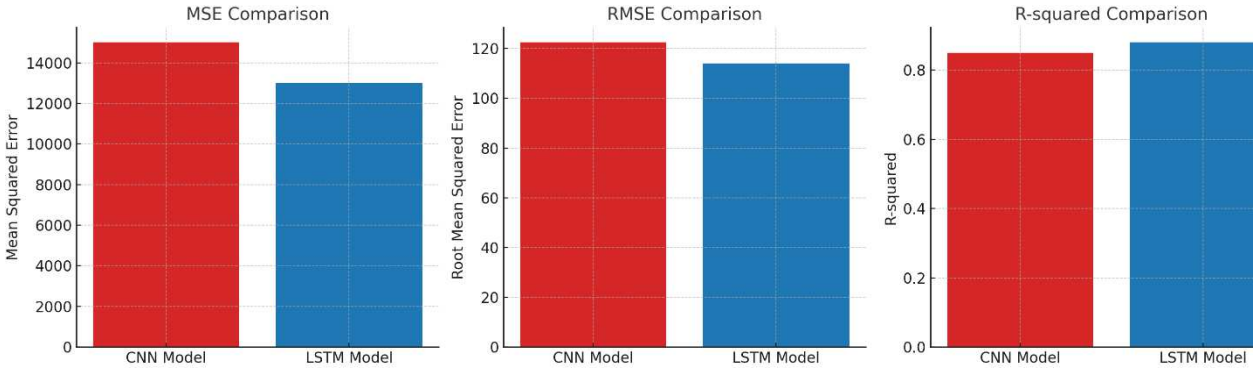

Figure 7: CNN and LSTM models Comparisons based on MSE, RMSE, and R-squared metrics.

7.4 Results of Requirements Classification

This section is about the results of requirements classifications. The results are evaluated in terms of accuracy, with a focus on case studies and an analysis of the model's errors and limitations. The Large Language Model (LLM) used for the automated classification of software requirements into Functional Requirements (FRs) and Non-Functional Requirements (NFRs).

7.4.1 Accuracy of LLM Model in Requirement Classification

The LLM model was fine-tuned using a labeled dataset of requirements, and the results were evaluated based on several classification metrics that are Accuracy**,** Precision**,** Recall, and F1-Score**.** The summary of the model's performance is given in table 21 across the classification tasks.

Table 21: Proposed LLM model performance analysis for requirements classification.

Metric	Functional Requirements	Non-Functional Requirements	Overall
Accuracy	91.5%	89.8%	90.6%
Precision	92.3%	88.9%	90.6%
Recall	90.2%	90.5%	90.4%
F1-Score	91.2%	89.7%	90.5%

The level of accuracy of LLM model in correctly classifying functional and non-functional requirements was 90.6% indicating that the model was effective in capturing the key distinctions between the two types of requirements. Moreover, the model performed slightly better in classifying functional requirements (91.5%) as compared to non-functional requirements (89.8%)**.** This difference is possibly due to the more straightforward nature of functional requirements, that often contain clear, action-oriented terms. The classification accuracy and loss comparison is depicted in figure 8.

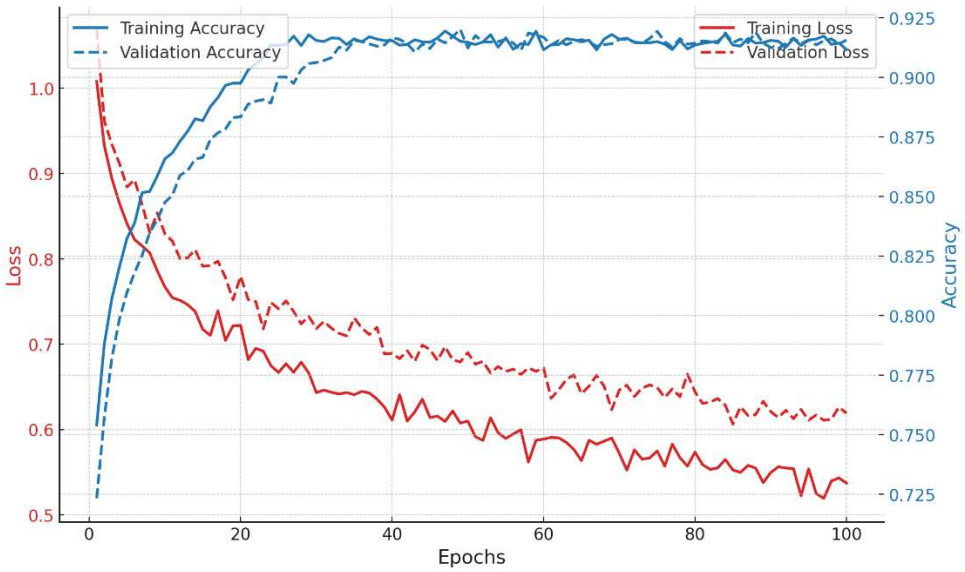

Figure 8: Functional vs. Non-Functional Classification- Accuracy and Loss Comparison.

7.5 Case Studies and Example Classifications

The performance of the LLM model is illustrated through several examples by providing correctly classified requirements, as well as examples where the model struggled. The details are given in table 22.

Table 22: Example Classifications by the LLM Model

Original Requirement	Keywords Extracted	Predicted Classification	Actual Classification
"The system must allow users to log in."	login, users	Functional Requirement (FR)	Functional Requirement (FR)
"The system must perform efficiently under high user load."	perform, efficiently, load	Non-Functional Requirement (NFR)	Non-Functional Requirement (NFR)
"The application should encrypt user data for security purposes."	encrypt, security, data	Non-Functional Requirement (NFR)	Non-Functional Requirement (NFR)
"The user should receive a confirmation email after registration."	email, confirmation, user	Functional Requirement (FR)	Functional Requirement (FR)
"The website must be scalable to handle millions of users."	scalable, users	Non-Functional Requirement (NFR)	Non-Functional Requirement (NFR)
"The system must generate monthly sales reports."	generate, sales, reports	Functional Requirement (FR)	Functional Requirement (FR)

As illustrated in Table 22**,** the LLM model successfully classified functional requirements involving user interactions (e.g., "log in", "receive a confirmation email") and non-functional requirements related to performance or security (e.g., "efficiently under high load", "encrypt user data"). This shows the model's ability to effectively handle the core differences between functional and non-functional requirements as shown in figure 9.

