Streamlining the E-bidding process through automated compliance check of procurement advertisements

FINAL YEAR PROJECT UG-2019

By

Muhammad Faseeh Naveed (G.L)

Mohammad Zaheer Ahmed

Mudassir Rasool

Muhammad Abdul Rehman

Project Advisor

Dr. Muhammad Umer Zubair

NUST Institute of Civil Engineering

School of Civil and Environmental Engineering

National University of Sciences and Technology

Islamabad, Pakistan

2023

Streamlining the E-bidding process through automated compliance check of procurement advertisements

FINAL YEAR PROJECT UG-2019

By

Muhammad Faseeh Naveed (G.L)	300732
Mohammad Zaheer Ahmed	286655
Mudassir Rasool	289047
Muhammad Abdul Rehman	282505

Project Advisor

Dr. Muhammad Umer Zubair

NUST Institute of Civil Engineering

School of Civil and Environmental Engineering

National University of Sciences and Technology, Islamabad, Pakistan

2023

This is to certify that the Final Year Project titled

Streamlining the E-bidding process through automated compliance check of procurement advertisements

Submitted by

Muhammad Faseeh Naveed (G.L)	300732
Mohammad Zaheer Ahmed	286655
Mudassir Rasool	289047
Muhammad Abdul Rehman	282505

Has been accepted towards the requirements

for the award of an undergraduate degree

Bachelor of Engineering in Civil Engineering

Dr. Muhammad Umer Zubair

Assistant Professor

NUST Institute of Civil Engineering (NICE)

School of Civil and Environmental Engineering (SCEE)

National University of Sciences and Technology, Islamabad, Pakistan

ACKNOWLEDGEMENT

We are grateful to Almighty Allah for giving us the opportunity to complete this research document by granting us strength, courage, and perseverance. Without the constant guidance and support from our Project Advisor, Dr. Muhammad Umer Zubair, and the respected faculty of the CEM department, this research would have been impossible to complete.

We would also like to thank Engr. Bakhtiar Hussain from Sukkhur IBA University for inspiring us to work on this subject as well as our industry correspondents who supported us throughout the course of the project and provided valuable feedback on improving our product to meet the industry needs.

ABSTRACT

The construction industry is an essential contributor to the economy in many countries. However, due to its unique nature, it has been perceived to lag behind in innovation. Bidding is a crucial component of construction projects that involves multiple stakeholders and generates significant paperwork. To address the challenges associated with paper-based bidding, electronic bidding (e-bidding) has been proposed as a solution. E-bidding is less costly and time-consuming, making it more efficient and effective. However, compliance checking of bid documents during the e-bidding process is currently performed manually, increasing the risk of errors and requiring significant manpower. This research proposes a framework for automated compliance verification of bid documents with predefined standards, specifically the regulations of the Public Procurement Regulatory Authority (PPRA) in Pakistan. By automating compliance verification, the e-bidding process can operate more efficiently, resulting in improved project outcomes and overall performance of the construction industry. The proposed framework also sheds light on the potential of Natural Language Processing (NLP) algorithms in automating compliance verification, which could contribute significantly to the advancement of the construction industry and the field of machine learning for compliance verification. In validating the performance of the automatic model built for this study, we discovered that its precision and recall were both 82% in comparison to manual review. This study is significant because a model capable of conducting PPRA compliance checks has been developed.

TABLE OF CONTENTS

CHAPTER 1

1. INTRODUCTION

1.1 Study Background:

The construction industry is a major economic pillar in many countries, contributing significantly to GDP and employment rates. When compared to other sectors, construction is often seen as "lagging behind" in terms of innovation because of the unique nature of its operations, which are often project-based, resource-intensive, and risk-related [1]. Bidding is one of the most essential components of any construction project. The conventional Bidding Process involves multiple stakeholders, including the client, architectural and engineering firms, general contractors, specialized contractors, material suppliers, manufacturers, etc. [2]. The construction sector as a whole has suffered as a result of the persistence of outdated practices, chief among which is the need to generate considerable amounts of paperwork when bidding on construction projects. For this reason, creating bidding paperwork has historically demanded a great deal of time, money, and resources from both the client and the contractors [3].

Electronic bidding (e-bidding) presents a possible solution for overcoming these challenges. An e-bidding system consists of electronically exchanging/transferring, publishing, communicating, accessing, receiving, and submitting all the bidding process through the medium of the internet, hence replacing the traditional paper-based bidding process. E-bidding, as opposed to a traditional paper-based bidding system, has been viewed as a less costly and less time-consuming system [4]. This can potentially make the business operation more efficient and effective for all the participating stakeholders. The adoption of this digital system has led to reductions in unproductive activities and expenses, including the need to print, scan, and transport documents [5]. Dissemination of project-related information, specifications, standards, and agreements, preparation and submission of an invitation to bid, pricing, and acceptance of completed bids are the primary features of e-bidding that may emerge in a typical construction project [6]. In a typical e-bidding process, the bid is initially entered into an online portal, where the bid details are automatically generated. The bid is received in a standard format by the contractor, who loads it into the estimating system. The contractor then prepares the bid and uploads it on the portal provided by the client.

The compliance checking of Bid documents during the e-bidding process is vital to successfully award the contract to the best bidder. Because the client submits bid documents to the contractor via an online platform, it is critical to ensure that the documents comply with local rules and regulations. It entails comparing the bid documents to specific predetermined standards. Currently, e-bidding is not being used to its full potential; compliance verification is done manually, increasing the possibility of error and necessitating a large workforce. Our research provides a framework for checking Bid documents' conformity with predefined criteria. This will ensure the E-Bidding process operates more efficiently.

As mentioned earlier, the initial phase in e-bidding is compliance checking, in which we regulate the bidding process using some predefined standards. PPRA regulates these standard laws in Pakistan. It monitors the entire bidding process for all public procurement. As the initial step of E-bidding, an automated system that verifies our bid's compliance with PPRA regulations will save a great deal of manpower, resources, and time. This automated method will ensure that the bidding process adheres to PPRA regulations. This system will take all bid documents, compare them to our application, and verify compliance with PPRA rules[7].

Due to the importance of PPRA clauses in construction contracts, this research presented an autonomous model of compliance monitoring based on rule-based extraction on natural language processing (NLP) to aid in contract management for construction firms[8].

The PPRA contract conditions have been extensively adopted as the standard contracts for construction projects in Pakistan for the past two decades. Government entities engaged in the procurement of construction projects also utilize these clauses. The PPRA contracts are renowned for outlining the duties, responsibilities, and risk management of the principal parties involved within a balanced framework. This research aims to autonomously identify and extract clauses that deviate from PPRA clauses and disadvantage contractors. In particular, the automatic extraction model is intended to detect intentionally and fraudulently modified clauses from the PPRA standard conditions, which are classified as violated clauses in this study.

1.2 Research Significance:

The significance of our work lies in the fact that it addresses the challenges encountered by the construction industry, specifically the cumbersome and paper-based nature of the bidding process, which hinders innovation and efficiency. Electronic tendering (e-bidding) has been proposed as a solution because it is considered to be less expensive and time-consuming, resulting in greater efficiency and effectiveness[9]. Nevertheless, compliance checking of proposal documents during the e-bidding process is currently performed manually, increasing the risk of errors and necessitating a substantial amount of manpower.

This research proposes a framework for automated compliance verification of proposal documents with predefined standards, specifically with the regulations of the Public Procurement Regulatory Authority (PPRA) in Pakistan, in order to improve the efficiency of the e-bidding process.

This research contributes to making the e-bidding process more efficient, resulting in improved project outcomes and improving the performance of the Pakistani construction industry as a whole.

The proposed framework employs machine learning algorithms to automate the conformance-checking procedure, thereby reducing human error and saving time and resources.[10] The findings of this study are applicable to policymakers, contractors, and other construction industry stakeholders in Pakistan and other nations confronting comparable challenges. The proposed framework can aid in streamlining the bidding procedure, reducing errors and costs, and enhancing the overall process's performance and effectiveness.

Furthermore, this research sheds light on the potential of machine learning and the use of Natural Language Processing (NLP) algorithms in automating compliance verification of bid documents [11]. This can contribute significantly to the advancement of the construction industry and the field of machine learning for compliance verification[12]. Therefore, this research is crucial for enhancing the overall performance and competitiveness of the construction industry by providing insights into the potential of these technologies to streamline and automate the compliance verification process, leading to cost and time savings, improved accuracy, and increased efficiency.

1.3 Problem Statement:

In the construction industry, the lack of automation in the compliance verification process of proposal documents during the e-bidding process results in significant time and cost expenditures, as well as an increased risk of errors. The cumbersome and paper-based nature of the tendering process, coupled with the manual nature of compliance checking, inhibits innovation and efficiency in the construction industry.

Consequently, the problem statement for this research is to propose a novel framework for automating the compliance verification process of proposal documents, specifically with the regulations of the Public Procurement Regulatory Authority (PPRA) in Pakistan in order to enhance the efficiency and effectiveness of the e-bidding process, reduce costs and time overheads, and enhance the overall performance of the construction industry.

The proposed framework seeks to streamline the compliance verification process, reduce human error, and conserve time and resources while ensuring conformity with predefined standards. This research aims to address the challenges encountered by the construction industry and enhance its competitiveness by revealing the potential of automation technologies for compliance verification.

1.4 Research Objectives:

The paper-based bidding method hinders the construction industry, which is essential to the economy. Compliance evaluation of proposal documents during e-bidding is crucial to contract award, but it is currently done manually, raising error risk and human resources. To overcome these issues, this thesis presents a framework for automated compliance verification of proposal documents with preset standards, specifically PPRA laws in Pakistan. E-bidding's potential to solve construction industry problems is crucial; hence, our study further strengthens and smoothens the e-bidding process. Natural Language processes (NLP) will be used to design and test the suggested framework. The study evaluates the proposed framework's influence on Pakistan's construction industry's performance and competitiveness and offers policymakers, contractors, and other stakeholders recommendations. Some of the significant research goals are:

- Identify the challenges faced by the construction industry in the bidding process and their impact on efficiency and innovation.
- Examine the potential of electronic tendering (e-bidding) as a solution to the challenges faced by the construction industry and its impact on efficiency and effectiveness.

- Propose a framework for automated compliance verification of proposal documents with predefined standards, specifically with the regulations of the Public Procurement Regulatory Authority (PPRA) in Pakistan, to improve the efficiency of the e-bidding process.
- Develop and test the proposed framework using natural language processing (NLP) techniques to ensure its effectiveness and efficiency in reducing errors and saving time and manpower.
- Evaluate the impact of the proposed framework on the performance and competitiveness of the construction industry in Pakistan and provide insights into its potential for other countries facing similar challenges.
- Provide recommendations for policymakers, contractors, and other stakeholders in the construction industry to enhance the efficiency and effectiveness of the bidding process and improve overall industry performance.

1.5 Limitation:

Some of the major limitations of our work are:

- The proposed framework may not be universally applicable to all construction projects and may require customization for specific contexts and regulatory requirements.
- The effectiveness of the proposed framework may depend on the quality and completeness of the bid documents and their compliance with the predefined standards, which may vary across contractors and projects.
- The proposed framework relies on NLP algorithms, which may not always accurately capture the nuances and complexities of human language and may generate false positives or false negatives in compliance verification.
- The cost and technical complexity of implementing and maintaining the proposed framework may be a barrier for small and medium-sized contractors who lack the necessary resources and expertise.
- The proposed framework may not entirely eliminate the need for human intervention and oversight, as some exceptional cases may require manual verification or judgment that cannot be automated.

- Currently, the prototype only pertains to the Bid advertisement and to some PPRA Rules and not to all bid documents or PPRA Rules.

1.6 Summary:

The construction industry faces challenges due to the paper-based and cumbersome nature of the bidding process, which hinders innovation and efficiency. Compliance checking of proposal documents during the e-bidding process is currently performed manually, increasing the risk of errors and requiring significant manpower. To address these challenges, this thesis proposes a framework for automated compliance verification of proposal documents with predefined standards, specifically with the regulations of the Public Procurement Regulatory Authority (PPRA) in Pakistan. The proposed framework employs machine learning algorithms and NLP techniques to streamline the compliance verification process, reduce human errors, and save time and resources while ensuring compliance with predefined standards. The research aims to evaluate the impact of the proposed framework on the performance and competitiveness of the construction industry in Pakistan and provide recommendations for enhancing the efficiency and effectiveness of the bidding process. The findings of this study are applicable to policymakers, contractors, and other construction industry stakeholders in Pakistan and other countries facing similar challenges. The proposed framework has the potential to automate compliance verification of bid documents, contributing significantly to the advancement of the construction industry and the field of machine learning for compliance verification. However, the proposed framework's limitations include customization requirements, reliance on the quality and completeness of bid documents, potential inaccuracies of NLP algorithms, and cost and technical complexity.

CHAPTER 2

2. LITERATURE REVIEW:

2.1 Benefits of E-Bidding:

Electronic bidding is a modern electronic way of conducting competitive bidding. Electronic platforms are used to link buyers and providers for online bidding. E-bidding has grown in popularity in recent years due to its many advantages, like higher process quality, lower procurement expenses, greater user satisfaction, faster responsiveness, improved customer service, less staff time needed, and greater managerial efficiency.

The traditional bidding process involves many processes, such as receiving, checking, copying, and distributing paper drawings, specifications, and a bill of quantities (BoQ), and is time-consuming and costly even for small construction projects. Technological e-bidding systems offer significant advantages over traditional bidding methods, reducing paperwork, postage, and photocopying while decreasing costs and saving time. E-bidding eliminates potential errors through system checks and provides a reliable and expedient exchange of information among bidders[13]. Access to archived data expands the range of potential bidders, and standardized procedures enable closer contractor-subcontractor relationships. E-bidding streamlines the bidding process, improving accuracy, transparency, and stakeholder alignment while broadening the candidate pool by making bid details easily accessible. Overall, e-bidding transforms the once paper-intensive and time-consuming process into a simplified, efficient, and optimized electronic workflow that delivers greater value across the procurement lifecycle.

To summarize, e-bidding provides various advantages to both buyers and suppliers, including cost savings, process improvement, user happiness, market growth, and shorter procurement cycle time. Companies may improve their competitiveness and market performance by taking advantage of the benefits of e-bidding.

2.2 Problematic Issues in e-Bidding:

E-bidding technology offers a number of benefits, including security, confidentiality, transparency, and cost reductions, that have enormous potential for use in procurement. However, these advantages are frequently overshadowed by fraudulent activities that exploit the system by favoring less-qualified bidders. In such corrupt environments, connections trump quality and determine the victors[14].

E-bidding's inherent complexity limits the number of qualified bidders and competition, resulting in a lack of diversity. Due to limited access to technology, insufficient skills, or the complexity of the bidding platforms, numerous qualified suitors are discouraged from participating. As a result, a few main competitors come to dominate the market.

Implementation and maintenance of secure electronic bidding systems also contribute to increased expenses, which are frequently passed on to purchasers in the form of additional fees. Training personnel on the technology and supplying the essential equipment are additional costs[15].

In addition, the absence of compliance checks for bidding documents based on standard norms, such as PPRA, necessitates manual review, resulting in higher costs, additional labor, and the possibility of partiality. The failure to verify compliance with applicable regulations can result in bid documents that do not satisfy the necessary criteria, posing problems later in the bidding and procurement process. Not only does the need for manual compliance review increase costs, but it also requires additional time and effort. If certain bidders' documentation is evaluated more leniently than others, this ultimately increases the overall costs for all parties involved and undermines the process's fairness. An efficient electronic bidding system with automated conformance checks would mitigate these drawbacks by reducing costs and labor while maintaining the integrity and objectivity of the bidding process.

To surmount these obstacles, procurement professionals must embrace and promote the development of e-bidding systems in a responsible manner. They can unlock the potential benefits of e-bidding by proactively addressing current issues, developing trust in the technology, and fostering inclusive and ethical implementation. E-bidding has the potential to revolutionize the procurement process by providing enhanced security, transparency, and value.

2.3 Models Used for Compliance Check:

Ensuring compliance with legal and regulatory standards is an essential process in the construction industry, finance, government, and numerous industries to guarantee individuals and organizations are abiding by legal and regulatory demands. Models for compliance checking have been created to automate this process, decreasing the necessity for manual inspection and raising efficiency. This article examines different models for compliance checking proposed in academic publications. Information retrieval (IR) and information extraction (IE) employ NLP extensively. Information retrieval (IR) is the process of locating information, typically unstructured text information, in a massive information repository. It is a very useful technology for extracting relevant information from enormous amounts of data. Unlike IR, IE is a process that extracts only pertinent information by defining what is to be extracted beforehand. If only the information corresponding to the name of a specific company is to be extracted from the documents, for instance, the IE method must first define the types and patterns of the information corresponding to the name of the specific company, and then the information corresponding to the patterns can be extracted. Defined IE systems as knowledge extraction systems for natural language texts. That IE only extracts the required data after devising extraction rules based on domain-specific expertise. Therefore, it is essential for the IE system to predefine the extraction elements and develop extraction algorithms. In previous research on IE, extraction elements were typically defined as named entities, relations between entities, and particular events. Rules based on pattern matching are used to extract meaningful information from text documents. These rules are a valuable method for determining the results when a particular sentence element's pattern is matched. Domain-specific pattern-matching rules can be defined in a variety of ways, but the majority can be expressed as "if pattern then result".

2.3.1 Rule-based Model

The rule-based model is one of the popular models for compliance checks. This model uses a set of predefined rules to check whether an action or process is compliant with regulations[16]. Rule-based NLP relies on explicitly programmed linguistic rules to analyze and represent text data. These rules are continually developed and refined to boost the accuracy of text processing. Machine learning-based NLP uses machine learning models trained on text features of sample texts. Rule-based NLP tends to yield higher precision and recall but requires more human labor. Hence, in this research, a rule-based approach was chosen for its likely superior performance. The model has proven effective in spotting compliance violations though limited by regulation complexity and the need for manual rule creation. Three main techniques of interest for compliance checking: process mapping and visualization, conformity assessment, and logic-based property[17].

2.3.2 Supervised Learning Model

Another compliance check method uses a trained learning model. This model applies machine learning algorithms to labeled sample data to learn how to recognize compliance violations. Then, it uses this knowledge to examine new data and identify violations. This method has been proven to find compliance violations effectively. However, it requires a large amount of sample data with known labels for training. The quality of the training data can also limit how well this model performs.

2.3.3 Unsupervised Learning Model

The model uses machine learning algorithms to spot patterns and oddities in data without needing labeled training data. This helps identify possible compliance violations. While this model can find compliance violations, it has limits. Workers must inspect flagged oddities to decide if they are actual violations.