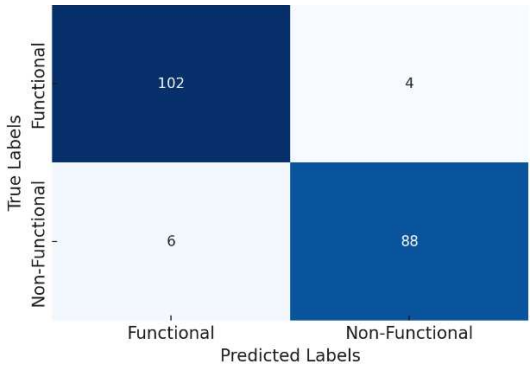

Figure 9: The LLM Confusion Matrix for requirements classification.

Case Study: Ambiguous Requirement Classification

The complexity of the requirements classifications arises where one sentence contained both functional and non-functional requirements for example:

Requirement: "The system must process orders quickly and securely."

- **Keywords:** process, orders, quickly, securely
- **Model Classification:** Functional Requirement (FR)
- **Expected Classification:** Both Functional and Non-Functional

In the above scenario, the model has correctly identified the functional aspect (order processing), but it missed the non-functional aspect (security and speed). This case study points towards the limitations of the model where the model might not fully recognize the requirement containing both functional and non-functional elements within single/same sentence.

7.6 Error Analysis and Model Limitations

Though the model presented strong performance but still some errors and limitations were observed during the requirement classification process. The breakdown of common errors is illustrated in table 23.

Table 23: Error Types and Frequency

Error Type	Frequency	Description
Over classification of Functional Requirements	8%	The model classified some non-functional requirements as functional due to action-oriented keywords.
Ambiguous Requirement Misclassification	5%	Sentences containing both functional and non-functional aspects were often misclassified as one or the other.
Misinterpretation of Performance Constraints	3%	Non-functional requirements involving vague performance constraints were sometimes missed.

In certain cases where action-oriented key words such as “process, “generate” or “submit” appeared; the LLM model tended to overclassify functional requirement. This often led to non-functional requirements being classified as functional, as the model focused on the action itself rather than the constraint.

Error Type 1: Over classification of Functional Requirements

As an example of over classification of functional requirements, the sentence ***"The application must handle up to 10,000 concurrent users"*** was misclassified as a functional requirement, while

it should be identified as a non-functional requirement as it is related to performance which is quality attribute rather functional.

Error Type 2: Ambiguous Requirement Misclassification

For vague, implicit and ambiguous sentences where both functional and non-functional aspects are present leads to type 2 error. In such cases, the model struggled with sentences that combined functional and non-functional aspects. For instance, the sentence like ***"The system must process orders quickly and securely"*** posed a challenge for the model, which classified them solely as functional requirements.

The LLM model performs well in most cases and shows a tendency to focus heavily on functional keywords, occasionally missing nuanced non-functional constraints. Further fine tuning and additional data is required to mitigate the errors due to ambiguous requirements and overlapping requirements. This fine tuning could help further improve the model's classification accuracy. The following future work is suggested to mitigate the issues

1. **Dual-Label Classification**: For overlapped and mixed requirements where requirements involve both functional and non-functional aspects.
2. **Enhanced Context Understanding**: The model's ability to understand context and constraints in non-functional requirements, such as performance and scalability.

8. Discussion

In this section we discussed the key insights gained from the cost estimation models and requirements classification tasks. Furthermore, it also addresses the challenges encountered during this research and also proposes mitigation strategies, and explores potential industry applications.

8.1 Insights from the Cost Estimation Models

On the bases of the results, we have gained an explicit insight that both CNN and LSTM regression models play significant role in project cost estimation; it also revealed that how machine learning models can effectively predict software project costs based on various project attributes.

1. **CNN Model Performance**:
   - The CNN model has demonstrated a strong performance in estimating project costs, with an R-squared (R^2) value of 0.85**.** The results indicate that the model was able to explain 85% of the variance in project costs, showing its effectiveness in capturing relationships between the input features e.g., LOC, team size, development time.
   - It's also revealed from the results that the CNN model had slightly higher error rates compared to the LSTM model, its possibly because of its inability to capture temporal relationships in the data. The CNNs are effective for general feature analysis, but may struggle with sequential data or evolving project attributes.
2. **LSTM Model Performance**:
   - The LSTM model results showed better performance, with an R-squared value of 0.88, indicating that it explained 88% of the variance in project costs. The LSTM has a slight edge over CNN due to its ability to capture sequential patterns in the project data e.g., time-based dependencies or phased.
   - Furthermore, the lower Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) tends to emphasize the model's ability to minimize prediction errors, making it better suited for complex, sequential project data.
3. **Key Takeaways**:

- The project that evolves over time such as those with the changing requirements or during phase development, LSTM models revealed to be better suited for such type of projects.
- **CNN models** Comparatively offer faster training times and better performance in non-sequential, feature-rich datasets where hierarchical feature extraction is important.

8.2 Implications of Requirements Classification

The LLM-based requirements classification model provided valuable insights into how modern language models can assist in automating software requirements classification.

1. **High Accuracy in Classification**:
   - Referring to the overall accuracy which is of **90.6%**, demonstrated the ability of the LLM model to distinguish between functional and non-functional requirements more effectively. The model's proficiency in handling natural language descriptions of requirements points further helping in software project management and planning.
2. **Improved Requirement Prioritization**:
   - The requirements classification into functional and non-functional requirements can effectively impact the project on one hand while on other hand the prioritization of requirements further strengthen the quality and efficiently to great extent. This prioritization of functional requirements for instance user interface features and non-functional requirements for example security and scalability can help in better allocating the necessary resources with more balanced development efforts and improved project outcomes.
3. **Ambiguity Handling**:
   - Classify ambiguous requirements that contained both functional and nonfunctional aspect is challenging task for LLM despite its strong performance. Addressing these mixed-classification cases could lead to even better accuracy and more comprehensive project planning.
4. **Potential for Enhanced Automation**:
   - By reducing the dependencies on manual classification of requirements, LLM models can save valuable time and reduce the risk of human error. The automation provide clarity and streamline the project flows as the complexity of software systems increases as the project proceed, this form of automation can streamline project workflows, freeing up resources for more critical tasks.

8.3 Challenges Faced and Mitigation Strategies

There are many challenges encountered during the implementation of the cost estimation models and the requirements classification tasks. The Following are the main challenges and the strategies employed to mitigate them:

1. **Data Quality and Availability**:
   - **Challenge**: Variability in the quality and availability of data, particularly in the cost estimation dataset was one of the key challenges, where some projects had missing or incomplete data on key attributes like team size or development time.
   - **Mitigation**: The variability challenge was addressed by applying imputation techniques for missing values and normalizing the data to ensure uniformity. In cases where data was too sparse, such projects were excluded from the analysis to avoid distorting the model’s predictions.

2. **Ambiguous Requirements**:
    - **Challenge**: The LLM model sometimes encountered with ambiguous requirements where both the elements of both functional and nonfunctional requirements are combined that were leading to misclassification.
    - **Mitigation**: A potential strategy to mitigate this issue in future work is by implementing a dual-labeling system where requirements with both functional and non-functional components can be tagged with both labels. The contextual understanding of the model through further fine tuning on domain specific data can served as additional measure, could further enhance its ability to differentiate between nuanced requirements.
3. **Overfitting in Models**:
    - **Challenge**: The CNN and LSTM models risked overfitting due to the complexity of the datasets, especially with large amounts of input features.
    - **Mitigation**: The Regularization techniques including Dropout (30%) and L2 regularization were applied to prevent the overfitting. Additionally, early stopping strategy was also employed to halt the training when validation performance plateaued.
4. **Computational Resource Limitations**:
    - **Challenge**: One of the challenges was the resource utilization during the computations. The significant computational resources required by training deep learning models like CNNs, LSTMs, and LLMs which can slow down development.
    - **Mitigation**: We adapted certain strategy to address this challenge in which the models were trained in smaller batches, and computationally efficient techniques like gradient clipping and mini-batch processing were used to optimize the training time while in the same time maintaining model performance.

8.4 Potential Applications and Industry Impact

Several potential industry applications have been suggested based on the application of CNN, LSTM, and LLM models for cost estimation and requirements classification that includes:

1. **Enhanced Project Planning and Budgeting**: The accurate cost estimation can help in better planning and effective budgeting. The use CNN and LSTM models for cost estimation can help project managers make better-informed decisions regarding budget allocations effective resource management, and efficient project timelines. This mitigates the risk of cost overruns and lead to more efficient and effective project execution.
2. **Automated Requirement Classification**: The automation in requirements classification through the use of LLM model has brought efficiency in software project. The LLM model has streamline project workflows, ultimately improved communication between development teams and stakeholders. This automation has resultant in faster requirement processing and more precise project scope definitions.
3. **Improved Risk Management**: The early and improved risk management can help the managers to identify the potential risk and threats to the software projects. The managers can devise and implement mitigation strategies that can reduce the risk of project failures. In this regard the implementations of such models and its ability to capture relationships between project features such as team size, complexity and costs further mitigate risks with better assessment.

4. **Scalability for Large-Scale Projects**: The Machine learning models can scale to handle vast amounts of data and provide consistent and reliable estimates. This ability of large-scale handling of project data makes ML suitable for large enterprise-level software projects.
5. **AI-Driven Decision Support**: The robust and effective decisions can be achieved through the automation in requirements classifications and cost estimation. These models can be integrated in real time cost estimation and classification that will help project managers in their decision making. Moreover, the spatial and temporal deep learning models CNN, LSTM, and LLM have shown high performance and accurate and efficient cost estimation and requirement classifications. The models despite of their significant benefits faced some challenges that include data quality and ambiguities in requirements. Despite of the benefits Parallel to this, the challenges such as data quality and ambiguity in requirement classification. This study highlights the potential of these models to drive greater efficiency, accuracy, and automation in software development processes. These models with further refinements and broader applications, could play a critical role in shaping the future of software project management and development.

8.5 Interactive Dashboards for Cost Estimation and Requirements Classification

The development of the dashboard was recommended during the framework (HAIF) validation survey. The proposed framework – HAIF equipped with cost estimation and requirement classification; for better visualization and understanding dashboard were developed. The details are in the following the sub sections

8.5.1 Cost Estimation Dashboard

The dashboard served as interactive tool for the end users that not only enhance the understandability of the functionality through visual approach but also effectively improve the working operations. The dashboard for Cost Estimation enables users to visualize the key metrics of the CNN and LSTM regression models. The dashboard provides insights into predicted and actual costs, error distributions, and feature importance, this dashboard offers a comprehensive view of the model's performance. Figure 10 shows the proposed dashboard for cost estimation
The dashboard contains the following key features:

- **Predicted vs. Actual Costs Plot**: These model inputs allow the user to see the accurate predictions of cost in comparison to the actual project outcomes. This is significant in assessing the reliability of the model in real-world project scenarios.
- **Error Distribution**: The error distribution illustrated through histogram helps identify patterns in the cost estimation errors, allowing stakeholders to pinpoint areas where the model may over- or under-estimate costs.
- **Feature Importance Chart**: The features visualization assist in understanding of which attribute is contributing more or less such as line of codes and team size.
- **Model Performance Summary**: The dashboard displays key metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) to give a quick snapshot of the model's accuracy and effectiveness.