2.3.4 Hybrid Model

The Hybrid model combines rule-based modeling, supervised learning, and unsupervised learning. This complex approach first uses defined rules to filter out non-compliant data. After that initial screening, the Hybrid model then applies two different machine learning algorithms to the remaining data to identify compliance issues. Supervised learning algorithms are trained using labeled known examples to spot similar patterns. Unsupervised learning algorithms are trained without labeled data and can recognize previously unknown patterns or groupings within the data. This Hybrid approach has proven to effectively identify compliance issues, though implementing it properly may require more effort than using just one machine learning technique alone. The Hybrid model benefits from the precision of rule-based screening to remove obvious non-compliant cases, followed by the power of supervised and unsupervised machine learning to discover harder-to-define violations within the remaining data. In short, while complex, the Hybrid approach aims to identify the most compliance violations possible through a multi-stage process of rules, supervised learning, and unsupervised learning. Additionally, the Hybrid paradigm has the benefit of being interpretable and explicable. The rule-based component provides transparency through the use of explicit rules that are readily understood and reviewed by compliance specialists. This transparency serves to justify and validate the process of compliance assessment. In addition, the supervised and unsupervised learning components can provide insights into the underlying patterns and associations within the data, providing compliance analysts with valuable information to comprehend the causes of compliance violations and make informed decisions. The interpretability of the Hybrid model enhances its applicability in regulatory audits, investigations, and decision-making processes, where a clear comprehension of compliance assessments is essential.

2.3.5 Multi-criteria Decision Analysis Model

The model uses a decision-making process that looks at multiple factors to determine compliance. It considers compliance based on different criteria.

The first important step is identifying what criteria are relevant for deciding how much compliance there is. The model looks at many different factors to determine compliance.

After determining the criteria, the model requires a method for making decisions based on the criteria. The weights or ratings assigned to the criteria are determined by how significant they are for compliance. Some criteria may also have minimum thresholds.

Based on how the circumstance performs on the various criteria, the decision-making process computes an overall compliance score. When all relevant criteria are considered, the outcome reflects how much compliance there is.

The approach tries to provide a more thorough and accurate evaluation of compliance than techniques that employ only one element by examining compliance from numerous criteria.

2.3.6 Semantic Analysis

Semantic analysis is the process of comprehending unstructured text data's implied meaning. Human language should not only be used according to the form of the language, but also be able to convey a precise meaning; thus, even a formally (grammatically) perfect sentence is ineffective if it is devoid of meaning. In a representative work in the field of syntactic structures, the following two sentences elucidate semantic analysis.

These two sentences contain the same terms, and the first sentence has an incomprehensible and grammatically incorrect sentence structure. Alternatively, although the second sentence contains the same words as the first, it is grammatically correct. However, the sentence is difficult to interpret in terms of its intended meaning because the second sentence is grammatically correct but semantically incomplete.

Thus, linguistic knowledge (such as lexical, syntactic, and semantic language) is utilized when humans use natural language for communication. In order for a machine to comprehend natural language, the contextual meaning must be understood through a semantic analysis as opposed to lexical and syntactic analyses alone. Even if they contain the same word combination, sentences in contracts may have distinct meanings depending on the order of their vocabulary combinations. Consequently, the purpose of this study is to construct a semantic analysis-based risk-information extraction model that is more accurate and efficient than a shallow NLP model.

2.3.7 Semantic Web Model

The idea of using a Semantic Web model for describing rules and compliance needs has some validity. The basic idea is that semantic technologies, which essentially enable data and information to be defined and linked in a machine-readable format, can be applied to how regulations and compliance rules are formally specified.

This type of formal representation of regulations and compliance requirements could enable a "compliance check" model where systems can automatically determine if a particular entity or situation complies with relevant rules. Such a system could check for compliance in a more thorough and accurate manner than what humans can do.

The rules and concepts in the domain of ontology help us understand the main ideas and connections within the topic of the text. This background can guide how information is pulled out by giving context to the terms and meanings used. Using meanings along with wording allows the information extraction to utilize this context knowledge, potentially getting more correct and complete results. Zhang and El-Gohary's study shows this clearly. Their precision went from 75% correct to 100% correct, and their recall went from capturing 75% to 95% of the critical information when they used semantic information extraction instead of just syntax[18].

2.3.8 Dependency Parser

Syntactic analysis is a method for structuring the intuitive logical process by which humans comprehend the sentence structure. Syntactic analysis typically employs two techniques: phrase-structure grammar (PSG) and dependency grammar (which analyzes sentences according to the dominance–dependence relationship between words).

In this paper, syntactic analysis is performed using dependency grammar. Using dependency grammar, it is possible to comprehend information about the subject and object of each sentence based on their relationship to the verb. This information is a crucial clue for identifying the information to be extracted, so it is simpler to identify the information to be extracted. Parsing was performed using Parsey McParseface, a dependency-grammar parser of SyntaxNet, an open-source model developed by Google. As the framework of a neural network, SyntaxNet is a potent machine-learning algorithm that performs NLP in artificial intelligence. When given a sentence as an input value, the SyntaxNet parser determines the grammatical role of each word by dividing each sentence according to the word level, designating a tag, and expressing the dependence of the words as the dependency parser tree. An example of SyntaxNet's dependency parser tree. Google SyntaxNet's Application Programming Interface (API) was utilized in conjunction with Python programming for the syntactic analysis of contracts.

2.4 E-Bidding in Construction Sector: A Global Perspective

Various countries have used electronic bidding to improve openness, efficiency, and competitiveness in public procurement, notably in the construction sector. In the United States, for example, the Federal Business Opportunities (FedBizOpps) website allows contractors to electronically search and see federal government construction-related solicitations. E-procurement is controlled in the European Union by the EU Public Procurement Directive, which requires the use of electronic methods for procurement over specific levels. Several European nations, notably Germany, France, and the United Kingdom have established e-procurement systems that allow for the electronic filing of bids and proposals for building projects. The government of India created an e-procurement platform in 2006, and the usage of e-bidding has progressively increased since then, particularly for building projects. Similarly, the Public Procurement Regulatory Authority (PPRA) in Pakistan has created an e-procurement system, which includes electronic bidding for building projects, which has helped eliminate corruption and promote transparency in the industry. While the use of e-bidding differs by country, it is clear that it has become an increasingly important part of public procurement procedures in the construction industry, and many countries are actively seeking to grow and enhance their e-procurement systems in this field.

2.4.1 Singapore:

Singapore has taken the lead in adopting online construction bidding. They have done a lot to improve the procurement process's effectiveness and efficiency.

There's the GeBIZ system, for example. It is an online marketplace where businesses may submit bids, have their offers examined, and win contracts.

Singapore has several safeguards in place to ensure that corporations respect the regulations. One is the Contractor Registration System. Companies must register, helping confirm only reputable rule-abiding firms can bid on government projects[19]. Singapore employs a risk-based approach to inspections as well. It is concerned with detecting and mitigating risks in the procurement process. They use data analysis to identify high-risk zones. They then do checks to remedy the faults. The Building Authority (BCA) is in charge of regulating and supervising the building sector. The BCA created construction standards and guidelines. This helps to guarantee that projects reach a high standard of quality.

In summary, Singapore's checks for online bidding aim to leverage technology and data analysis to identify and fix risks. They confirm only reputable rule-abiding companies can bid. This helps projects finish on time, within budget, and at a high-quality level.

2.4.2 Saudi Arabia

Saudi Arabia has also made significant advances in implementing electronic bidding in the construction business. To assist all government institutions with their procurement operations, the Saudi government built the "Etimad" platform. The platform provides an online marketplace where vendors can participate in bidding procedures like prequalification and electronic bid submission. The Etimad platform also features an automatic compliance check capability for bidding papers, which helps to guarantee that all submissions meet the relevant technical specifications and other standards[20].

Etimad's compliance check tool analyses bidding papers and flags any inconsistencies or noncompliance concerns using an algorithm. Incomplete forms, missing signatures, and erroneous or inadequate information are all flagged by the system. Saudi Arabia has also made significant strides in implementing e-bidding in the construction sector. To simplify procurement procedures across all government agencies, the Saudi government built the "Etimad" platform. The platform offers an online marketplace where suppliers may participate in bidding procedures such as prequalification and electronic bid submission. The Etimad platform also contains a tool for automatically evaluating bidding documents for compliance, assisting in ensuring that all submissions meet the necessary technical requirements and other criteria.

Etimad's compliance check tool analyses bid documents using an algorithm to identify any discrepancies or noncompliance issues. The system looks for things like missing signatures, incomplete forms, and incorrect or inadequate information. If any problems are detected, the supplier is informed.

The proposed electronic bidding system for public construction contracts in Saudi Arabia has three parts: the components, framework, and category of the model. For an efficient bidding system, all parts must work together smoothly. The steps it follows are similar to traditional bidding to make the transition easy for people. The reason for this is to smoothly transition people to the new electronic system[20].

1. Publication and Announcement
2. Prequalification
3. Communication Tools
4. Documents Management

5. Blogging
6. Submission of the offers
7. Analyzing the Bids

2.5 Research Gaps in E-bidding Compliance Check: A Review of Literature

The concept of E-Bidding is not a novelty in this modern world of advanced technology. Many researchers have previously worked on E-Bidding Processes and ways to improve their efficiency.

There are numerous articles on the benefits of E-Bidding, but there are only a few articles that focus on problems and the implementation of it; each bidding system may have a different or unique process, which means that the issues identified are specifically for one certain system.

Current online bidding processes applied in construction projects lack the intelligence to fully automate the bidding process. They fail to effectively facilitate the electronic exchange of documents between parties due to insufficient integration between different systems, formats, and applications. As a result, no existing solution has completed the loop of an end-to-end intelligent bidding process.

CHAPTER 3

3. METHODOLOGY

Figure 3.1 depicts the entire process of creating the compliance checking program (CCP) in stages: (1) Examining PPRA rules and extracting those rules which are violated frequently (2) Collection of bids from the user in the specified template (3) Feeding the file to prototype and Extraction of data using python (4) Compliance of extracted data with pre-built PPRA rules in the prototype.

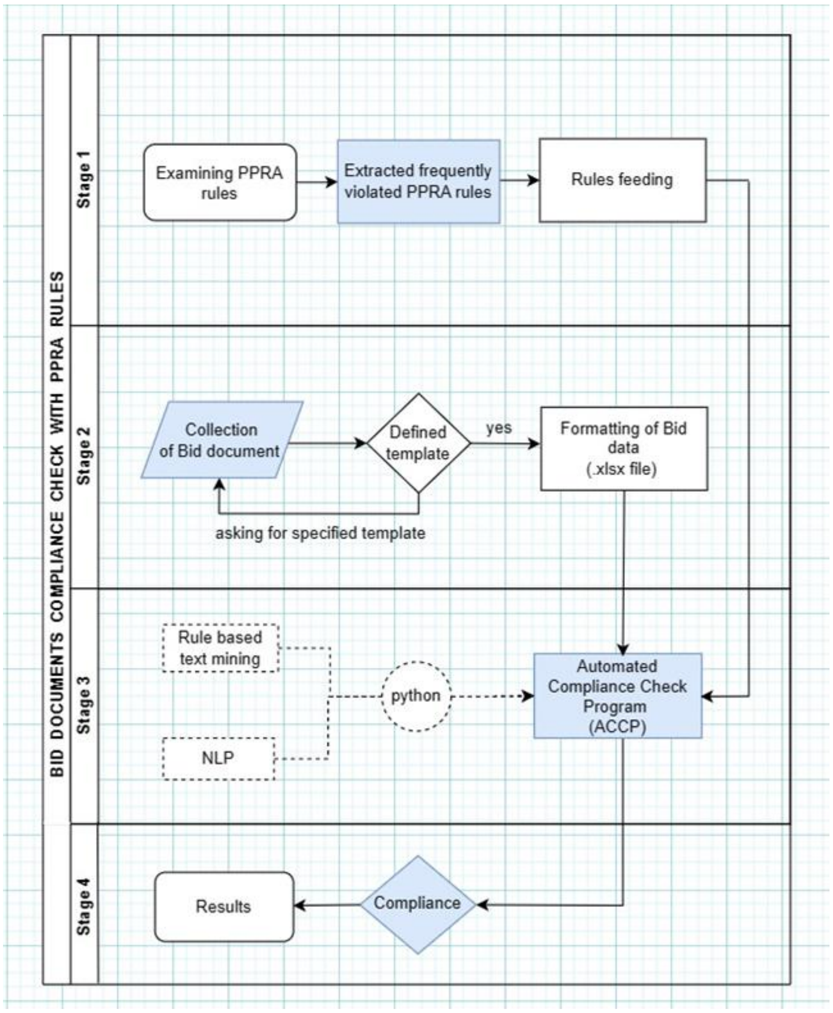

BID DOCUMENTS COMPLIANCE CHECK WITH PPRA RULES
Stage 1
Examining PPRA rules
Extracted frequently violated PPRA rules
Rules feeding
Stage 2
Collection of Bid document
Defined template
yes
Formatting of Bid data (.xlsx file)
asking for specified template
Stage 3
Rule based text mining
NLP
python
Automated Compliance Check Program (ACCP)
Stage 4
Results
Compliance

Figure: 3.1 Flowchart of Methodology

To develop an automatic model that extracts the violated clauses from the advertisement, it is necessary to define beforehand the information to be extracted as violated clauses. In this section, we identify the violation factors in the documents and define the violation-related sentences and expressions that correspond to the violated clauses in order to obtain the raw data necessary for constructing and learning the model's extraction rules. Due to the fact that this study concentrates more on the violation-extraction methodology than on the violated identification in the documents, the sentences related to the hazardous violated clauses defined in this study do not account for all potential risk factors. Therefore, the purpose of this study is to support existing contract-management duties by automatically extracting the contract risks that frequently occur in international construction contracts. In addition, since clauses deviating from the PPRA standard conditions for contracts that are adverse to the contractor are defined as contract risks, the risk level of the PPRA contracts themselves is not encompassed by the contract risk. In addition, a violation-free clause is defined in this paper as a clause that has not been altered from the PPRA standard conditions.

3.1 Examining PPRA Rules

Understanding PPRA rules is the first step in developing the Compliance Checking Program (CCP). These regulations establish the legal framework for procurement in Pakistan and regulate the tendering process for construction projects. Compliance requirements are specified for the bid documents based on PPRA regulations. The examination of bidding advertisements and PPRA regulations led us to the conclusion that many rules are frequently violated, though not all. Therefore, we decided to include them in CCP for their violation check-in documents.

3.2 Problem Identification

It entails identifying the specific compliance issues that are frequently violated in Pakistani construction project proposal documents. In order to identify these issues, the PPRA rules and regulations were analyzed to determine the rules that were violated most frequently.

On the basis of an examination of the PPRA rules and regulations, specific issues that are frequently violated in Pakistani construction project proposal documents were identified. Among the most frequently violated compliance issues in the procurement compliance area are the following:

- Misrepresentation of Information
- Bid submission after the deadline
- No Bid showing on online platforms
- Violation of bid security amount

These compliance issues were then utilized to construct the rules that the prototype for compliance checking would use to evaluate the bid documents. The code will extract necessary data that will be used to verify that documents comply with PPRA regulations.

3.3 Why Python

Python is a potent and popular programming language used extensively for data processing and analysis, machine learning, and natural language processing. It is highly regarded for its simplicity, legibility, and usability, making it an ideal language for rapidly and efficiently developing prototypes and applications. Natural language processing (NLP) capabilities are one of Python's most notable characteristics. Python offers a number of potent NLP libraries, such as NLTK and SpaCy, that are designed to perform a variety of tasks, including tokenization, lemmatization, and named entity recognition, among others. These libraries have made Python an attractive option for text analysis and processing duties, such as the development of the Compliance Checking Program (CCP).

Compliance checking requires analyzing text-based documents against a set of predefined rules or regulations, making the use of Python's NLP libraries in our CCP particularly pertinent. We utilized NLP libraries to extract pertinent information, such as dates, names, and keywords, from these documents, which can then be compared to the predefined compliance standards. In addition, Python's compatibility with other programming languages and data sources makes it an attractive option for CCP.

In addition, Python's NLP processing capabilities and interoperability with other programming languages and data sources make it a potent and efficient choice for developing prototypes for conformance checking. This Python feature assisted us in developing the front end of our CCP in conjunction with JavaScript.

3.4 Extraction and Compliance Checking

The compliance checking Program (CCP) employs the Python programming language to extract data from solicitation documents and validate its conformance with the PPRA's built-in Python standards.

To extract information from the proposal documents, we used the docx2txt and pypdf libraries, which convert Word documents and pdfs into plain text format, respectively. This enabled us to readily extract text data from the documents, such as project details, contractor information, bid prices, etc. After extracting the data, we processed the text using the spaCy and nltk libraries to conduct natural language processing. Using rule-based text mining, the spaCy library allowed us to extract specific entities from text, such as dates, quantities, and contractor names, in any format. We used the numerizer library to convert numeric data into a standard format, such as converting various currencies into Pakistani Rupees (PKR), in order to verify that the extracted data met PPRA standards. We also used the spacy.matcher library to match specific patterns in the text data, such as matching organization names with the PPRA database of registered organizations that we provided to our code, followed by the use of if and else statements to perform compliance checks, using the flask framework, all the results are then displayed as compliance or noncompliance. APIs are used for backend and frontend communication.

3.5 Development of Extraction Rules

In the extraction rule development phase, the patterns of the to-be-extracted violation-related sentences were analyzed, and extraction rules were developed based on the risk patterns. According to their purpose, the extraction rules developed in this study can be categorized into three distinct types: (1) preprocessing rules, which are rules for simplifying sentence structures; (2) syntactic rules, for locating SVO tuple based on the grammatical structure of sentences; and (3) semantic rules, for context-based analysis utilizing the lexicon. The phases of rule development and implementation for contract-risk extraction. In the second phase (information extraction), the three categories of extraction rules (preprocessing rules, syntactic rules, and semantic rules) developed in the first phase (rule development) are applied.

3.6 Extraction Elements

The automatic-extraction model of violated clauses is the process of transforming unstructured text data into structured data by extracting valuable semantic information from a contract. This model employs an IE algorithm in which the computer recognizes and extracts the contract clauses pertaining to contract violations. In order for a computer to recognize target information in unstructured text data, the information that should be extracted must be identified first. In the majority of studies employing IE, it is typical to isolate only specific sentence elements, such as a named entity or an event. Since comprehending the context of a sentence is essential for determining whether or not it poses a risk, it is not sufficient to extract only a few elements from each sentence. Defined as a subject-verb-object (SVO) tuple that corresponds to the subject part (S), the verb part (V), and the object part (O), this study isolates all the elements that comprise each sentence. Thus, all elements of the sentence are allocated to the SVO tuple, preventing the generation of missing elements.