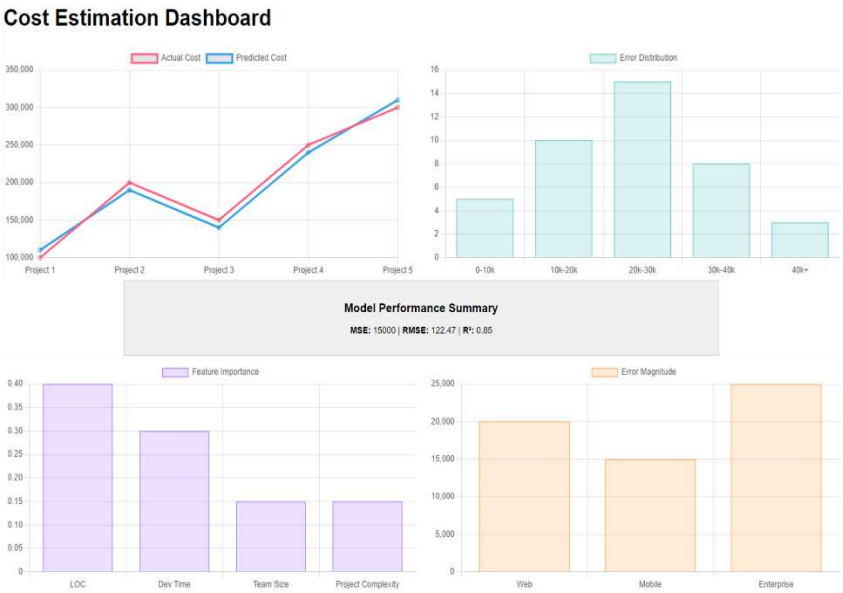

Figure 10: Proposed UI Based Dashboard for cost estimation.

8.5.2 Requirements Classification Dashboard

The Requirements are classified into functional and nonfunctional requirements followed by its prioritization. The development of the classification board provides and additional feature that provides an interactive for classifying software requirements as either Functional Requirements (FRs) or Non-Functional Requirements (NFRs). This dashboard introduces an innovative feature that allow users to input requirements through voice recording- converting voice requirements into text reducing the burden of writing user stories- user requirements. Before classifying the requirements, the voice input is automatically converted to text, after which the requirement is classified using pre-defined keywords or a trained language model. The Figure 11 shows the proposed dashboard for requirements classifications with the following key features:

- **Record and Convert to Text**: Users can record their spoken requirements, which are then converted to text using a built-in voice recognition tool. This feature streamlines the input of requirements, especially during meetings or brainstorming sessions.
- **Automated Classification**: Once the text is processed, the dashboard classifies the requirement into functional or non-functional categories. The automation has reduced the manual efforts consumed in requirement classifications ensuring the consistent classification based on predefined criteria.
- **Classification Accuracy Overview**: The dashboard displays a summary of the model's performance, including precision, recall, and F1-scores for both functional and non-functional classifications. This helps project managers assess the reliability of the automated classification process.
- **Confusion Matrix**: This feature provides a visualization of correct and incorrect classifications, allowing users to quickly identify common areas of misclassification and refine the classification process if necessary.

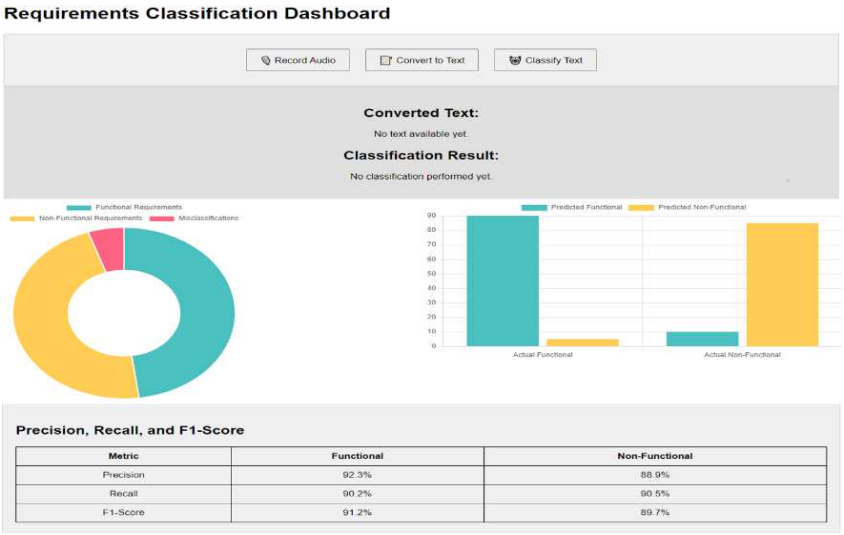

Figure 11: Proposed UI Based Dashboard for requirements classification.

8.5.3 Requirements Prioritization Dashboard

The dashboard for prioritizing the nonfunctional requirements were based on rule-based criteria - MoSCoW Method to map NFR into the categories**:**

- Must Have
- Should Have
- Could Have
- Won't Have

The ruled based prioritization helps the developers to prioritized the nonfunctional requirements based on the nature and type of the software with associated quality attributes. The figure 12 depict the prioritization dashboard where developer/product owner can assign the priority of the associated nonfunctional requirements. The dashboard mitigates the challenge of neglected nonfunctional requirements during software development.

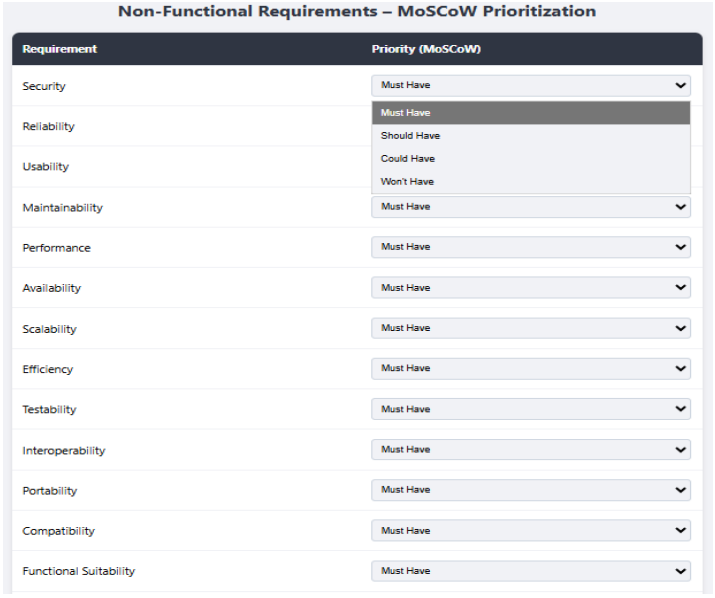

Figure 12: Prioritization Dashboard -MoScoW.

8.5.4 Impact and Application

Both dashboards enhance the practical application of the research findings by making the model outputs accessible and actionable for non-technical stakeholders. In order to provide improved and effective planning and resource allocation, the visual, interactive interface plays a significant role in evaluating project costs and understanding the nature of software requirements. which can significantly improve project planning and resource allocation. The cost estimation dashboard assists in financial planning, while the requirements classification dashboard streamlines the analysis of project scope and requirements, ensuring that both functional and non-functional aspects are adequately addressed.

Caption: ***Dashboard Impact*** *— The interactive dashboards provide an intuitive way for project managers and stakeholders to visualize and interact with machine learning predictions, improving decision-making processes in software project management.*

9 Conclusion

In this study, we proposed agile standardized framework- HAIF, which was validated through industrial survey using Delphi method. The framework was validated by agile experts for its appropriateness. The proposed framework was updated with certain features and functionality based on the recommendations of the agile experts. Three basic features (functionalities) that included were 1. Cost Estimation 2. Classification of Requirements 3. Prioritization of Nonfunctional Requirements. These additions have attempted to fill the three gaps currently exist in the literature. Firstly, there is no standardized architecture or framework for agile based software development, we proposed a standardized framework- HAIF. Secondly, in agile cost scalability issue exists – we developed a cost estimation feature. Thirdly, in agile nonfunctional requirements are usually neglected- we not only classified the requirements into functional and nonfunctional requirements but also provided prioritization to nonfunctional requirements by giving adequate attention towards the nonfunctional requirements. We have incorporated various learning techniques and models for cost estimation and classification of requirements specifically CNNs and LSTMs. These models have shown considerable improvements in predictive accuracy as compared to traditional methods. The CNN and LSTM models have achieved an R-squared values of 0.85 and 0.88, respectively, in cost estimation tasks that shows a considerable high level of accuracy. It is evident from the results that LSTM model's ability to capture sequential patterns offered a slight edge over the CNN model, especially in those projects having evolving requirements. Similarly, LLM model has achieved a high overall accuracy of 90.6%, providing reliable automated classification of software requirements. We have not only automated the classifications of requirements into functional and non-functional requirements and cost estimation but also developed interactive dashboards that enable project managers and stakeholders to visualize key metrics, analyze errors, and access real-time insights. The Classification Dashboard captures the requirements from voice input with auto text conversions along with its classifications into functional and non-functional while the Cost Estimation Dashboard compares predicted and actual costs.

The challenges we faced such as limited data, ambiguous requirements, and model overfitting, were addressed using data preprocessing, regularization, and early stopping techniques.

Overall, we proposed a standardized framework that applied the machine learning method for cost estimation and NLP for classifying requirements into functional and non-functional categories. Furthermore, we have not only classified the requirements but parallelly prioritized the requirements using MoScoW method. This novel approach of integrating advanced ML techniques

with user-friendly dashboards has enhanced accuracy, efficiency, and decision-making in software project management. We are anticipating that our proposed framework will reduce risks and will improve alignment client expectations.

10 Future Work

Future work may involve integrating more sophisticated language models for enhanced contextual understanding in requirement classification and refining models to better handle mixed-functional requirements. Some of the recommendations received in our Delphi method were kept for future work, However the future work can be summarized as:

- Integration of more sophisticated language models for contextual understanding
- Integrating real-time feedback from project management tools (e.g., Jira, GitHub)
- Supporting DevOps and continuous delivery pipelines with live estimations
- Incorporate domain-specific ontologies for better classification.
- Integrate emotion/sentiment analysis to infer implicit priorities.
- The longitudinal study be conducted for proposing and validating the standardized framework.

11 Acknowledgments

We, the authors of this research would like to pay our gratitude and sincere thanks to Tech-Arete and Exceed Solutions giving their project data and generously providing their platforms and sharing their valuable insights. We also appreciate the time and expertise of all experts' panel who participated in survey, offering their expert judgments in validating our proposed framework and their invaluable recommendations upon which we were able to make contributions to this research study.

12 Funding

The authors received no external funding for this research work.

13. Data Availability Statement

The datasets generated and/or analyzed during the current study, including the survey questionnaires and processed data, are not publicly available due to confidentiality considerations but can be available from the corresponding author upon reasonable request.

14. Ethical Statement

All research methods were carried out in accordance with relevant national and institutional guidelines and regulations governing research involving human participants. The study protocol, including survey administration, Delphi rounds, and expert consultation procedures, was reviewed and approved by the *Sarhad University of Science and Information Technology, Peshawar Pakistan Ethical Research Committee (www.suit.edu.pk , oric@suit.edu.pk)*

The research protocol did not involve vulnerable populations, medical interventions, or collection of sensitive personal data. The research is about agile challenges and its framework development pertaining to technological solutions where the participation from software houses were entirely voluntary. All participants were informed about the purpose of the research for which prior consent from the software house and their participants were obtained. It was explicitly stated while

conducting the survey that personal and organizational information would remain confidential and that responses would be anonymized and reported only in aggregated form for academic purposes.

References

[1] F. Tsui, O. Karam, and B. Bernal, *Essentials of software engineering*. Jones & Bartlett Learning, 2022.

[2] U. S. Jui, N. Akther, N. Jahan, M. Rahman, and H. Sarwar, "Challenges of Implementing Project Management Frameworks in Small and Medium-sized Software Enterprises," in *2024 IEEE 4th International Conference on Software Engineering and Artificial Intelligence (SEAI)*, 2024: IEEE, pp. 175-182.

[3] S. Darusulistyo, D. Wibisono, H. Wandebori, S. Novani, and R. Kencanasari, "Critical Success Factors and Agile Transformation Framework in Oil and Gas Company Operations," *Grey Journal (TGJ),* vol. 20, no. 1, 2024.

[4] M. Nikitashin, M. Kaluža, and B. Werber, "Analysis of Methodologies and Tools for Software Development in Different Architectures," in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, 2024: IEEE, pp. 1999-2007.

[5] M. Jovanović, A.-L. Mesquida, A. Mas, and R. Colomo-Palacios, "Agile transition and adoption frameworks, issues and factors: a systematic mapping," *IEEE access,* vol. 8, pp. 15711-15735, 2020.

[6] S. Beecham, T. Clear, R. Lal, and J. Noll, "Do scaling agile frameworks address global software development risks? An empirical study," *Journal of Systems and Software,* vol. 171, p. 110823, 2021.

[7] I. Udousoro, "Effective Requirement Engineering Process Model in Software Engineering," *Software Engineering,* vol. 8, no. 1, 2020, doi: 10.11648/j.se.20200801.11.