3.7 A step-by-step explanation of functions used

3.7.1 Step 1: User Uploads

The user initiates the procedure by uploading a document intended for construction bidding via the JavaScript-based front-end interface. The interface facilitates the user's selection of the desired file and initiates the required backend processes to manage the uploaded document.

A validation procedure is conducted to ensure the validity and suitability of the transmitted file. This validation phase ensures that the document follows the required structure and is not corrupted. If the validation procedure detects any issues, the user is presented with an error message indicating the problem immediately.

Given that this Construction Contracting Party (CCP) only accepts specific proposal document templates, the uploaded file is sent to the backend for processing. During this phase, the PyPDF library is used to facilitate data purification and preprocessing. PyPDF, a Python library designed particularly for manipulating PDF document content, is indispensable for performing tasks such as removing superfluous characters, resolving encoding issues, and preparing the data for subsequent processing stages.

In our implementation, we take advantage of a variety of functions and configurations offered by PyPDF and other complementary tools. These include the use of stop words to eradicate frequently used but insignificant words, lemmatizers to reduce words to their root form for consistent analysis, Tabula for extracting structured data from PDF tables, and numerizers for handling numerical data appropriately. These techniques collectively aid in cleaning and preprocessing the data in a manner that improves its quality and enables more precise analysis and interpretation.

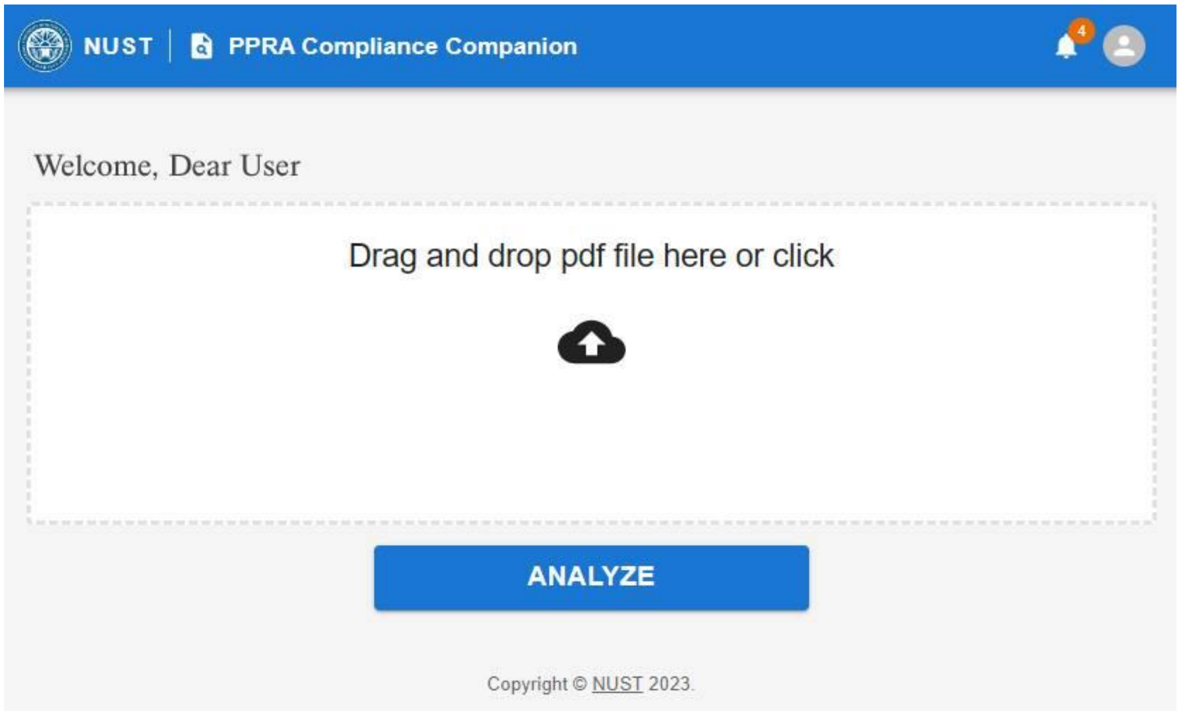
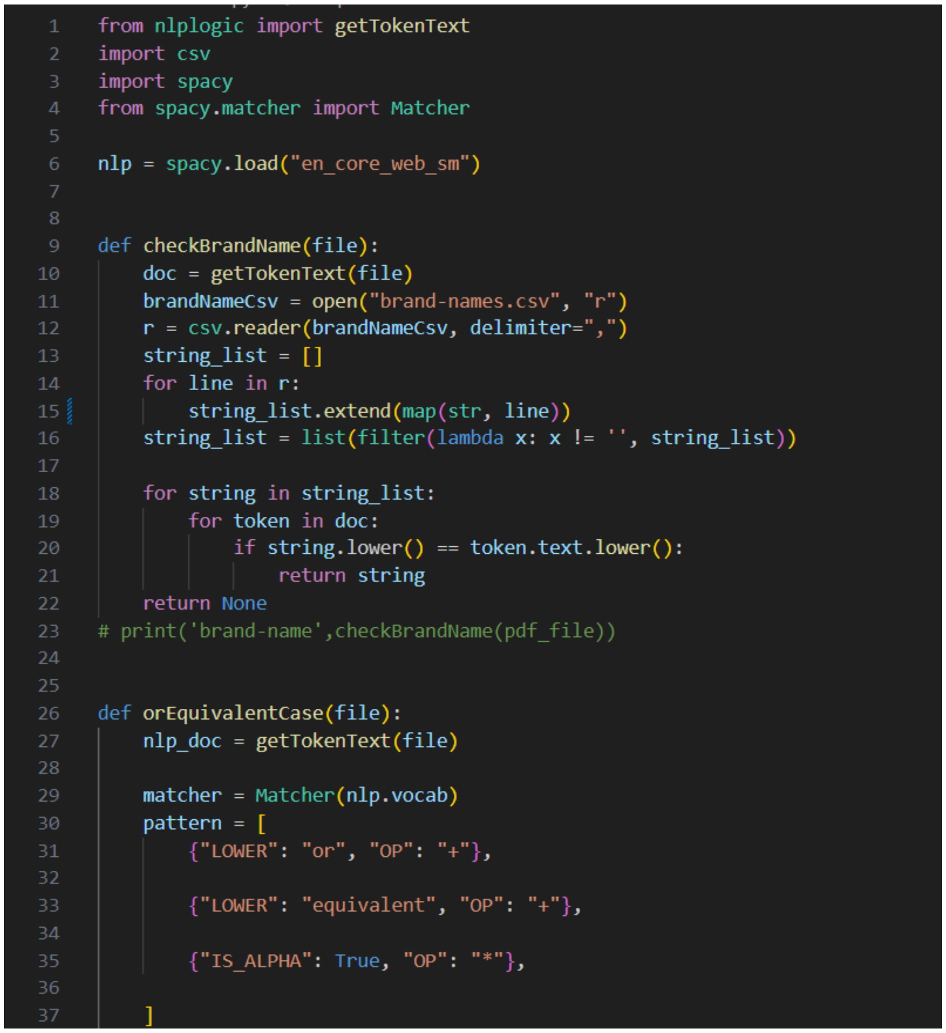

NUST
PPRA Compliance Companion
4
Welcome, Dear User
Drag and drop pdf file here or click
ANALYZE
Copyright © NUST 2023.

Fig 3.5.1 User uploads

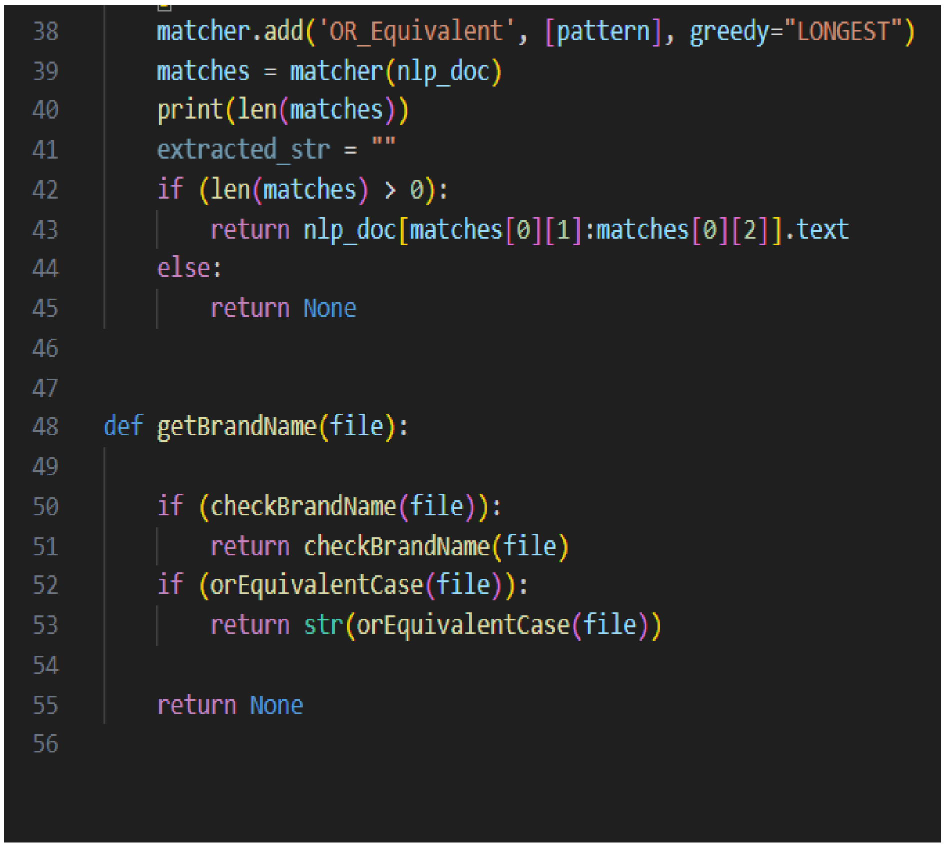
3.7.2 Step 2: Data Extraction

In the second phase of the compliance check program for bidding documents with Pakistan's PPRA rules, Natural Language Processing (NLP) libraries and rule-based models are used to extract data. This phase involves the use of the Spacy, NLTK (Natural Language Toolkit), Tabula, and CSV libraries. In addition, functions such as 'numerizer,' 'len," matcher.add()', 're,' and multiple iterative functions are utilized to extract the necessary information from the tendering documents. The purpose of this phase is to extract pertinent information that will be used for further analysis and compliance verification.



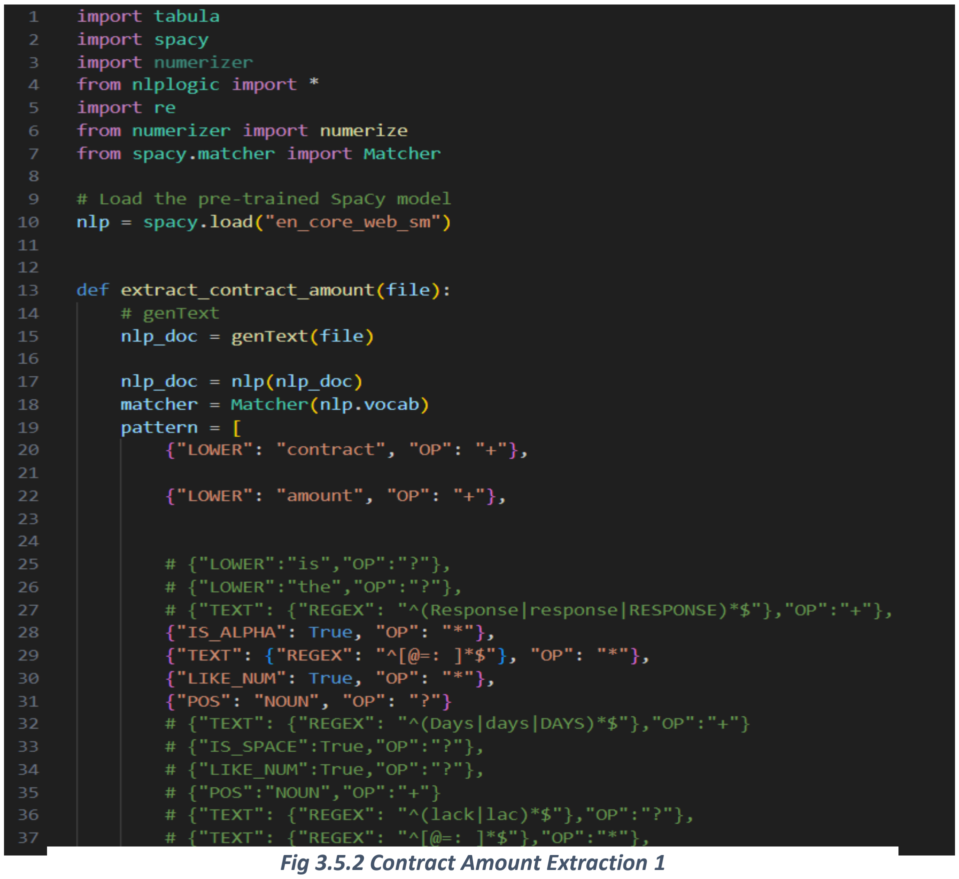
```python
from nlplogic import getTokenText
import csv
import spacy
from spacy.matcher import Matcher

nlp = spacy.load("en_core_web_sm")


def checkBrandName(file):
    doc = getTokenText(file)
    brandNameCsv = open("brand-names.csv", "r")
    r = csv.reader(brandNameCsv, delimiter=",")
    string_list = []
    for line in r:
        string_list.extend(map(str, line))
    string_list = list(filter(lambda x: x != '', string_list))

    for string in string_list:
        for token in doc:
            if string.lower() == token.text.lower():
                return string
    return None
# print('brand-name',checkBrandName(pdf_file))


def orEquivalentCase(file):
    nlp_doc = getTokenText(file)

    matcher = Matcher(nlp.vocab)
    pattern = [
        {"LOWER": "or", "OP": "+"},

        {"LOWER": "equivalent", "OP": "+"},

        {"IS_ALPHA": True, "OP": "*"},

    ]
```

Fig 3.5.2 Brand Name Extraction 1



```python
    matcher.add('OR_Equivalent', [pattern], greedy="LONGEST")
    matches = matcher(nlp_doc)
    print(len(matches))
    extracted_str = ""
    if (len(matches) > 0):
        return nlp_doc[matches[0][1]:matches[0][2]].text
    else:
        return None


def getBrandName(file):

    if (checkBrandName(file)):
        return checkBrandName(file)
    if (orEquivalentCase(file)):
        return str(orEquivalentCase(file))

    return None

```

Fig 3.5.2 Brand Name Extraction 2

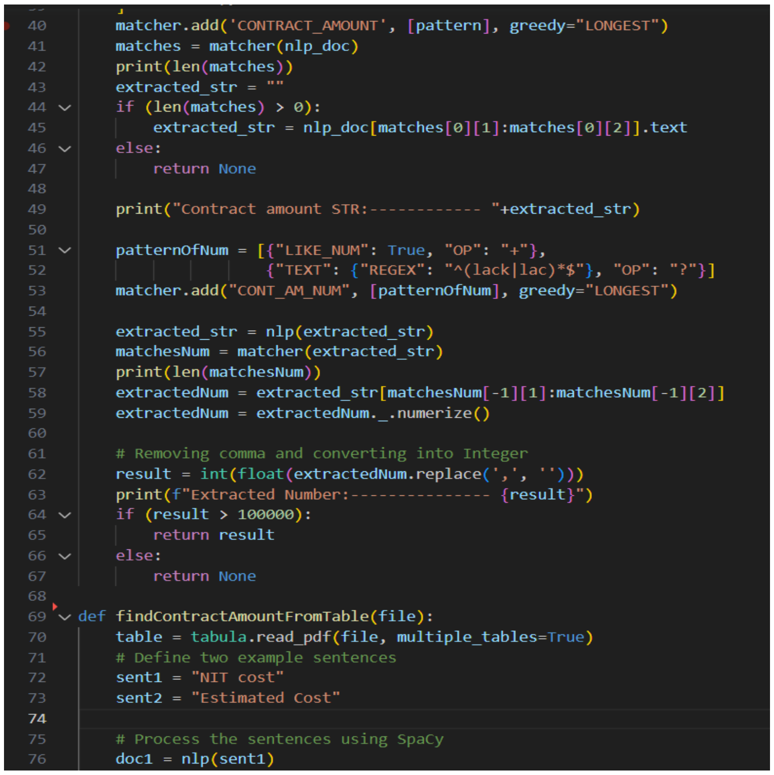


```python
import tabula
import spacy
import numerizer
from nlplogic import *
import re
from numerizer import numerize
from spacy.matcher import Matcher

# Load the pre-trained SpaCy model
nlp = spacy.load("en_core_web_sm")


def extract_contract_amount(file):
    # genText
    nlp_doc = genText(file)

    nlp_doc = nlp(nlp_doc)
    matcher = Matcher(nlp.vocab)
    pattern = [
        {"LOWER": "contract", "OP": "+"},

        {"LOWER": "amount", "OP": "+"},


        # {"LOWER":"is","OP":"?"},
        # {"LOWER":"the","OP":"?"},
        # {"TEXT": {"REGEX": "^(Response|response|RESPONSE)*$"},"OP":"+"},
        {"IS_ALPHA": True, "OP": "*"},
        {"TEXT": {"REGEX": "^[@=: ]*$"}, "OP": "*"},
        {"LIKE_NUM": True, "OP": "*"},
        {"POS": "NOUN", "OP": "?"}
        # {"TEXT": {"REGEX": "^(Days|days|DAYS)*$"},"OP":"+"}
        # {"IS_SPACE":True,"OP":"?"},
        # {"LIKE_NUM":True,"OP":"?"},
        # {"POS":"NOUN","OP":"+"}
        # {"TEXT": {"REGEX": "^(lack|lac)*$"},"OP":"?"},
        # {"TEXT": {"REGEX": "^[@=: ]*$"},"OP":"*"},
```

Fig 3.5.2 Contract Amount Extraction 1

The tendering documents are preprocessed using Spacy and NLTK to initiate the data extraction procedure. Spacy is a robust library for natural language processing with sophisticated features for tokenization, part-of-speech tagging, and entity recognition. NLTK, on the other hand, offers additional resources and tools for NLP duties. After the documents have been preprocessed, rule-based models are used to identify particular patterns or entities related to PPRA rules. These models are developed utilizing the matcher function of Spacy, which permits the definition of custom rules and patterns to match against the text. These criteria are added to the matcher object using the'matcher.add()' function. Tabula, a library for extracting tables from PDFs, is then used to extract tabular data from the solicitation documents. This is especially helpful for retrieving structured data, such as item descriptions, quantities, prices, and other pertinent information required for compliance verifying.