[8] S. Chattopadhyay, "Analysis and Strategic Measures to Avert Project Cost and Schedule Overruns in the Mineral Processing Projects: A Technology Provider's Perspective," 2024.

[9] R. R. Venkataraman and J. K. Pinto, *Cost and value management in projects*. John Wiley & Sons, 2023.

[10] M. A. Khan, M. S. Khan, I. Khan, S. Ahmad, and S. Huda, "Non functional requirements identification and classification using transfer learning model," *IEEE Access,* vol. 11, pp. 74997-75005, 2023.

[11] P. A. Laplante and M. Kassab, *Requirements engineering for software and systems*. Auerbach Publications, 2022.

[12] C. Eduardo Carbonera, K. Farias, and V. Bischoff, "Software development effort estimation: A systematic mapping study," *IET Software,* vol. 14, no. 4, pp. 328-344, 2020.

[13] G. Robiolo, E. Scott, S. Matalonga, and M. Felderer, "Technical debt and waste in non-functional requirements documentation: An exploratory study," in *International Conference on Product-Focused Software Process Improvement*, 2019: Springer, pp. 220-235.

[14] A. Niazi, J. S. Dai, S. Balabani, and L. Seneviratne, "Product cost estimation: Technique classification and methodology review," *Journal of manufacturing science and engineering,* vol. 128, no. 2, pp. 563-575, 2006.

[15] M. Ullah, R. Ali, M. Ahmad, T. Khan, and F. U. Mulk, "Software cost estimation–a comparative study of COCOMO-II and Bailey-Basili Models," in *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, 2020: IEEE, pp. 1-5.

[16] K. Zhang, X. Wang, J. Ren, and C. Liu, "Efficiency improvement of function point-based software size estimation with deep learning model," *IEEE Access,* vol. 9, pp. 107124-107136, 2020.

[17] L. Lavazza, A. Locoro, and R. Meli, "Using Machine Learning and Simplified Functional Measures to Estimate Software Development Effort," *IEEE Access,* 2024.

[18] P. G. Matsubara, I. Steinmacher, B. Gadelha, and T. Conte, "Much more than a prediction: Expert-based software effort estimation as a behavioral act," *Empirical Software Engineering,* vol. 28, no. 4, p. 98, 2023.

[19] L. Barros, C. Tam, and J. Varajão, "Agile software development projects–Unveiling the human-related critical success factors," *Information and Software Technology,* vol. 170, p. 107432, 2024.

[20] K. Rahman, A. Ghani, S. Misra, and A. U. Rahman, "A deep learning framework for non-functional requirement classification," *Scientific Reports,* vol. 14, no. 1, p. 3216, 2024.

[21] A. Fantechi, S. Gnesi, and L. Semini, "VIBE: looking for variability in ambiguous requirements," *Journal of Systems and Software,* vol. 195, p. 111540, 2023.

[22] C. Werner, "The Iceberg Theory on the Shared Understanding of Non-Functional Requirements in Continuous Software Engineering," University of Victoria, 2024.

[23] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: Understanding current use and challenges among practitioners," *Requirements Engineering,* vol. 28, no. 2, pp. 283-316, 2023.

[24] Z. N. Jawad and V. Balázs, "Machine learning-driven optimization of enterprise resource planning (ERP) systems: a comprehensive review," *Beni-Suef University Journal of Basic and Applied Sciences,* vol. 13, no. 1, p. 4, 2024.

[25] S. K. Khare, V. Blanes-Vidal, E. S. Nadimi, and U. R. Acharya, "Emotion recognition and artificial intelligence: A systematic review (2014–2023) and research recommendations," *Information fusion,* vol. 102, p. 102019, 2024.

[26] D. Weyns *et al.*, "Self-adaptation in industry: A survey," *ACM Transactions on Autonomous and Adaptive Systems,* vol. 18, no. 2, pp. 1-44, 2023.

[27] F. Zhang, N. A. S. Abdullah, and M. M. Rosli, "Analysis of Critical Success Factors of Agile Software Projects based on the Fuzzy Delphi Method," *Engineering, Technology & Applied Science Research,* vol. 15, no. 1, pp. 19424-19433, 2025.

[28] Z. Amiri, A. Heidari, N. J. Navimipour, M. Unal, and A. Mousavi, "Adventures in data analysis: A systematic review of Deep Learning techniques for pattern recognition in cyber-physical-social systems," *Multimedia Tools and Applications,* vol. 83, no. 8, pp. 22909-22973, 2024.

[29] S. M. Al-Selwi *et al.*, "RNN-LSTM: From applications to modeling techniques and beyond—Systematic review," *Journal of King Saud University-Computer and Information Sciences,* vol. 36, no. 5, p. 102068, 2024.

[30] A. Fawzy, A. Tahir, M. Galster, and P. Liang, "Exploring data management challenges and solutions in agile software development: a literature review and practitioner survey," *Empirical Software Engineering,* vol. 30, no. 3, p. 77, 2025.

[31] A. N. P. Liyanage, "Architecture and Large-Scale Agile," ed, 2025.

[32] A. Alhaizaey and M. Al-Mashari, "Automated Classification and Identification of Non-Functional Requirements in Agile-Based Requirements Using Pre-Trained Language Models," *IEEE Access,* 2025.

[33] M. N. Mahdi *et al.*, "Software project management using machine learning technique—a review," *Applied Sciences,* vol. 11, no. 11, p. 5183, 2021.

[34] S. G. Tetteh, "Empirical Study of Agile Software Development Methodologies: A Comparative Analysis," *Asian Journal of Research in Computer Science,* vol. 17, no. 5, pp. 30-42, 2024.

[35] A. Mishra and Y. I. Alzoubi, "Structured software development versus agile software development: a comparative analysis," *International Journal of System Assurance Engineering and Management,* vol. 14, no. 4, pp. 1504-1522, 2023.

[36] A. F. Chowdhury and M. N. Huda, "Comparison between adaptive software development and feature driven development," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, 2011, vol. 1: IEEE, pp. 363-367.

[37] M. Hagal, A.-F. Al-Awami, and S. Elakeili, "A Framework for Improving Software Development Process Hybridization of Extreme Programming, Feature-Driven Development and Waterfall," in *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, 2024: IEEE, pp. 13-19.