The 'numerizer' function is utilized to manage numerical values within the documents. This function converts written numbers to their numerical representation, making comparisons and analyses simpler. It can convert "three" to "3" and "twenty-two" to "22". Regular expressions ('re') are used to recognize and derive particular patterns or formats from text. Dates, contract numbers, and vendor names can be captured using these patterns.

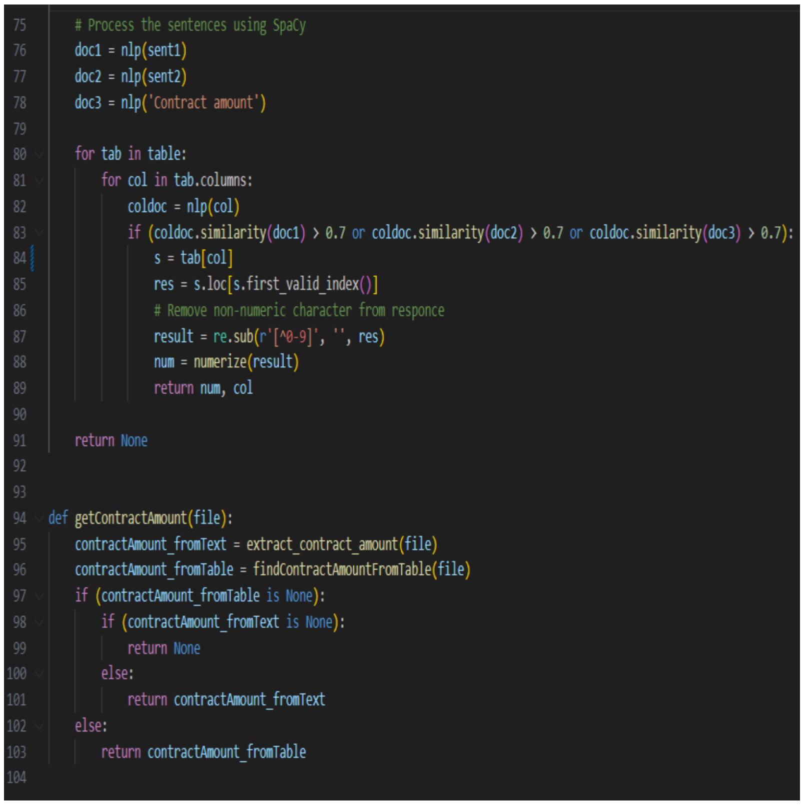


```
    matcher.add('CONTRACT_AMOUNT', [pattern], greedy="LONGEST")
    matches = matcher(nlp_doc)
    print(len(matches))
    extracted_str = ""
    if (len(matches) > 0):
        extracted_str = nlp_doc[matches[0][1]:matches[0][2]].text
    else:
        return None

    print("Contract amount STR:------------ "+extracted_str)

    patternOfNum = [{"LIKE_NUM": True, "OP": "+"},
                    {"TEXT": {"REGEX": "^(lack|lac)*$"}, "OP": "?"}]
    matcher.add("CONT_AM_NUM", [patternOfNum], greedy="LONGEST")

    extracted_str = nlp(extracted_str)
    matchesNum = matcher(extracted_str)
    print(len(matchesNum))
    extractedNum = extracted_str[matchesNum[-1][1]:matchesNum[-1][2]]
    extractedNum = extractedNum._.numerize()

    # Removing comma and converting into Integer
    result = int(float(extractedNum.replace(',', '')))
    print(f"Extracted Number:--------------- {result}")
    if (result > 100000):
        return result
    else:
        return None

def findContractAmountFromTable(file):
    table = tabula.read_pdf(file, multiple_tables=True)
    # Define two example sentences
    sent1 = "NIT cost"
    sent2 = "Estimated Cost"

    # Process the sentences using SpaCy
    doc1 = nlp(sent1)
```

Fig 3.5.2 Contract Amount Extraction 2

Multiple iterative functions are implemented during the extraction process to iterate over the document's content and extract relevant data. These functions enable the systematic retrieval of information from various sections or portions of the proposal documents, ensuring the Extraction of all relevant data. CSV library is used to provide built-in data for reference or comparison. This enables the program to access predefined data stored in CSV format, which can be used to cross-reference and validate the extracted data from the proposal documents.

Overall, the data extraction phase of the compliance check program is a crucial component for obtaining the required information from the proposal documents. Relevant data is extracted systematically using various libraries, rule-based models, and iterative functions, including structured information from tables, numerical values, patterns using regular expressions, and additional built-in data for comparison purposes. This extracted data will serve as the basis for subsequent stages, such as compliance checking and analysis, which will ensure adherence to Pakistan's PPRA regulations.



```python
    # Process the sentences using SpaCy
    doc1 = nlp(sent1)
    doc2 = nlp(sent2)
    doc3 = nlp('Contract amount')

    for tab in table:
        for col in tab.columns:
            coldoc = nlp(col)
            if (coldoc.similarity(doc1) > 0.7 or coldoc.similarity(doc2) > 0.7 or coldoc.similarity(doc3) > 0.7):
                s = tab[col]
                res = s.loc[s.first_valid_index()]
                # Remove non-numeric character from responce
                result = re.sub(r'[^0-9]', '', res)
                num = numerize(result)
                return num, col

    return None


def getContractAmount(file):
    contractAmount_fromText = extract_contract_amount(file)
    contractAmount_fromTable = findContractAmountFromTable(file)
    if (contractAmount_fromTable is None):
        if (contractAmount_fromText is None):
            return None
        else:
            return contractAmount_fromText
    else:
        return contractAmount_fromTable

```

Fig 3.5.2 Contract Amount Extraction 3

3.7.3 Step 3: Compliance Check

The extracted data endures a comprehensive evaluation during the compliance check phase of the bidding documents' compliance program with PPRA regulations in Pakistan. In addition to predefined PPRA principles incorporated into the program's code, Spacy's matcher is utilized. Using if-else functions, the status of compliance is determined. Using Spacy's matcher, PPRA compliance requirements are reflected by defining specific patterns and regulations. These patterns consist of entities, keywords, and phrases that indicate compliance or noncompliance. The PPRA's predefined standards serve as a guide for assessing compliance. Incorporated as built-in data, they define PPRA compliance guidelines and criteria.

If-else functions compare extracted data to predetermined principles. Using the matcher, the presence or absence of specified patterns or entities is evaluated. Based on the outcome, appropriate actions or notifications are triggered. Successful compliance allows progression to the next phase or initiates a compliance notification. Noncompliance generates alerts that call attention to areas that require attention. If-else functions can manage exceptions and various scenarios, taking into consideration thresholds or tolerance levels in compliance evaluation. The compliance check ensures that the bidding procedure adheres to PPRA regulations. Using Spacy's matcher, predefined rules, and if-else functions, the program assesses compliance in a systematic manner. Identified noncompliance areas are subject to appropriate actions. Maintaining an effective compliance check program necessitates routine reviews and revisions to keep up with the ever-changing PPRA regulations.

3.7.4 Step 4: Showing Output

In the concluding phase of the compliance check program for bidding documents with Pakistan's PPRA rules, the program's outputs are displayed through an intuitive front end. This step entails web development with the Flask framework, APIs, CORS (Cross-Origin Resource Sharing), and functions like 'getall()'. Users are able to upload PDF files, which are then analyzed in the background. The results are displayed, signifying which rules have been broken or adhered to.

- Flask, a Python web framework, is utilized to facilitate the user interface and interaction. Flask enables the development of web applications by providing a solid framework for managing requests and responses. It permits seamless integration between the program's interface and backend components.
- Application Programming Interfaces (APIs) are used to facilitate communication between the UI and backend. These APIs allow data and queries to be transferred from the user interface to the backend processing.
- Cross-Origin Resource Sharing (CORS) is implemented to manage potential web browser security restrictions. It enables the UI to access and retrieve data from the backend in a secure manner, ensuring a seamless user experience.
- The 'getall()' function is used to retrieve and process submitted PDF files on the server's backend. This function accesses the extracted relevant data and conducts additional analysis to ascertain compliance status.

After the analysis is complete, the program generates an output that specifies which principles are violated or adhered to by the bid documents. This information is then displayed on the frontend, giving users a thorough understanding of the compliance status. The web development aspect of this phase ensures that users can interact with the program with ease, upload their bid documents, and receive immediate feedback on compliance. The incorporation of Flask, APIs, CORS, and functions such as 'getall()' enables the seamless transfer of data and results between the frontend and backend components. By displaying the conformance status of the bid documents, this step increases transparency and enables users to take corrective action in the event of noncompliance. It provides users with actionable insights and facilitates a seamless and efficient bidding process that adheres to Pakistan's PPRA rules.

Notably, continuous improvements and enhancements to the frontend interface may be required to provide a user-friendly and intuitive experience. Regular maintenance and revisions ensure that the program remains user-friendly and accessible.

3.8 Detailed Explanation for Rules Used for Extraction

3.8.1 Brand Name (Specification) Code Explanation:

The Code:

```python
from nlplogic import getTokenText
import csv
import spacy
from spacy.matcher import Matcher

nlp = spacy.load("en_core_web_sm")


def checkBrandName(file):
    doc = getTokenText(file)
    brandNameCsv = open("brand-names.csv", "r")
    r = csv.reader(brandNameCsv, delimiter=",")
    string_list = []
    for line in r:
        string_list.extend(map(str, line))
    string_list = list(filter(lambda x: x != '', string_list))

    for string in string_list:
        for token in doc:
            if string.lower() == token.text.lower():
                return string
    return None
```

Code Explanation:

1. **from nlplogic import getTokenText**: This line imports a function called **getTokenText** from a module called **nlplogic**.
2. **import csv**: This line imports Python's built-in CSV module which allows the program to read and write CSV files.
3. **import spacy**: This line imports the spaCy natural language processing library.
4. **from spacy.matcher import Matcher**: This line imports the Matcher class from the spaCy library which allows the program to match patterns in text.
5. **nlp = spacy.load("en_core_web_sm")**: This line loads a pre-trained spaCy model for English.

6. **def checkBrandName(file):**: This line defines a function called **checkBrandName** that takes a file as input.

7. **doc = getTokenText(file)**: This line calls the **getTokenText** function from the **nlplogic** module to get a spaCy Doc object representing the text in the file.

8. **brandNameCsv = open("brand-names.csv", "r")**: This line opens a CSV file called "brand-names.csv" in read mode and creates a file object called **brandNameCsv**.

9. **r = csv.reader(brandNameCsv, delimiter=",")**: This line creates a CSV reader object called **r** using the file object **brandNameCsv** and specifies that the delimiter in the file is a comma.

10. **string_list = []**: This line creates an empty list called **string_list**.

11. **for line in r:**: This line starts a loop that iterates over each row in the CSV file.

12. **string_list.extend(map(str, line))**: This line uses the **map()** function to convert each item in the row to a string and then adds those strings to the **string_list** using the **extend()** method.

13. **string_list = list(filter(lambda x: x != '', string_list))**: This line removes any empty strings from **string_list** using the **filter()** function and then converts the result to a list using the **list()** function.

14. **for string in string_list:**: This line starts a loop that iterates over each string in **string_list**.

15. **for token in doc:**: This line starts a nested loop that iterates over each token in the spaCy Doc object **doc**.

16. **if string.lower() == token.text.lower():**: This line checks if the lowercase version of the current string in **string_list** matches the lowercase version of the current token's text. If there is a match, it means that the current string is a brand name mentioned in the file.

17. **return string**: This line returns the brand name as a string if it is found in the file.

18. **return None**: This line returns None if no brand name is found in the file.

Code Continue:

```
def orEquivalentCase(file):
    nlp_doc = getTokenText(file)

    matcher = Matcher(nlp.vocab)
    pattern = [
        {"LOWER": "or" "OP": "+"},

        {"LOWER": "equivalent", "OP": "+"},

        {"IS_ALPHA": True, "OP": "*"},

    ]
    matcher.add('OR_Equivalent', [pattern], greedy="LONGEST")
    matches = matcher(nlp_doc)
    print(len(matches))
    extracted_str = ""
    if (len(matches) > 0):
        return nlp_doc[matches[0][1]:matches[0][2]].text
    else:
        return None


def getBrandName(file):

    if (checkBrandName(file)):
        return checkBrandName(file)
    if (orEquivalentCase(file)):
        return str(orEquivalentCase(file))

    return None
```

Code Explanation:

1. The **orEquivalentCase** function takes a file as input.
2. **getTokenText** function is called with the file, which returns the text of the file after tokenizing it.
3. The **Matcher** class from the **spacy.matcher** module is instantiated and passed the **nlp.vocab** object.
4. A **pattern** is defined as a list of dictionaries that specify the rules for the text matching. In this case, the pattern is composed of three dictionaries:
    - The first dictionary matches the lowercase string **"or"** one or more times.

- The second dictionary matches the lowercase string **"equivalent"** one or more times.
- The third dictionary matches any alphabetic character zero or more times.

5. The pattern is added to the **matcher** object with a label of **'OR_Equivalent'** and a greedy strategy of **'LONGEST'**. This means that the longest possible match will be returned if there are multiple matches.
6. The **matcher** object is called with the **nlp_doc** object, which returns a list of matches.
7. If there is at least one match, the text of the first match is returned as a string. Otherwise, **None** is returned.
8. The **getBrandName** function takes a file as input.
9. The **checkBrandName** function is called with the file. If it returns a non-**None** value, that value is returned.
10. The **orEquivalentCase** function is called with the file. If it returns a non-**None** value, that value is returned.
11. If both **checkBrandName** and **orEquivalentCase** return **None**, **None** is returned.

Code Summary:

This code aims to extract a brand name from a file. It follows a two-step approach. First, it checks if the brand name is explicitly mentioned in the file. If it finds a direct mention, it extracts and returns the brand name.

If no explicit mention is found, the code proceeds to the second step. It searches for the phrase "or equivalent" in the file. This phrase is commonly used to indicate that a specific brand name is not required, but an equivalent alternative is acceptable. If the phrase "or equivalent" is found, the code assumes that the preceding word or phrase represents the brand name, and it extracts and returns that word or phrase as the brand name.

In summary, the code attempts to identify the brand name by first looking for a direct mention and then considering the phrase "or equivalent" as an alternative indicator of the brand name.

3.8.2 Brand Names (CSV File):

Some of the names used in CSV files are

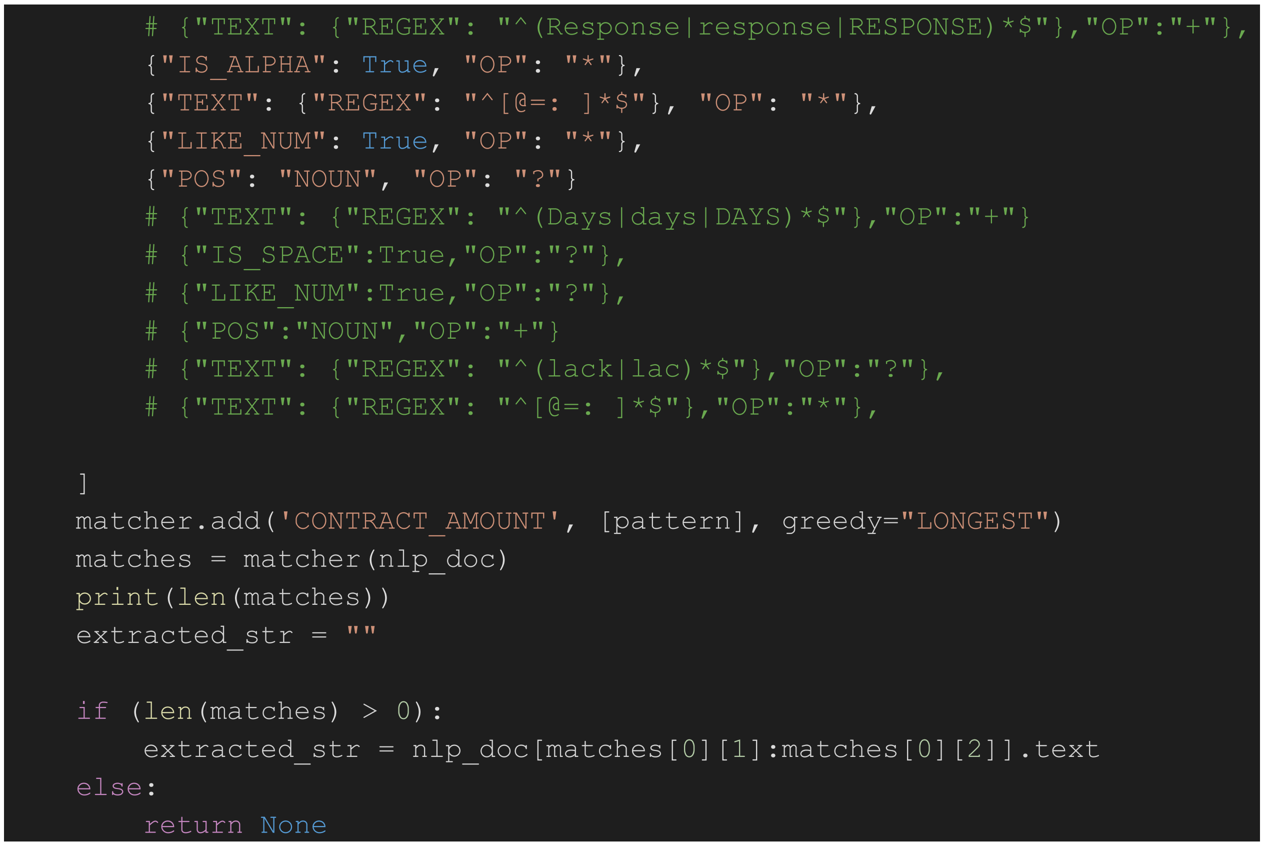
1. Lucky Cement Limited,
2. D.G. Khan Cement Company Limited,

3. Bestway Cement Limited,
4. Fauji Cement Company Limited,
5. Maple Leaf Cement Factory Limited,
6. Kohat Cement Company Limited,
7. Pioneer Cement Limited,
8. Cherat Cement Company Limited,
9. Attock Cement Pakistan Limit ed,
10. Thatta Cement Company Limited,
11. Caterpillar Inc,
12. Komatsu Ltd,
13. JCB,
14. Volvo Construction Equipment,
15. Hitachi Construction Machinery,
16. Liebherr Group,
17. Terex Corporation,

3.8.3 Contract Amount Code Explanation:

The Code:

```
import tabula
import spacy
import numerizer
from nlplogic import *
import re
from numerizer import numerize
from spacy.matcher import Matcher

# Load the pre-trained SpaCy model
nlp = spacy.load("en_core_web_sm")


def extract_contract_amount(file):
    # genText
    nlp_doc = genText(file)

    nlp_doc = nlp(nlp_doc)
    matcher = Matcher(nlp.vocab)
    pattern = [
        {"LOWER": "contract", "OP": "+"},

        {"LOWER": "amount", "OP": "+"},


        # {"LOWER":"is","OP":"?"},
        # {"LOWER":"the","OP":"?"},
```



```
        # {"TEXT": {"REGEX": "^(Response|response|RESPONSE)*$"},"OP":"+"},
        {"IS_ALPHA": True, "OP": "*"},
        {"TEXT": {"REGEX": "^[@=: ]*$"}, "OP": "*"},
        {"LIKE_NUM": True, "OP": "*"},
        {"POS": "NOUN", "OP": "?"}
        # {"TEXT": {"REGEX": "^(Days|days|DAYS)*$"},"OP":"+"}
        # {"IS_SPACE":True,"OP":"?"},
        # {"LIKE_NUM":True,"OP":"?"},
        # {"POS":"NOUN","OP":"+"}
        # {"TEXT": {"REGEX": "^(lack|lac)*$"},"OP":"?"},
        # {"TEXT": {"REGEX": "^[@=: ]*$"},"OP":"*"},

    ]
    matcher.add('CONTRACT_AMOUNT', [pattern], greedy="LONGEST")
    matches = matcher(nlp_doc)
    print(len(matches))
    extracted_str = ""

    if (len(matches) > 0):
        extracted_str = nlp_doc[matches[0][1]:matches[0][2]].text
    else:
        return None
```

Code Explanation:

1. **import tabula**: This imports the tabula module which is used to read PDF files and extract data from them.
2. **import spacy**: This imports the spacy module which is used for natural language processing.
3. **import numerizer**: This imports the numerizer module which is used to convert text representing numbers to numerical values.
4. **from nlplogic import ***: This imports all functions defined in the nlplogic module.
5. **import re**: This imports the re module which is used for regular expressions.
6. **from numerizer import numerize**: This imports the numerize function from the numerizer module.
7. **from spacy.matcher import Matcher**: This imports the Matcher class from the spacy.matcher module.

8. **nlp = spacy.load("en_core_web_sm")**: This loads the pre-trained SpaCy model for the English language.
9. **def extract_contract_amount(file):**: This defines a function named **extract_contract_amount** that takes a file as input.
10. **nlp_doc = genText(file)**: This calls the **genText** function from the **nlplogic** module to convert the PDF file into a text document.
11. **nlp_doc = nlp(nlp_doc)**: This passes the text document through the pre-trained SpaCy model to create a parsed document with linguistic annotations.
12. **matcher = Matcher(nlp.vocab)**: This initializes a Matcher object with the vocabulary of the pre-trained SpaCy model.
13. **pattern = [...]**: This defines a list of dictionaries that specify a pattern to match in the parsed document.

matcher.add('CONTRACT_AMOUNT', [pattern], greedy="LONGEST"): This adds the pattern to the Matcher object, giving it a name of 'CONTRACT_AMOUNT'.

14. **matches = matcher(nlp_doc)**: This applies the Matcher to the parsed document to find all matches for the pattern.
15. **print(len(matches))**: This prints the number of matches found.
16. **extracted_str = ""**: This initializes a variable named **extracted_str** to an empty string.
17. **if (len(matches) > 0):**: This checks if any matches were found.
18. **extracted_str = nlp_doc[matches[0][1]:matches[0][2]].text**: This extracts the text corresponding to the first match found by the Matcher.
19. **else: return None**: This returns **None** if no matches were found.

Code Continue:

```
print("Contract amount STR:------------ "+extracted_str)
```

```
    patternOfNum = [{"LIKE_NUM": True, "OP": "+"},
                    {"TEXT": {"REGEX": "^(lack|lac)*$"}, "OP": "?"}]
    matcher.add("CONT_AM_NUM", [patternOfNum], greedy="LONGEST")

    extracted_str = nlp(extracted_str)
    matchesNum = matcher(extracted_str)
    print(len(matchesNum))
    extractedNum = extracted_str[matchesNum[-1][1]:matchesNum[-1][2]]
    extractedNum = extractedNum._.numerize()

    # Removing comma and converting into Integer
    result = int(float(extractedNum.replace(',', '')))
    print(f"Extracted Number:--------------- {result}")
    if (result > 100000):
        return result
    else:
        return None
```

Code Explanation:

1. **print("Contract amount STR:------------ "+extracted_str)**: Prints a string indicating that the extracted string for the contract amount is being displayed.

2. **patternOfNum = [{"LIKE_NUM": True, "OP": "+"}, {"TEXT": {"REGEX": "^(lack|lac)*$"}, "OP": "?"}]**: Defines a pattern to match any sequence of numbers followed by an optional "lac" or "lack" (to account for values in Indian numbering system where "1 lac" is equivalent to "100,000").

3. **matcher.add("CONT_AM_NUM", [patternOfNum], greedy="LONGEST")**: Adds the above-defined pattern to the matcher with the label "CONT_AM_NUM".

4. **extracted_str = nlp(extracted_str)**: Converts the extracted string to a SpaCy **Doc** object for further processing.

5. **matchesNum = matcher(extracted_str)**: Matches the defined pattern with the extracted string and stores the results in **matchesNum**.

6. **print(len(matchesNum))**: Prints the number of matches found for the defined pattern.

7. **extractedNum = extracted_str[matchesNum[-1][1]:matchesNum[-1][2]]**: Extracts the substring that matches the defined pattern from the extracted string.

8. **extractedNum = extractedNum._.numerize()**: Converts the extracted number (in string format) to its numeric equivalent using the **numerize** function from the **numerizer** package.

9. **result = int(float(extractedNum.replace(',', '')))**: Removes any commas from the extracted numeric string and converts it to an integer.

10. **print(f"Extracted Number:--------------- {result}")**: Prints the extracted integer value.

11. **if (result > 100000):**: Checks if the extracted value is greater than 100000 (since the function is looking for contract amounts that are greater than this threshold).

12. **return result**: If the extracted value is greater than 100000, returns the integer value. Otherwise, returns **None**.

Code Continue:

```
def findContractAmountFromTable(file):
    table = tabula.read_pdf(file, multiple_tables=True)
    # Define two example sentences
    sent1 = "NIT cost"
    sent2 = "Estimated Cost"

    # Process the sentences using SpaCy
    doc1 = nlp(sent1)
    doc2 = nlp(sent2)
    doc3 = nlp('Contract amount')

    for tab in table:
        for col in tab.columns:
            coldoc = nlp(col)
            if (coldoc.similarity(doc1) > 0.7 or coldoc.similarity(doc2) >
0.7 or coldoc.similarity(doc3) > 0.7):
                s = tab[col]
                res = s.loc[s.first_valid_index()]
                # Remove non-numeric character from responce
                result = re.sub(r'[^0-9]', '', res)
                num = numerize(result)
                return num, col

    return None
```

```python
def getContractAmount(file):
    contractAmount_fromText = extract_contract_amount(file)
    contractAmount_fromTable = findContractAmountFromTable(file)
    if (contractAmount_fromTable is None):
        if (contractAmount_fromText is None):
            return None
        else:
            return contractAmount_fromText
    else:
        return contractAmount_fromTable
```

Code Explanation:

1. Reads the PDF file **file** using **tabula.read_pdf()** and stores the resulting tables in the variable **table**.
2. Initializes three sentences using string literals **sent1 = "NIT cost"**, **sent2 = "Estimated Cost"**, and **doc3 = nlp('Contract amount')**.
3. Processes each sentence using SpaCy's **nlp()** and stores them in the variables **doc1**, **doc2**, and **doc3**.
4. Loops through each table in **table**, and then each column in the table.
5. Processes each column as a document using **nlp()** and stores the resulting document in the variable **coldoc**.
6. Checks if the similarity of **coldoc** to **doc1**, **doc2**, or **doc3** is greater than 0.7 using the **coldoc.similarity()** function. If it is, it retrieves the first row of the corresponding column and stores it in the variable **res**.
7. Removes any non-numeric characters from **res** using the **re.sub()** function and stores the result in the variable **result**.
8. Converts **result** into an integer using the **numerize()** function and stores the result in the variable **num**.
9. Returns a tuple containing **num** and the column in which it was found if **num** was found, or **None** if no matching column was found.

10. Calls **extract_contract_amount(file**) to attempt to extract the contract amount from the text of the PDF file.
11. Calls **findContractAmountFromTable(file)** to attempt to extract the contract amount from the tables of the PDF file.
12. If f**indContractAmountFromTable(file) returns None** and **extract_contract_amount(file)** also returns **None**, returns **None**.
13. If **findContractAmountFromTable(file)** returns **None** and **extract_contract_amount(file)** returns a number, **returns** that number.
14. If **findContractAmountFromTable(file)** returns a tuple containing a number and a column, returns that tuple.

Code Summary:

The code you described is focused on extracting the contract amount from a given file. It utilizes different techniques, including natural language processing (**NLP**), regular expressions, and a tool called **Tabula**, which is designed to extract tables from **PDF** documents. The goal is to identify and extract the contract amount accurately. Here's a breakdown of the code's functions and their roles:

NLP-based function: This function utilizes **NLP** techniques to process the text from the file. It may involve tasks like tokenization, part-of-speech tagging, and named entity recognition (**NER**). These techniques help identify numerical values or patterns that represent the contract amount.

Regular expression-based function: Regular expressions are powerful tools for pattern matching. This function uses regular expressions to search for specific patterns or formats that typically represent contract amounts, such as currency symbols, decimal numbers, or written numbers. It applies pattern matching algorithms to extract the relevant information.

Tabula-based function: Tabula is a tool that specializes in extracting tables from **PDF** files. In cases where the contract amount is presented in a table format, this function utilizes Tabula to extract the table from the PDF and identify the contract amount within the table structure.

Combination function: The final function combines the results from the previous techniques. It applies a prioritization logic, giving precedence to the **NLP**-based extraction, followed by the regular expression-based extraction, and then the Tabula-based extraction. By combining these results, the function returns the most likely contract amount as a numerical value.

In summary, the code employs a multi-step approach, utilizing **NLP**, regular expressions, and Tabula, to extract the contract amount from the given file. It combines the results from these techniques to improve accuracy and reliability in identifying the contract amount.

3.8.4 Bid Opening Rule:

The Code:

```
import spacy
from spacy.matcher import Matcher
import re
from nlplogic import getTokenText

# Load the pre-trained SpaCy model
nlp = spacy.load("en_core_web_sm")


def getBidTimes(file):
    nlp_doc = getTokenText(file)

    matcher = Matcher(nlp.vocab)
    # Define the pattern to match "hour" or "hours" with a four-digit
number before it
    pattern = [{"LIKE_NUM": True, "LENGTH": 4, "OP": "+"},
               {"LOWER": {"IN": ["hour", "hours"]}}]

    matcher.add('BID_OPEN_CLOSE', [pattern], greedy="LONGEST")
    matches = matcher(nlp_doc)
    # Print the matched spans

    bids = []
    for match_id, start, end in matches:
        bids.append(nlp_doc[start:end].text)

    patternDigit = [{'IS_DIGIT': True, 'LENGTH': 4}]
    matcher.add("BID_SECURITY_NUM", [patternDigit], greedy="LONGEST")
pend(int(bid[:4]))

    if (len(bids) > 0):
        return [max(timeBids), min(timeBids)]
    else:
        return None
```

Code Explanation:

1. **import spacy**: imports the **spacy** library for Natural Language Processing (NLP).
2. **from spacy.matcher import Matcher**: imports the **Matcher** class from the **spacy.matcher** module.
3. **import re**: imports the **re** module for regular expression matching.
4. **from nlplogic import getTokenText**: imports a function called **getTokenText** from a user-defined module **nlplogic**.
5. **nlp = spacy.load("en_core_web_sm")**: loads a pre-trained SpaCy model called "en_core_web_sm" into the variable **nlp**.
6. **def getBidTimes(file):**: defines a function called **getBidTimes** that takes a single parameter **file**.
7. **nlp_doc = getTokenText(file)**: calls the **getTokenText** function from the **nlplogic** module to extract text from the **file** parameter and create an **nlp** Doc object.
8. **matcher = Matcher(nlp.vocab)**: creates a **Matcher** object using the vocabulary of the **nlp** model.
9. **pattern = [{"LIKE_NUM": True, "LENGTH": 4, "OP": "+"}, {"LOWER": {"IN": ["hour", "hours"]}}]**: defines a pattern to match tokens that are numbers with a length of four followed by the word "hour" or "hours".
10. **matcher.add('BID_OPEN_CLOSE', [pattern], greedy="LONGEST")**: adds the pattern to the **matcher** object with the label 'BID_OPEN_CLOSE' and sets it to be greedy and match the longest possible sequence.
11. **matches = matcher(nlp_doc)**: applies the **matcher** object to the **nlp_doc** and returns a list of matched spans.
12. **bids = []**: creates an empty list to store matched bids.
13. **for match_id, start, end in matches:**: iterates over each matched span.
14. **bids.append(nlp_doc[start:end].text)**: extracts the text of each matched span and appends it to the **bids** list.

15. **patternDigit = [{'IS_DIGIT': True, 'LENGTH': 4}]**: defines a pattern to match tokens that are four-digit numb
16. **matcher.add("BID_SECURITY_NUM", [patternDigit], greedy="LONGEST")**: adds the pattern to the **matcher** object with the label 'BID_SECURITY_NUM' and sets it to be greedy and match the longest possible sequence.
17. **timeBids = []**: creates an empty list to store the time values of the matched bids.
18. **for bid in bids:**: iterates over each bid in the **bids** list.
19. **timeBids.append(int(bid[:4]))**: extracts the first four characters (i.e., the four-digit number) from each bid and converts it to an integer, which is then appended to the **timeBids** list.
20. **if (len(bids) > 0):**: checks if there are any matched bids.
21. **return [max(timeBids), min(timeBids)]**: returns a list containing the maximum and minimum time values from the **timeBids** list.
22. **else:**: if there are no matched bids,
23. **return None**: returns **None**

Code Summary:

The code uses the SpaCy library to extract bid times from a given file. The first step is tokenization, where the text from the file is broken down into individual units called tokens using the getTokenText() function. This step prepares the text for further analysis. Next, the code defines a pattern to match bid times using the SpaCy Matcher. The pattern consists of a four-digit number followed by the words "hour" or "hours." This pattern is designed to identify bid times expressed in hours.

Once the pattern is defined, the code applies the SpaCy Matcher to the tokenized text. The matcher searches for sequences of tokens that match the defined pattern. Whenever a match is found, the corresponding bid times are extracted.

The extracted bid times are initially in string format. To work with them numerically, the code converts them to integers. This conversion allows for easier comparison and calculation.

After converting the bid times to integers, the code determines the minimum and maximum bid times from the extracted values. It does this by finding the smallest and largest values among the bid times.

Finally, the code returns the minimum and maximum bid times as a list. This list provides the user with both the minimum and maximum bid times, allowing them to access this information conveniently.

In summary, the code utilizes SpaCy for tokenization and pattern matching to extract bid times expressed in hours from the given file. It then converts the extracted bid times to integers, finds the minimum and maximum values, and returns them as a list. This approach enables efficient extraction and analysis of bid time information.

3.8.5 Bid Security:

The Code:

```
import spacy
from spacy.matcher import Matcher
import re
from nlplogic import getTokenText

# Load the pre-trained SpaCy model
nlp = spacy.load("en_core_web_sm")
def getTwoPercent(file):
    nlp_doc = getTokenText(file)

    matcher = Matcher(nlp.vocab)
    # Define the pattern to match "hour" or "hours" with a four-digit
number before it
    pattern1 = [
        # {"LIKE_NUM": True, "LENGTH": 1, "OP": "+"},
        {"ORTH": "2"},
        {"IS_PUNCT": True, "OP": "?"},
    ]
    pattern2 = [
        # {"LIKE_NUM": True, "LENGTH": 1, "OP": "+"},
        {"LOWER": "two"},
        {"LOWER": "percent", "OP": "?"},
    ]
    matcher.add('TWO_PERCENT', [pattern1, pattern2], greedy="LONGEST")
    matches = matcher(nlp_doc)
    # Print the matched spans

    bids = []
    for match_id, start, end in matches:
        bids.append(nlp_doc[start:end].text)

    if (len(bids) > 0):
        return bids
    else:
        return None
```

Code Explanation:

1. **import spacy**: Importing the **spacy** library to perform natural language processing tasks.
2. **from spacy.matcher import Matcher**: Importing the **Matcher** class from **spacy.matcher** module to perform pattern matching on text.
3. **import re**: Importing the **re** module to perform regular expression matching.
4. **from nlplogic import getTokenText**: Importing **getTokenText** function from **nlplogic** module.
5. **nlp = spacy.load("en_core_web_sm")**: Loading the **en_core_web_sm** model of **spacy** library for English language process
6. **def getTwoPercent(file)**: Defining a function named **getTwoPercent** that takes a file as input.
7. **nlp_doc = getTokenText(file)**: Calling the **getTokenText** function to read the text from the file and preprocess it.
8. **matcher = Matcher(nlp.vocab)**: Initializing a new matcher object with the loaded **nlp** model vocabulary.
9. **pattern1 = [...]**: Defining a pattern to match "2" or "2." or "2," followed by a percentage sign.

10. **pattern2 = [...]**: Defining a pattern to match "two" or "two." or "two," followed by an optional "percent" keyword.

11. **matcher.add('TWO_PERCENT', [pattern1, pattern2], greedy="LONGEST")**: Adding the defined patterns to the matcher object with a label **TWO_PERCENT** and setting the matching strategy to "LONGEST".

12. **matches = matcher(nlp_doc)**: Matching the defined patterns in the **nlp_doc** text and getting the list of matches.

13. **bids = []**: Initializing an empty list **bids** to store the matched patterns.

14. **for match_id, start, end in matches:**: Looping through the matches and extracting the matched span's start and end positions.

15. **bids.append(nlp_doc[start:end].text)**: Extracting the matched text span from **nlp_doc** and appending it to the **bids** list.

16. **if (len(bids)** > **0):**: Checking if any pattern matched or not.