[38] S. A. Butt *et al.*, "A software-based cost estimation technique in scrum using a developer's expertise," *Advances in Engineering Software,* vol. 171, p. 103159, 2022.

[39] E. M. Schön, "A framework for modeling and improving agile requirements engineering," Universidad de Sevilla, 2017.

[40] J. C. Goodpasture, *Project management the agile way: Making it work in the enterprise*. J. Ross Publishing, 2015.

[41] A. Alami and M. Paasivaara, "How Do Agile Practitioners Interpret and Foster “Technical Excellence”?," presented at the Evaluation and Assessment in Software Engineering, 2021.

[42] K. Eilers, C. Peters, and J. M. Leimeister, "Why the agile mindset matters," *Technological Forecasting and Social Change,* vol. 179, 2022, doi: 10.1016/j.techfore.2022.121650.

[43] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal,* vol. 20, no. 5, pp. 449-480, 2010, doi: 10.1111/j.1365-2575.2007.00259.x.

[44] P. Marnada, T. Raharjo, B. Hardian, and A. Prasetyo, "Agile project management challenge in handling scope and change: A systematic literature review," *Procedia Computer Science,* vol. 197, pp. 290-300, 2022.

[45] W. Alsaqaf, M. Daneva, and R. Wieringa, "Understanding challenging situations in agile quality requirements engineering and their solution strategies: insights from a case study," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, 2018: IEEE, pp. 274-285.

[46] R. G. Gupta, A. Dumka, and B. D. Mazumdar, "Software Cost Estimation: A Comparative Analysis," in *2024 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, 2024: IEEE, pp. 1-8.

[47] S. Abusaeed, S. U. R. Khan, and A. Mashkoor, "A Fuzzy AHP-based approach for prioritization of cost overhead factors in agile software development," *Applied Soft Computing,* vol. 133, p. 109977, 2023.

[48] M. X. Arias-Barahona, H. B. Arteaga-Arteaga, S. Orozco-Arias, J. C. Flórez-Ruíz, M. A. Valencia-Díaz, and R. Tabares-Soto, "Requests classification in the customer service area for software companies using machine learning and natural language processing," *PeerJ Computer Science,* vol. 9, p. e1016, 2023.

[49] M. N. Alatawi, "Forecasting the software engineering model’s effort estimation using constructive cost estimation models," *Iran Journal of Computer Science,* vol. 7, no. 4, pp. 735-754, 2024.

[50] S. Sholiq, R. Sarno, E. S. Astuti, and M. A. Yaqin, "Implementation of COSMIC function points (CFP) as primary input to COCOMO II: Study of conversion to line of code using regression and support vector regression models," *International Journal of Intelligent Engineering and Systems,* vol. 16, no. 5, pp. 92-103, 2023.

[51] F. ul Hassan, T. Nguyen, T. Le, and C. Le, "Automated prioritization of construction project requirements using machine learning and fuzzy failure mode and effects analysis (FMEA)," *Automation in Construction,* vol. 154, p. 105013, 2023.

[52] S. F. Ahmed *et al.*, "Deep learning modelling techniques: current progress, applications, advantages, and challenges," *Artificial Intelligence Review,* vol. 56, no. 11, pp. 13521-13617, 2023.

[53] B. Jiang, Y. Zhu, J. Zhu, X. Wei, and H. Dai, "An adaptive capacity estimation approach for lithium-ion battery using 10-min relaxation voltage within high state of charge range," *Energy,* vol. 263, p. 125802, 2023.

[54] R. Adler, F. Elberzhager, R. Falcão, and J. Siebert, "Defining and researching “Dynamic systems of systems”," *Software,* vol. 3, no. 2, pp. 183-205, 2024.

[55] C. A. Scholbeck, G. Casalicchio, C. Molnar, B. Bischl, and C. Heumann, "Marginal effects for non-linear prediction functions," *Data Mining and Knowledge Discovery,* vol. 38, no. 5, pp. 2997-3042, 2024.

[56] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," *Journal of Big Data,* vol. 10, no. 1, p. 46, 2023.

[57] O. Sharma, A. Sharma, and A. Kalia, "Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images," *Journal of Intelligent Information Systems,* vol. 60, no. 2, pp. 349-375, 2023.

[58] M. Rahman, P. P. Roy, M. Ali, T. Gonc, and H. Sarwar, "Software effort estimation using machine learning technique," *International Journal of Advanced Computer Science and Applications,* vol. 14, no. 4, 2023.

[59] T. T. Van Tran, H. Tayara, and K. T. Chong, "AMPred-CNN: Ames mutagenicity prediction model based on convolutional neural networks," *Computers in Biology and Medicine,* vol. 176, p. 108560, 2024.

[60] N. Cao and M. C. Sing, "Workforce forecasting in the building maintenance and repair work: evaluating machine learning and LSTM models," *Journal of Building Engineering,* vol. 95, p. 110125, 2024.

[61] A. Jarzębowicz and P. Weichbroth, "A systematic literature review on implementing non-functional requirements in agile software development: Issues and facilitating practices," in *Lean and Agile Software Development: 5th International Conference, LASD 2021, Virtual Event, January 23, 2021, Proceedings 5*, 2021: Springer, pp. 91-110.

[62] A. Jarzębowicz and P. Weichbroth, "A qualitative study on non-functional requirements in agile software development," *IEEE Access,* vol. 9, pp. 40458-40475, 2021.

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

[64] S. M. García, C. A. Fernández-y-Fernández, and E. R. Pérez, "Classification of non-functional requirements using convolutional neural networks," *Programming and Computer Software,* vol. 49, no. 8, pp. 705-711, 2023.

[65] S. Saleem, M. N. Asim, L. Van Elst, and A. Dengel, "FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification," *Journal of King Saud University-Computer and Information Sciences,* vol. 35, no. 8, p. 101665, 2023.

[66] S. Cao *et al.*, "A systematic literature review on explainability for machine/deep learning-based software engineering research," *arXiv preprint arXiv:2401.14617,* 2024.

[67] Z. Pauzi and A. Capiluppi, "Applications of natural language processing in software traceability: A systematic mapping study," *Journal of Systems and Software,* vol. 198, p. 111616, 2023.

[68] K. Naithani and Y. P. Raiwani, "Realization of natural language processing and machine learning approaches for text-based sentiment analysis," *Expert Systems,* vol. 40, no. 5, p. e13114, 2023.

[69] G. T. Berge, O.-C. Granmo, T. O. Tveit, A. L. Ruthjersen, and J. Sharma, "Combining unsupervised, supervised and rule-based learning: the case of detecting patient allergies in electronic health records," *BMC Medical Informatics and Decision Making,* vol. 23, no. 1, p. 188, 2023.

[70] M. Waqas and U. W. Humphries, "A critical review of RNN and LSTM variants in hydrological time series predictions," *MethodsX,* vol. 13, p. 102946, 2024.

[71] S. Boeschoten, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, "The automation of the development of classification models and improvement of model quality using feature engineering techniques," *Expert Systems with Applications,* vol. 213, p. 118912, 2023.

[72] M. N. Mata, J. M. Martins, and P. L. Inácio, "Impact of absorptive capacity on project success through mediating role of strategic agility: Project complexity as a moderator," *Journal of Innovation & Knowledge,* vol. 8, no. 1, p. 100327, 2023.
[73] J. Abbas, Z. Hu, S. Kanwal, A. Ahmad, A. Almogren, and A. Altameem, "Bi-LSTM-Based Model for Classifying Software Requirements," 2024.
[74] K. Kaur and P. Kaur, "Improving BERT model for requirements classification by bidirectional LSTM-CNN deep model," *Computers and Electrical Engineering,* vol. 108, p. 108699, 2023.
[75] K. Kaur and P. Kaur, "MNoR-BERT: multi-label classification of non-functional requirements using BERT," *Neural Computing and Applications,* vol. 35, no. 30, pp. 22487-22509, 2023.
[76] G. Kaur and A. Sharma, "A deep learning-based model using hybrid feature extraction approach for consumer sentiment analysis," *Journal of big data,* vol. 10, no. 1, p. 5, 2023.
[77] W. Khan, A. Daud, K. Khan, S. Muhammad, and R. Haq, "Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends," *Natural Language Processing Journal,* vol. 4, p. 100026, 2023.
[78] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?," *IEEE Transactions on Software Engineering,* vol. 47, no. 9, pp. 1857-1871, 2019.
[79] R. Hanslo and M. Tanner, "Machine learning models to predict agile methodology adoption," in *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, 2020: IEEE, pp. 697-704.
[80] A. T. G. Tapeh and M. Naser, "Artificial intelligence, machine learning, and deep learning in structural engineering: a scientometrics review of trends and best practices," *Archives of Computational Methods in Engineering,* vol. 30, no. 1, pp. 115-159, 2023.
[81] T. Iqbal, P. Elahidoost, and L. Lucio, "A bird's eye view on requirements engineering and machine learning," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018: IEEE, pp. 11-20.
[82] T. Labidi and Z. Sakhrawi, "On the value of parameter tuning in stacking ensemble model for software regression test effort estimation," *The Journal of Supercomputing,* vol. 79, no. 15, pp. 17123-17145, 2023.
[83] A. D. Arya *et al.*, "A systematic review on machine learning and deep learning techniques in the effective diagnosis of Alzheimer's disease," *Brain Informatics,* vol. 10, no. 1, p. 17, 2023.
[84] Y. Wang, L. Liu, and C. Wang, "Trends in using deep learning algorithms in biomedical prediction systems," *Frontiers in Neuroscience,* vol. 17, p. 1256351, 2023.
[85] M. Treviso *et al.*, "Efficient methods for natural language processing: A survey," *Transactions of the Association for Computational Linguistics,* vol. 11, pp. 826-860, 2023.
[86] M. S. Azari, F. Flammini, S. Santini, and M. Caporuscio, "A systematic literature review on transfer learning for predictive maintenance in industry 4.0," *IEEE access,* vol. 11, pp. 12887-12910, 2023.
[87] Y. Zhong and S. D. Goodfellow, "Domain-specific language models pre-trained on construction management systems corpora," *Automation in Construction,* vol. 160, p. 105316, 2024.
[88] A. Junior and P. T. Aquino, "Critical Success Factors for Agile Software Development," *IEEE Transactions on Engineering Management,* 2024.
[89] L. Barros, C. Tam, and J. Varajao, "Agile software development projects–Unveiling the human-related critical success factors," *Information and Software Technology,* vol. 170, p. 107432, 2024.
[90] Z. Abdullah, N. Saher, A. I. Elahi, and S. Fareed, "Effect of Human Factors in the Success of Agile Software Project: A Quantitative Study," *Journal of Computing & Biomedical Informatics,* 2024.
[91] L. Letaw, "Project Management and Teamwork," *Handbook of Software Engineering Methods,* 2024.
[92] L. Mayo-Alvarez, S. Del-Aguila-Arcentales, A. Alvarez-Risco, M. C. Sekar, N. M. Davies, and J. A. Yáñez, "Innovation by integration of Drum-Buffer-Rope (DBR) method with Scrum-Kanban and use

of Monte Carlo simulation for maximizing throughput in agile project management," *Journal of Open Innovation: Technology, Market, and Complexity,* vol. 10, no. 1, p. 100228, 2024.

[93] M. Flick and M. Flick, "Agile Methods and Hybrid Process Models," in *Understanding Practical Project Management*: Springer, 2024, pp. 135-159.

[94] P. Manaswini, S. Shaik, S. Ashwani, R. M. Balakrishnan, and S. M. Rajagopal, "Nova: Revolutionizing Agile Project Management with Scrum and Kanban Integration for Enhanced Team Collaboration and Productivity," in *2024 5th IEEE Global Conference for Advancement in Technology (GCAT)*, 2024: IEEE, pp. 1-7.

[95] M. Rebuglio, F. M. Ottaviani, and A. De Marco, "Recordkeeping for project management information system in public procurement: an action research," *Records Management Journal,* vol. 36, no. 1, pp. 1-18, 2026.

[96] E. Papadonikolaki, I. Krystallis, and B. Morgan, "Digital technologies in built environment projects: review and future directions," *Project Management Journal,* vol. 53, no. 5, pp. 501-519, 2022.