17. **return bids**: If any pattern matched, then returning the matched patterns.

18. **else:**: If no pattern matched, then returning None.

Code Summary:

The code defines a function called getTwoPercent that takes a file as input. The function's purpose is to extract any occurrences of the phrases "2%" or "two percent" from the text content of the file. Here's how the function works:

First, the function reads the text content of the file. This step involves accessing the file, reading its contents, and storing the text for further processing.

Next, the function utilizes the SpaCy library to search for occurrences of "2%" or "two percent" within the text. SpaCy provides a Matcher object that allows for defining patterns to identify specific phrases or patterns in the text.

The function uses SpaCy's Matcher to define patterns that match the strings "2%" or "two percent". These patterns are designed to capture the exact occurrences of these phrases within the text. Once the patterns are defined, the function applies the Matcher to the text. It searches for sequences of tokens that match the defined patterns. Whenever a match is found, the corresponding string is extracted.

The extracted strings, representing occurrences of "2%" or "two percent", are collected and stored in a list.

If any matches are found, the function returns the list of matched strings. This allows the user to access and utilize the extracted information.

However, if no matches are found, the function returns None. This indicates that there were no occurrences of "2%" or "two percent" in the text.

In summary, the getTwoPercent function reads the text content of a file, uses SpaCy's Matcher to define patterns for matching "2%" or "two percent" strings, searches for these patterns in the text, and returns a list of matched strings if any are found. If no matches are found, it returns None. This function enables the extraction of "2%" or "two percent" occurrences from the given file for further analysis or processing.

3.9 Use of Natural Language Processing (NLP):

3.9.1 Introduction to NLP

NLP stands for Natural Language Processing. NLP is a field of study and technology that concentrates on the interaction between computers and human language. It involves the creation of algorithms and models that enable computers to comprehend, interpret, and generate meaningful human language.

3.9.2 Importance of NLP in relation to the undertaking

In the context of the project, NLP is crucial for automating compliance verification of proposal documents in the e-bidding process of the construction industry. The framework seeks to streamline the compliance checking process by employing NLP techniques, making it more efficient, accurate, and error-free.

3.9.3 Difficulties Regarding the Compliance Verification of Bid Documents:

3.9.3.1 Bidding on Paper and Manual Compliance Verification

Traditionally, the construction industry has relied on paper-based tendering processes, which involve multiple parties and generate voluminous documentation. Checking the compliance of proposal documents manually is a laborious and time-consuming process.

3.9.3.2 Extensive and Error-Prone Procedure

Verifying conformance manually is susceptible to human error, such as missing or misinterpreting specific requirements. In addition, it requires a substantial amount of manpower and time, delaying the evaluation and decision-making process.

3.9.3.3 Necessity for Automation in Compliance Compliance Verification

There is a growing need to automate conformance verification to address these issues. The automation of the e-bidding process can reduce errors, save time, and increase its overall efficacy.

3.9.4 Framework Proposed for Automated Compliance Verification:

3.9.4.1 Framework Overview

The research proposes a framework for automated compliance verification of proposal documents with predefined standards, specifically the regulations of Pakistan's Public Procurement Regulatory Authority (PPRA). The framework intends to automate the compliance checking procedure using NLP algorithms.

3.9.4.2 Utilizing Natural Language Processing Algorithms for Compliance Verification

By facilitating the analysis and interpretation of bid documents written in natural language, NLP algorithms play a significant role in the framework. These algorithms are capable of extracting pertinent data, identifying key elements, and comparing them to predefined standards.

3.9.4.3 Advantages of Automated Compliance Checking

Using NLP to automate compliance verification has multiple advantages. It reduces reliance on manual labour, saves time, reduces errors, and ensures a more efficient and effective electronic tendering process. The framework enables stakeholders to make rapid, well-informed decisions, resulting in enhanced project outcomes and industry-wide performance.

3.9.5 NLP Techniques and Their Framework Application:

3.9.5.1 Recognising Named Entities (NER)

Named Entity Recognition is a Natural Language Processing (NLP) technique that identifies significant entities within text, such as project names, client names, contractual terms, and regulatory requirements. Compliance verification becomes more precise and efficient when these entities are accurately extracted from bid documents.

3.9.5.2 Parsing of Syntactic Structure

Syntactic parsing entails analysing the grammatical structure of proposal documents' sentences. By comprehending the relationship between words and their roles within a sentence, natural language processing algorithms can extract meaningful information and identify clauses, conditions, and obligations specified in the documents.

3.9.5.3 Lexical Analysis

NLP algorithms can comprehend the contextual meaning of proposal documents thanks to semantic analysis. It facilitates comprehension of the text's intent, purpose, and requirements. The framework can compare proposal document content to predefined compliance standards by analysing the semantics.

3.9.5.4 Emotional Analysis

It is possible to use sentiment analysis to gauge the overall sentiment conveyed in bid documents. This analysis can provide additional insight into compliance by identifying any favourable or unfavourable sentiment associated with particular clauses, requirements, and contractual terms.

3.9.6 The Role of Natural Language Processing in Efficient and Accurate Compliance Verification:

3.9.6.1 Extraction of Key Information from Bid Documents

NLP algorithms excel at identifying and extracting pertinent information from proposal documents. They are able to identify relevant sections, requirements, and clauses, extracting project-specific details that are essential for ensuring compliance.

3.9.6.2 Comparison to Established Standards

The NLP-based framework can evaluate the level of compliance by comparing extracted information from proposal documents with predefined compliance standards. It can identify any deviations, omissions, or inconsistencies, enabling stakeholders to make informed decisions based on accurate assessments of compliance.

Reducing Errors and Guaranteeing AccuracyError reduction is one of the significant benefits of using NLP for compliance verification. Manual compliance monitoring is susceptible to human error, whereas NLP algorithms offer a more systematic and consistent approach. By automating the process, the framework reduces the possibility of human error and increases the precision of compliance verification.

3.9.7 NLP-based Compliance Verification in Electronic Bidding:

3.9.7.1 Increased Productivity and Time Savings

By automating compliance verification with NLP, the e-bidding process is significantly streamlined. NLP algorithms are capable of rapidly analysing proposal documents, extracting pertinent information, and comparing it to predefined standards. This streamlined procedure saves time and enables stakeholders to evaluate proposals more effectively, thereby expediting decision-making and minimising project delays.

3.9.7.2 Cost Savings and Resource Efficiency

By automating compliance verification, organisations reduce their reliance on manual labour, resulting in cost savings. The framework reduces the need for extensive personnel, enabling construction firms to allocate their resources more efficiently. The time saved by automation can be redirected to other crucial duties, thereby increasing overall productivity and cost-efficiency.

3.9.7.3 Increased Precision and Risk Mitigation

NLP algorithms improve the accuracy and consistency of conformance verification. They can more precisely identify potential errors, inconsistencies, and deviations in bid documents than manual processes. By reducing human error, the framework reduces the likelihood of compliance-related issues and their potential repercussions, such as contractual disputes or legal complications.

3.9.8 Potential Effects of NLP Beyond Compliance Verification:

3.9.8.1 Innovations in Machine Learning for Compliance Monitoring

The successful application of NLP techniques to automate the compliance verification of bid documents paves the way for future advances in machine learning for compliance verification. It enables the development of more sophisticated NLP algorithms and models that are tailored to industry-specific requirements and regulatory frameworks.

3.9.8.2 NLP's Application in Other Domains

The application of NLP extends beyond the construction industry. The same framework and techniques can be adapted and applied to the verification of compliance in other domains, including the legal, financial, healthcare, and government sectors. NLP algorithms can assist in automating the analysis of legal contracts, financial documents, medical records, and regulatory compliance reports, thereby expediting processes and guaranteeing adherence to standards.

3.9.8.3 Future Paths and Opportunities

The utilization of NLP in conformance verification has bright future prospects. As NLP algorithms continue to develop and incorporate techniques such as deep learning and contextual comprehension, the precision and efficiency of compliance verification will increase. In addition, the combination of NLP with other emergent technologies, such as blockchain or smart contracts, could revolutionize the way compliance is ensured across multiple industries.

NLP is the underlying technology that enables the proposed framework for automating compliance verification of proposal documents in the e-bidding process of the construction industry. The framework streamlines the conformance checking process by employing NLP algorithms, thereby enhancing efficiency, precision, and risk mitigation. The advantages include time savings, cost reduction, improved accuracy, and the potential for compliance verification applications in additional domains. The successful integration of NLP in compliance verification not only enhances the performance of the construction industry, but also advances machine learning techniques in the field of compliance verification.

3.10 NLP Logic used:

3.10.1 The Code:

```
import spacy
from flask import Flask, request
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import PyPDF2
import numerizer
from spacy.matcher import Matcher
import re
import tabula
from numerizer import numerize

nlp = spacy.load("en_core_web_sm")
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))


def preprocess(text):
    # Tokenize the text data into words
    words = nlp(text)

    # Remove stop words from the text data
    tokens = nlp(text)
    tokens = [token.text.lower(
    ) for token in tokens if not token.is_stop and token.text.lower() not
in stop_words]

    # Perform lemmatization on the text data
    tokens = [lemmatizer.lemmatize(token) for token in tokens]

    # Join the cleaned words back into a single string

    clean_text = ' '.join(tokens)

    return clean_text


def genText(file):
    # with open('sample.pdf','rb') as pdf_file:
    pdf_reader = PyPDF2.PdfReader(file)
    text = ''
    for page in pdf_reader.pages:
        text += page.extract_text()

    clean_text = preprocess(text)

    return clean_text


def getTokenText(file):
    doc = nlp(genText(file))
    return doc
```

3.10.2 Code Explanation:

1. **import spacy**: imports the spacy module for natural language processing.
2. **from flask import Flask, request**: imports the Flask framework for building web applications.
3. **import nltk**: imports the Natural Language Toolkit for text processing.
4. **from nltk.corpus import stopwords**: imports the stopwords module from the NLTK corpus.
5. **from nltk.stem import WordNetLemmatizer**: imports the WordNetLemmatizer class from the NLTK stem module.
6. **import PyPDF2**: imports the PyPDF2 module for reading PDF files.
7. **import numerizer**: imports the numerizer module for converting numbers to digits.
8. **from spacy.matcher import Matcher**: imports the Matcher class from the spacy.matcher module.
9. **import re**: imports the regular expressions module for pattern matching.
10. **import tabula**: imports the tabula module for reading tables from PDF files.
11. **from numerizer import numerize**: imports the numerize function from the numerizer module.

12. **nlp = spacy.load("en_core_web_sm")**: loads the pre-trained SpaCy model for English language processing.

13. **lemmatizer = WordNetLemmatizer()**: creates an instance of the WordNetLemmatizer class for lemmatizing words.

14. **stop_words = set(stopwords.words('english'))**: creates a set of English language stop words.

15. **def preprocess(text):**: defines a function called **preprocess** that takes a text string as input.

16. **words = nlp(text)**: tokenizes the text data into words using the SpaCy model.

17. **tokens = nlp(text)**: tokenizes the text data into tokens using the SpaCy model.

18. **tokens = [token.text.lower() for token in tokens if not token.is_stop and token.text.lower() not in stop_words]**: removes stop words and converts the text to lowercase.

19. **tokens = [lemmatizer.lemmatize(token) for token in tokens]**: lemmatizes the tokens using the WordNetLemmatizer.

20. **clean_text = ' '.join(tokens)**: joins the cleaned tokens back into a single string.

21. **return clean_text**: returns the cleaned text.

22. **def genText(file):**: defines a function called **genText** that takes a PDF file as input.

23. **pdf_reader = PyPDF2.PdfReader(file)**: creates a PDF reader object for the input file.

24. **text = ''**: initializes an empty string for storing the extracted text.

25. **for page in pdf_reader.pages:**: loops through each page of the PDF file.

26. **text += page.extract_text()**: extracts the text from the current page and appends it to the **text** string.

27. **clean_text = preprocess(text)**: calls the **preprocess** function to clean the extracted text.

28. **return clean_text**: returns the cleaned text.
29. **def getTokenText(file):**: defines a function called **getTokenText** that takes a PDF file as input.
30. **doc = nlp(genText(file))**: generates a SpaCy document object for the cleaned text using the **genText** function.
31. **return doc**: returns the SpaCy document object.

Code Summary:

The code you described defines several functions for processing and extracting text from a PDF file. It imports various libraries including PyPDF2, spaCy, Flask, NLTK, and Tabula. Let's dive into the functions and their roles:

preprocess() function: This function is responsible for preprocessing the input text. It takes the input text as an argument and performs several text processing steps. It tokenizes the text, removing stop words (commonly occurring words with little semantic meaning), and performs lemmatization (reducing words to their base or dictionary form). These steps aim to clean and standardize the text for further analysis or extraction.

genText() function: This function extracts text from a PDF file. It takes a PDF file as input, utilizes the PyPDF2 library to read the file's content, and then preprocesses the extracted text using the preprocess() function. By applying preprocessing techniques, the function ensures that the resulting text is cleaned and ready for further processing. Finally, it returns the clean text obtained from the PDF file.

getTokenText() function: This function takes a PDF file as input. It leverages the genText() function to extract text from the PDF file and preprocesses the extracted text using the preprocess() function. The result is a processed "doc" object, which is often a spaCy representation of a document or a collection of text. The function returns this "doc" object, which can be further utilized for various natural language processing (NLP) tasks.

In summary, the code provides functions for processing and extracting text from a PDF file. It utilizes libraries like PyPDF2, spaCy, Flask, NLTK, and Tabula. The preprocess() function tokenizes, removes stop words, and performs lemmatization on the input text. The genText() function extracts text from a PDF file and preprocesses it using the preprocess() function, returning clean text. The getTokenText() function combines the functionality of genText() and preprocess() to extract text from a PDF file, preprocess it, and return a "doc" object. These functions enable efficient text extraction and preprocessing from PDF files for further analysis or application-specific tasks.

3.11 Methodology Summary:

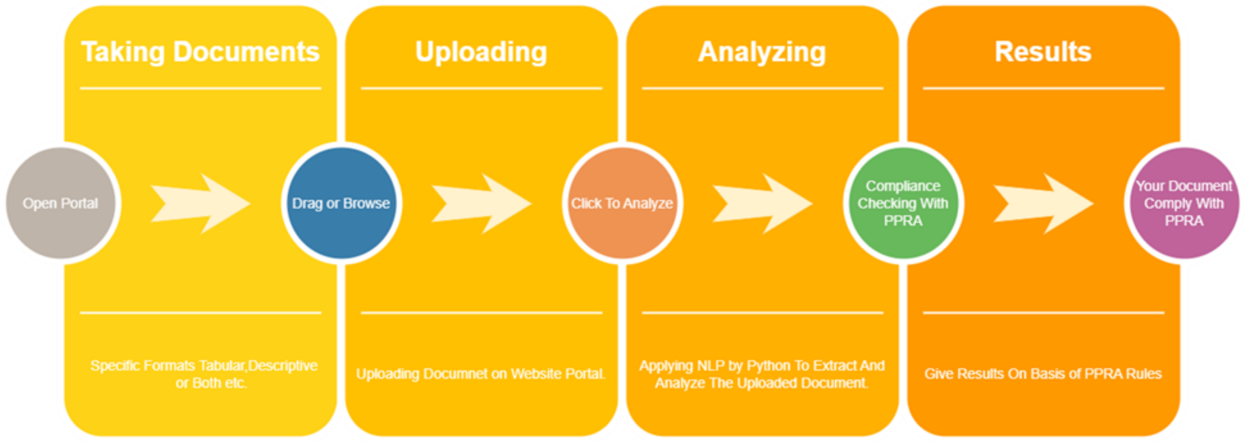
Essential to assuring transparency, fairness, and competition in the procurement process is the development of a software prototype for the compliance checking of bid documents with PPRA regulations. The methodology consists of identifying the pertinent rules, developing a compliance checklist, creating a software prototype, parsing the tendering documents, comparing with the checklist, identifying noncompliance, outputting results, and validating results. The prototype software should be user-friendly, efficient, and accurate at identifying PPRA compliance. This project concludes with a Python-based solution for efficiently and automatically determining PPRA compliance in construction proposal documents. Utilizing pre-trained modules and Spacy.matcher guarantees accurate and trustworthy results. The developed prototype can be expanded and refined to serve other industries where PPRA compliance is required. Five phases comprise the proposed methodology: user uploads and validation, data extraction, compliance check, and output generation. The comprehensive methodology ensures that all required data is collected and precisely analyzed, and that the compliance status is presented to the user in a transparent manner.

CHAPTER 4

4. RESULTS AND CONCLUSION:

4.1 Introduction to the Process Framework:

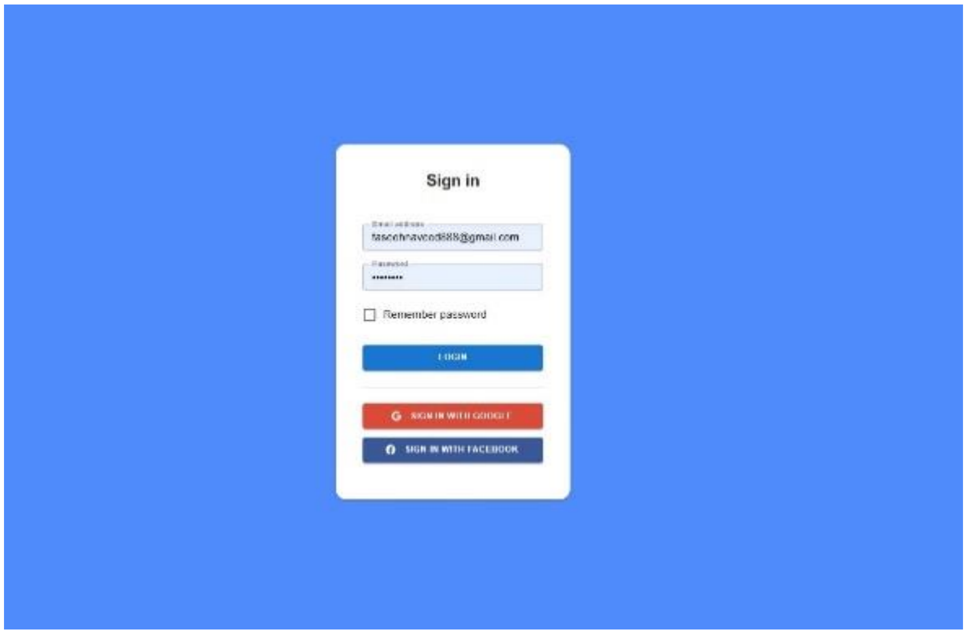
Process The process framework of the system that was designed to analyze uploaded documents to determine whether they adhere to or violate PPRA Rules is shown in Figure 1. The document that needs to be analysed is taken as the first step in the process structure. Uploading the papers to the website portal is the first step. After that, click to analyze. The system will then look at the documents using NLP Python to determine whether they conform with the PPRA Rules about response time, bid security, and integrity pact, among other things.


Taking Documents
Uploading
Analyzing
Results
Open Portal
Drag or Browse
Click To Analyze
Compliance Checking With PPRA
Your Document Comply With PPRA
Specific Formats Tabular,Descriptive or Both etc.
Uploading Documnet on Website Portal.
Applying NLP by Python To Extract And Analyze The Uploaded Document.
Give Results On Basis of PPRA Rules

The software uses Python's NLP platform to extract the needed data at the backend. And with the aid of Python libraries, modules, and other functions, it makes a final determination regarding whether the uploaded document complies with PPRA rules.

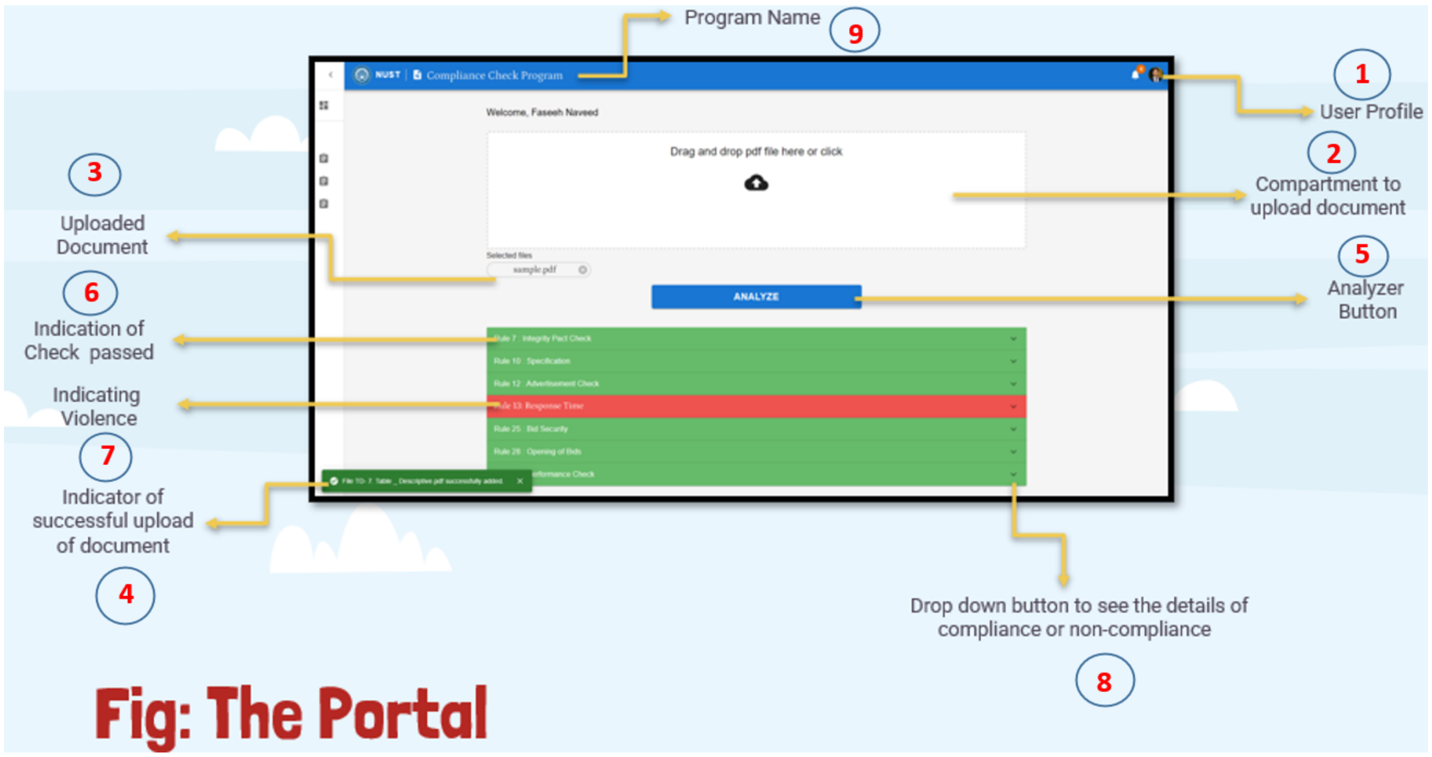
4.2 Introduction To Graphical User Interface (GUI):

Figure 3 depicts the Compliance Checking Programme (CCP) portal on the internet website, whose user-friendly layout makes it simple to analyze documents by dragging them into a drop box. To use the portal's services, a user must first log in using a confirmed email. As shown in the Figure.



faseehnaveed888@gmail.com

Sign in interface


Program Name
9
NUST
Compliance Check Program
1
User Profile
2
Compartment to upload document
5
Analyzer Button
3
Uploaded Document
6
Indication of Check passed
Indicating Violence
7
Indicator of successful upload of document
4
Drop down button to see the details of compliance or non-compliance
8
Fig: The Portal
Drag and drop pdf file here or click
Selected files
sample.pdf
ANALYZE

Welcome, Faseeh Naveed

Rule 7 : Integrity Pact Check
Rule 10 : Specification
Rule 12 : Advertisement Check
Rule 13: Response Time
Rule 25 : Bid Security
Rule 28 : Opening of Bids
Performance Check

File TD- 7. Table _ Descriptive pdf successfully added

Portal Interface

4.3 Portal Explanation Step by Step:

4.3.1 User Profile:

- The Compliance Check Program portal allows users to establish their own profiles, providing flexibility and convenience.
- They may log in using their existing Google or Facebook accounts, or they may establish a new account using their email address and password.
- This multichannel authentication allows users to access the portal via their favored method.

4.3.2 Document Upload Compartment:

- The interface of the portal includes a designated, conspicuously displayed section where users can easily upload their documents.
- This section allows users to readily locate and access the document upload functionality.
- By providing an intuitive section, the portal seeks to streamline the document submission process and improve the user experience.

4.3.3 Document Upload:

- Upon logging in to the portal, users can simply upload their documents in PDF format, thereby streamlining the submission process.
- To assure standardized document processing, the portal only supports the PDF format.
- The submitted documents must be in text format for efficient analysis and Extraction of pertinent information. The portal provides a clear view of the uploaded document, allowing users to verify its accuracy before proceeding with the compliance check.

4.3.4 Indication of Successful Document Upload:

- To provide users with instantaneous feedback, the portal displays a popup window upon successful document upload.
- This message assures users that their document has been uploaded securely and is available for analysis.
- By immediately acknowledging a successful upload, the portal inspires user confidence and promotes process transparency.

4.3.5 Analyze Button:

- Once the document has been uploaded, users can initiate the compliance check procedure by clicking the "Analyze" button.
- Behind the scenes, the system analyzes the uploaded PDF using sophisticated algorithms and NLP techniques.
- The system extracts pertinent information from the document, such as project details, deadlines, and requirements, for further evaluation.

4.3.6 Indication of Passed Check:

- When the compliance check is effectively passed, the portal displays green bars to indicate this.
- The portal also provides comprehensive information on how the analyzed document conforms to PPRA's rules and regulations.
- This indication ensures that the bidding process is transparent and compliant by assuring users that their document satisfies the compliance standards.

4.3.7 Violations detection:

- If any violations are discovered during the compliance check, the portal immediately highlights them with red bars.
- The presence of red bars alerts users to deviations from mandated PPRA rules and regulations.

- By identifying violations, the portal aids users in resolving noncompliance issues and ensures the tendering process is transparent

4.3.8 Dropdown Feature:

- To provide comprehensive information, the portal displays the compliance check results using a submenu feature.
- Users can access specific details regarding the compliance status, corresponding PPRA rule, and rule number via the navigation menu.
- This feature enables users to have a thorough comprehension of the compliance assessment and quickly access pertinent information for further actions or clarifications.

4.4 Website Name:

- For our website, Our **Compliance Checking Program** (CCP) has been named as **Compliance Companion.**

4.4.1 Summary:

By offering several login choices, convenient document upload features, clear indicators of check results, and thorough compliance information via the dropdown menu, the Compliance Check Programme portal promotes a good user experience. The portal aims to speed automatic compliance checks and support a compliant and fair bidding process by placing a strong emphasis on user usability and transparency.

4.5 CCP Portal Framework

4.5.1 Development of a secure and scalable CCP (Compliance Check Program) portal framework:

The assignment entails organizing and creating a CCP portal framework that complies with regulations and is expandable to support rising document and user volumes over time. Success requires careful consideration of compliance needs and scalable technologies.

4.5.2 Implementation of user-friendly interfaces and intuitive navigation for seamless user experience:

An iterative design method must be used to produce a user interface for a portal. This entails a user-centric strategy that optimises interactions and speeds navigation within the portal. To guarantee a consistent and engaging experience across various devices and screen sizes, responsive design principles should be used. Integrating user feedback and usability testing will help the interface get better over time. Additionally, to enable users navigate the portal efficiently, helpful tooltips, detailed instructions, and help documentation should be implemented. In general, developing user-focused interfaces that are both intuitive and visually appealing requires a continuous refinement and feedback integration process.

4.5.3 Integration of advanced technologies, including Python, NLP, and data processing algorithms:

Python is the programming language utilised to create the CCP portal's core. To extract, examine, and interpret textual data from uploaded documents, NLP techniques are used. In order to pre-process and transform the retrieved data for effective analysis, data processing algorithms are implemented. Additionally, machine learning techniques are used to improve the automation and accuracy of compliance inspections. Collaboration with domain experts and data scientists is essential to ensuring the best choice and application of pertinent technology. Overall, the CCP site can carry out compliance checks quickly and accurately because to the utilisation of Python, NLP, data processing, and machine learning technologies, as well as expert collaboration.

The CCP portal framework's integration of cutting-edge technologies like Python, Natural Language Processing, and data processing algorithms gives the compliance check programme complex capabilities. User-friendly interfaces and clear navigation improve the overall user experience while facilitating secure and scalable compliance checks. By utilising these cutting-edge technologies, the CCP interface streamlines, automates, and accurately verifies compliance, allowing stakeholders to navigate the complexity of compliance confidently and easily.

4.6 Sequence of Process:

Our CCP Portal Framework is developed following the sequence of processes explained in the coming following points:

4.6.1 Getting Document Uploaded by Third Party:

Ability for third parties to upload documents safely and securely. logon to a portal using a confirmed email. Including user authentication and access control controls to protect sensitive data. The document has a certain format, such as descriptive, tabular, or both.

4.6.2 Analyzing of Tender Notice:

Intelligent parsing and Extraction of critical information, such as project information, timeframes, and requirements, from tender announcements. NLP approaches used to understand and interpret tender notices' unstructured text. Classification and grouping of the retrieved data for subsequent analysis.

4.6.3 Comparison of Tender Notice against PPRA Rules:

Creation of a thorough rule-based mechanism to assess tender announcements against PPRA regulations and standards. Identification of compliance flaws, violations, or contradictions automatically in tender notices. Creation of thorough reports exposing noncompliance issues for additional examination and remedial action.

4.6.4 Checking Brand Name (Specifications):

The tender notice's brand names and specifications are examined for compliance with procurement laws. Verification of technical specifications and conformance to PPRA standards. Identifying any variations or biases in brand preferences that can obstruct fair competition

4.6.5 Checking Response Time:

Automated response time monitoring to guarantee compliance with tender notice requirements to due dates. Notifying necessary parties of probable delays or concerns with compliance with response requirements. Facilitating prompt and effective dialogue between buyers and bidder throughout the bidding process. And check that the mentioned response time is as per the PPRA Rules.

4.6.6 Bid Security 0r Performance Check:

Checking whether bid security or performance guarantees are present and legitimate as required by procurement regulations. Authentication of financial records and bid security-related paperwork. Making certain that bidders meet the financial requirements needed to take part in the bidding process. And check whether the performance and bid percentages comply with PPRA Rules.

4.6.7 Advertisement Checking:

Thorough examination of the advertisements' content, structure, and completeness in accordance with PPRA rules. Finding any errors, omissions, or false information in the advertisement. Ensuring accurate and compliant advertisements to ensure transparency, fairness, and equal opportunity for all prospective bidders. Easily accessible through websites and print media(newspaper).

4.6.8 Integrity Pact:

Verification of the presence of and adherence to integrity pacts, as necessary, to sustain morally upright and open behavior. Evaluation of each party's dedication to upholding integrity and avoiding unethical behavior during the bidding process. Finding any violations or breaches of the integrity contract and taking the required steps to address them.

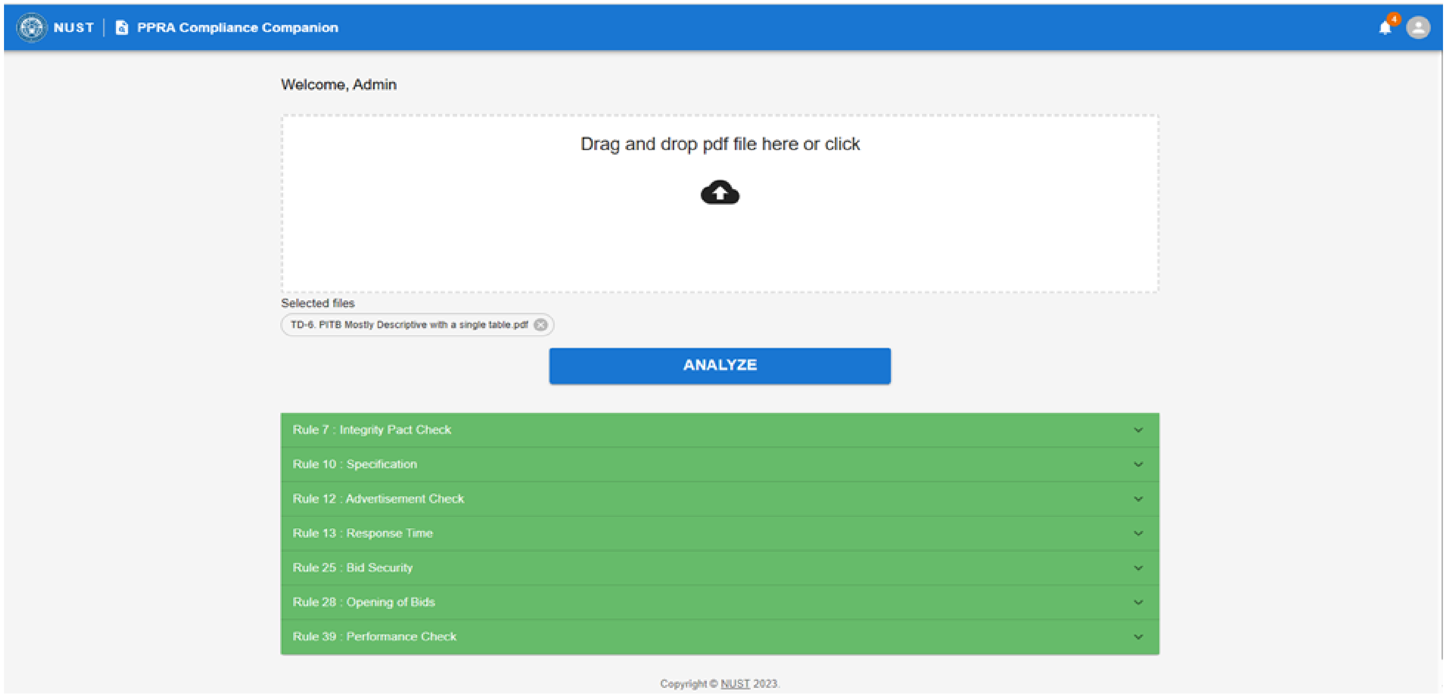
4.6.9 Bid Opening:

Verification of compliance with the PPRA regulations' specified bid opening procedures. Examining bid opening procedures, checking for the presence of authorized staff, and adhering to deadlines. To safeguard the integrity of the procurement system, transparency, fairness, and compliance must be ensured during the bid opening process. The CCP portal's integration of these features allows the compliance check program to improve the bidding process' efficiency, accuracy, and transparency. It enables stakeholders to make informed decisions, reduce riskguarantee a fair and competitive procurement environment by automating compliance checks and offering thorough reports.

4.7 Results:

The application of technology has revolutionized numerous industries, including public procurement. In this study, the researchers created a prototype that uses the Python programming language and various Python libraries, such as Beautiful Soup, Pandas, and NumPy, for data extraction and analysis to verify compliance with Pakistan's Public Procurement Regulatory Authority (PPRA) rules. This novel approach has the potential to considerably improve the efficiency and effectiveness of compliance checking, as it can quickly and accurately extract the necessary data from construction bid documents and check for PPRA rule compliance. The study revealed that only 30 of the 50 bid documents examined were entirely compliant with PPRA rules, while 10 documents contained minor violations and 10 contained major violations as shown in **figure 2 and figure 3** . This demonstrates the need for a more thorough and robust compliance surveillance system to ensure that all bidders comply with PPRA regulations. The researchers also identified the most common violations, which were related to the required documentation, such as bid bonds, technical specifications, and bills of quantities that were absent or incomplete.

In addition, the study revealed that some bidders did not satisfy PPRA eligibility requirements. This suggests that the eligibility screening procedure must be improved to ensure that only eligible bidders are permitted to participate in the bidding process.



Welcome, Admin

Drag and drop pdf file here or click

Selected files

TD-6. PITB Mostly Descriptive with a single table.pdf

Rule 7 : Integrity Pact Check

Rule 10 : Specification

Rule 12 : Advertisement Check

Rule 13 : Response Time

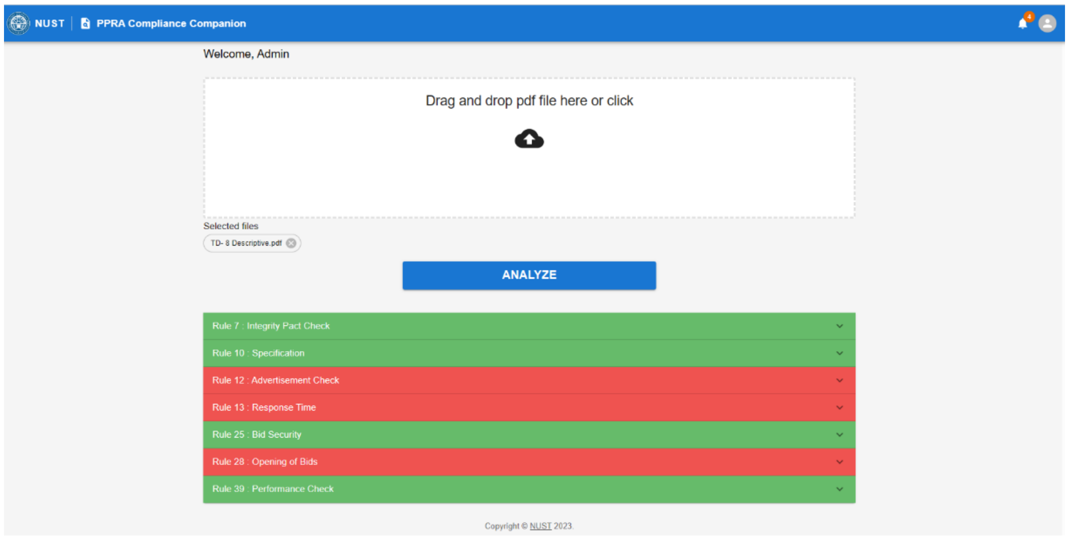
Rule 25 : Bid Security

Rule 28 : Opening of Bids

Rule 39 : Performance Check

Copyright © NUST 2023.

All Rules Compliant



Welcome, Admin

Drag and drop pdf file here or click

Selected files

TD- 8 Descriptive.pdf

Rule 7 : Integrity Pact Check

Rule 10 : Specification

Rule 12 : Advertisement Check

Rule 13 : Response Time

Rule 25 : Bid Security

Rule 28 : Opening of Bids

Rule 39 : Performance Check

Copyright © NUST 2023.

Some of the Rules are non-Compliant

The use of technology, such as the prototype created in this study, can significantly aid in streamlining the compliance checking procedure and ensuring that all bidders satisfy the necessary requirements prior to contract awarding.

The use of the Python programming language and various Python libraries for compliance checking of PPRA rules in Pakistan with construction tendering documents has the potential to significantly enhance the system's efficiency and effectiveness. This can help reduce the number of violations and ensure that all bidders comply with the PPRA's rules and regulations. To ensure the utmost level of accuracy and compliance, this method should be used in conjunction with traditional compliance monitoring techniques.

4.8 Testing and Evaluation:

4.8.1 Calculation for F1 Score:

In machine learning and natural language processing tasks, the F1 score is a commonly employed metric. It finds a balance between precision and recall by taking false positives and false negatives into account. The F1 score provides a comprehensive measure of the model's overall performance by supplying a single value. A higher F1 score, ranging from 0 to 1, indicates a superior balance between precision and recall. It is especially beneficial when both precision and recall are essential to a given task. False positives and false negatives are included in the F1 score, allowing for a more nuanced evaluation of the model's performance and highlighting areas for improvement.

A naive model that consistently predicts the majority class will have an F1 score of 0.0 for a balanced binary classification issue (provided there are any examples of the minority class in the test set). Therefore, you should strive for an F1 score significantly higher than 0.5 for a balanced classification problem. A score above 0.7 could be considered good, whereas a result above 0.9 is outstanding.

We used **50** different samples of advertisements in tabular, descriptive, and combined tabular-descriptive forms to train and fine-tune our CCP. After the algorithm has been trained, we use the **F1 Score** to evaluate it by inserting **15** unseen advertisements.

4.8.2 Calculations for Compliance Check Program:

Confusion Matrix	**Predicted Positive**	**Predicted Negative**
Actual Positive	TP = 7	FN = 2
Actual Negative	FP = 1	TN = 5

Total Data Set of Documents for Training = **50**

Total Data Set of Documents for Validation = **15**

The scale for F1 Score is from 0 to 1 with 0 being worst and 1 being best value.

Precision = **TP / (TP + FP)** = 7 / (7+1) = 0.875

Recall = **TP / (TP + FN)** = 7 / (7 +2) = 0.780

Finally, compute F1 Score: F1 = 2 * (Precision * Recall) / (Precision + Recall)

F1 Score = 2* (0.875 * 0.780) / (0.875 + 0.780) = 0.820

After Validation of approximately 15 unseen documents against our model, we obtain an F1 score of 0.82 or 82%, where the true positive (TP) is 7 and the false negative (FN) is 2, indicating that our system extracts the values which are actually present in the document, but our model does not extract the required value which is present in the document. The TP and FN fall within the pertinent category. Now, for the irrelevant category, our False Positive (FP) is 1, which indicates that our system assumes a value that is not present in the document, whereas our True Negavtive (TN) is 5, which indicates that our model does not select values that are not present in the documents. The precision is 0.875% and the recall is 78%.

4.9 Discussion:

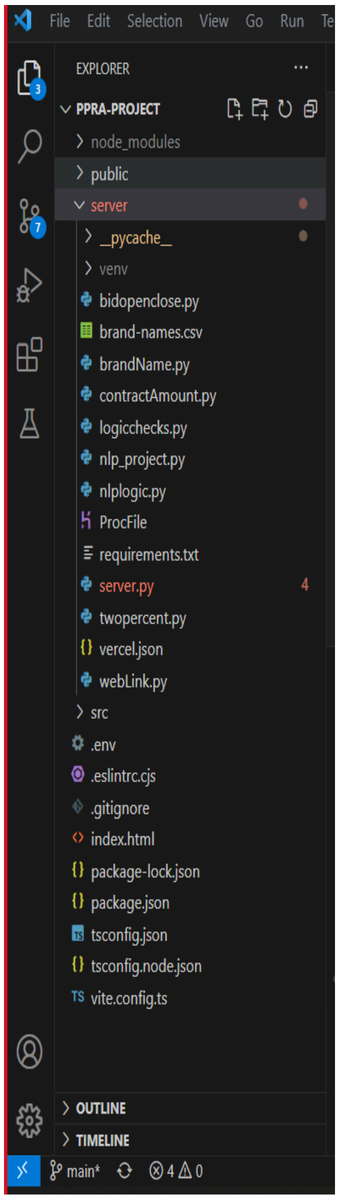
We use following library, modules and function related to Natural Language Process (NLP) in Framework of Compliance Checking Process (CCP).

4.9.1 Use of Natural Language Process (NLP):

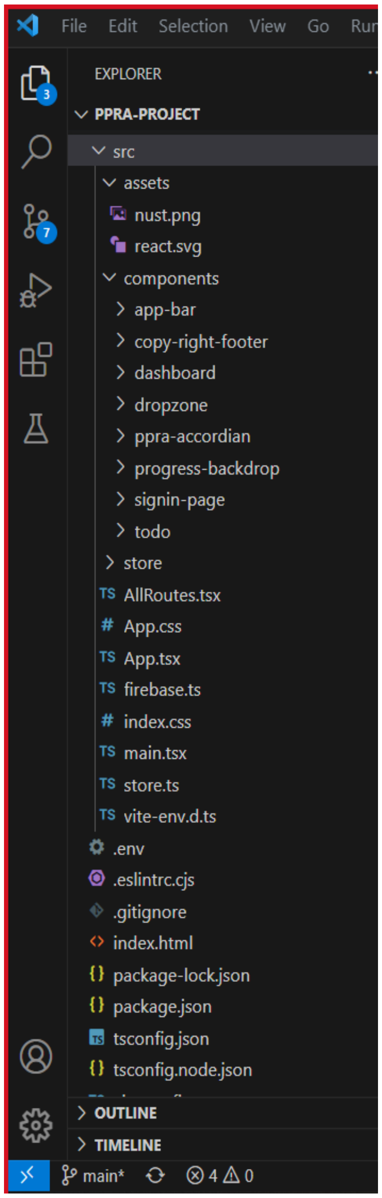
A branch of computer science and artificial intelligence called "natural language processing" (NLP) studies how language and computers interact. Python is a well-known programming language that has a large user community, strong libraries, and is frequently utilized in the NLP industry. The NLTK (Natural Language Toolkit), spaCy, TextBlob, and gensim are just a few of the Python packages that are specifically made for NLP. For tasks like tokenization, stemming, part-of-speech tagging, sentiment analysis, and named entity recognition, these libraries give developers the tools they need. Python can also be used for machine learning in NLP, enabling programmers to build models that can absorb knowledge from a vast quantity of textual input and generate predictions based on it. Developers can build programmes that comprehend and analyze human language, such as chatbots, sentiment analysis tools, and language translators, with the aid of Python and its NLP packages. Overall, Python and NLP have made it simpler for programmers to create intelligent apps that can comprehend and engage with human language, making it an important tool in today's technological environment.

4.9.2 Spacy (NLP Library):

Due to its effectiveness and speed, the Python Natural Language Processing (NLP) module spaCy is becoming more and more well-liked among programmers. It offers a more efficient method of carrying out typical NLP operations including dependency parsing, named entity recognition, part-of-speech tagging, tokenization, and more. SpaCy is the perfect tool for handling massive datasets because it was designed with a performance focus and is optimized for large-scale text processing. With more than 50 pre-trained models accessible for various languages, one of spaCy's distinctive advantages is its capacity to support many languages. For developers that need to work with multilingual data, this makes it a useful tool. Additionally, spaCy is made to be simple to use, with a straightforward API that enables developers to create unique NLP pipelines quickly. The ability of spaCy to recognize entities is another important aspect. It can recognize a variety of entities, including names, businesses, places, and more. Custom entities can also be taught to be recognized. This makes it a useful tool for software programmes like chatbots and search engines that must extract pertinent data from text. Overall, spaCy is a robust and effective NLP technology that works well with big datasets and multilingual text. It is a useful tool for developers trying to create intelligent programmes that can process and comprehend human language because of its simplicity of use and adaptable pipeline.



```
PPRA-PROJECT
  > node_modules
  > public
  v server
    > _pycache_
    > venv
    bidopenclose.py
    brand-names.csv
    brandName.py
    contractAmount.py
    logicchecks.py
    nlp_project.py
    nlplogic.py
    ProcFile
    requirements.txt
    server.py                4
    twopercent.py
    vercel.json
    webLink.py
  > src
  .env
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
  tsconfig.json
  tsconfig.node.json
  vite.config.ts
```



```
PPRA-PROJECT
  v src
    v assets
      nust.png
      react.svg
    v components
      > app-bar
      > copy-right-footer
      > dashboard
      > dropzone
      > ppra-accordian
      > progress-backdrop
      > signin-page
      > todo
    > store
    AllRoutes.tsx
    App.css
    App.tsx
    firebase.ts
    index.css
    main.tsx
    store.ts
    vite-env.d.ts
  .env
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
  tsconfig.json
  tsconfig.node.json
```

4.9.3 Use of Spacy Modules:

Popular Python Natural Language Processing (NLP) library spaCy includes a number of pre-built modules that make it simple to carry out a variety of NLP tasks. A pipeline made up of these modules can be used to carry out a variety of text processing operations.

4.9.3.1 Tokenizer:

The input text is divided up into tokens, such as words or punctuation marks, by the tokenizer module. White spaces, punctuation, and other special characters are used to indicate the borders of the tokens according to a set of rules.

4.9.3.2 Part-of-speech (POS) Tagger:

With the help of the POS tagger module, each token in a sentence can be classified as a noun, verb, adjective, or another type of token depending on its function. Numerous NLP tasks, including sentiment analysis, text categorization, and named entity identification, benefit from this knowledge.

4.9.3.3 Dependency Parser:

The relationships between the words in a sentence are determined by the dependency parser module. It can be used to create a tree structure that depicts the sentence's grammatical structure, with each word acting as a node and the connections between them acting as edges.

4.9.3.4 Named Entity Recognition (NER):

The NER module is used to locate entities in the input text, including names of people, places, businesses, and dates. Additionally, it can be trained to recognise unique items that are distinct to a given domain or application.

4.9.3.5 Text Classification:

The supplied text is categorised into one or more specified categories using the text classification module. Applications like sentiment analysis, spam detection, and topic modelling can all benefit from this.

Similarity: To determine how closely two or more texts are similar, use the similarity module. Applications like search engines, plagiarism detectors, and recommendation systems can all benefit from this.

Overall, developers may easily complete numerous NLP jobs without having to start from scratch thanks to spaCy's pre-built modules. These components can be put together in a pipeline to

4.9.4 Use of Spacy Functions:

SpaCy is a potent Python Natural Language Processing (NLP) module that offers several functions for handling and analysing text input. Several of spaCy's most popular features are listed below and shown in Figure 1 And Figure 2:

4.9.4.1 load():

To load a pre-trained model into memory, use the load() function. For a variety of NLP tasks, spaCy offers pretrained models for several languages that may be downloaded.

4.9.4.2 nlp():

To apply a number of NLP components to the input text, a processing pipeline is built using the nlp() function. Using this function, a tailored pipeline with certain NLP tasks can be made.

4.9.4.3 tokenize():

The input text is divided into tokens, such as words and punctuation, using the tokenize() method. The borders of the tokens are determined by this function using a set of rules.

4.9.4.4 pos_tag():

To determine the part of speech for each token in a sentence, use the pos_tag() method.Numerous NLP applications, including sentiment analysis, text categorization, and named entity identification, can make use of this data.

4.9.4.5 dependency_parse():

To determine the connections between the words in a sentence, use the dependency_parse() method. It can be used to create a tree structure that illustrates the sentence's grammatical structure.

4.9.4.6 ner():

The named entities in the input text, such as persons, locations, and organisations, are identified using the ner() function. It is also possible to teach this function to recognise unique entities.

4.9.4.7 similarity():

The similarity between two or more texts is calculated using the similarity() function. Applications like recommendation systems, plagiarism detection, and search engines can all benefit from this feature.

Overall, spaCy offers a large selection of features that can be utilised to carry out different NLP tasks. An NLP processing pipeline that is tailored to the needs of an application can be made by combining these features.

4.10 Conclusion:

The findings of the research demonstrate the enormous potential of the Python programming language for PPRA compliance checking of construction bid documents. The prototype developed by the researchers demonstrates the viability of using technology to expedite the compliance monitoring process, saving time and effort for authorities responsible for monitoring compliance and promoting bidding process transparency.

In addition, the study's analysis highlights the significance of awareness and training for applicants and procurement officials to ensure PPRA compliance. By providing guidance and training to participants in the procurement process, the PPRA can ensure that all parties understand and adhere to the rules, thereby enhancing the overall transparency and impartiality of the bidding process.

Overall, the study provides a useful instrument for monitoring compliance with PPRA regulations, which can aid in ensuring fair and transparent procurement processes in Pakistan's construction industry. Future research has the potential to further enhance the prototype's precision and efficiency. Exploring the use of machine learning algorithms for compliance checking could considerably improve the capabilities of the prototype. In addition, analyzing the impact of compliance monitoring on the quality and cost-effectiveness of construction projects could shed additional light on the advantages of using technology for compliance checking during the procurement process.

This study represents a significant step toward using technology to promote transparency, fairness, and conformance in Pakistan's construction industry's procurement process. There is enormous potential to further improve and enhance the efficiency and effectiveness of compliance monitoring to ensure equitable and transparent procurement processes in Pakistan and beyond because of continuing technological advancements.

CHAPTER 5

5. References:

1. Nannan Wang, Z.G., Zhuhuizi Xu, Zhankun Liu, Yu Han,, *A quantitative investigation of the technological innovation in large construction companies,*

Technology in Society. 2021.

2. Halaris, C., et al. *A system for virtual tendering and bidding*. in *Proc., Panhellinic Conf. in Informatics*. 2001.
3. Sayed, A.M., et al., *Drivers of e-bidding implementation in the Saudi Arabian construction industry.* Built Environment Project and Asset Management, 2019.
4. Wang, W.-C., *Electronic-Based Procedure for Managing Unbalanced Bids.* Journal of Construction Engineering and Management, 2004. **130**(3): p. 455-460.
5. Nesan Lenin, J., *Integrated E-bidding framework for construction.* International Journal of Construction Education and Research, 2011. **7**(4): p. 243-258.
6. Arslan, G., et al., *E-bidding proposal preparation system for construction projects.* Building and Environment, 2006. **41**(10): p. 1406-1413.
7. Mahfouz, T. and A. Kandil, *Litigation outcome prediction of differing site condition disputes through machine learning models.* Journal of Computing in Civil Engineering, 2012. **26**(3): p. 298-308.
8. Lee, J., J.-S. Yi, and J. Son, *Development of automatic-extraction model of poisonous clauses in international construction contracts using rule-based NLP.* Journal of Computing in Civil Engineering, 2019. **33**(3): p. 04019003.
9. Nizakat, R.Z., et al., *E-PROCUREMENT IN CONSTRUCTION INDUSTRY OF PAKISTAN: A CASE STUDY OF CONTRACTOR RELATED TO PUBLIC SECTOR PROJECTS.*
10. Rashidi, A., et al., *A Scientometric Analysis of Construction Bidding Research Activities.* Buildings, 2023. **13**(1): p. 220.
11. Thananusak, T., *Science Mapping of the Knowledge Base on Sustainable Entrepreneurship, 1996–2019.* Sustainability, 2019. **11**(13): p. 3565.
12. Zhang, J. and N.M. El-Gohary, *Automated information transformation for automated regulatory compliance checking in construction.* Journal of Computing in Civil Engineering, 2015. **29**(4): p. B4015001.
13. Fourie, D. and C. Malan, *Technological Approach to Ensure Ethical Procurement Management.* Factoring Ethics in Technology, Policy Making, Regulation and AI, 2021: p. 13.
14. Matthew, K., K. Patrick, and K. Denise, *The effects of fraudulent procurement practices on public procurement performance.* International Journal of Business and Behavioural Sciences, 2013. **3**(1): p. 17-27.
15. Aibinu, A.A. and A.M. Al-Lawati, *Using PLS-SEM technique to model construction organizations' willingness to participate in e-bidding.* Automation in construction, 2010. **19**(6): p. 714-724.
16. Caron, F., J. Vanthienen, and B. Baesens, *Comprehensive rule-based compliance checking and risk management with process mining.* Decision Support Systems, 2013. **54**(3): p. 1357-1369.
17. Zhang, J. and N.M. El-Gohary, *Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking.* Journal of Computing in Civil Engineering, 2016. **30**(2): p. 04015014.

18. Zhang, J. and N. El-Gohary, *Extraction of construction regulatory requirements from textual documents using natural language processing techniques*, in *Computing in Civil Engineering (2012)*. 2012. p. 453-460.
19. Tung, L.L. and O. Rieck, *Adoption of electronic government services among business organizations in Singapore.* The Journal of Strategic Information Systems, 2005. **14**(4): p. 417-440.
20. Al-Yahya, M. and K. Panuwatwanich, *Implementing e-tendering to improve the efficiency of public construction contract in Saudi Arabia.* International Journal of Procurement Management, 2018. **11**(3): p. 267-294.