



جامعة طيبة

كلية علوم وهندسة الحاسب الآلي

(قسم الطلاب)

KINGDOM OF SAUDI ARABIA

Ministry of Education

Taibah University

College of Computer Science and

Engineering

(Male Section)

**AI-Powered Requirements Analysis Modeling**

**Graduation Project 2**

**by**

Ali Talal Al-Ahmadi 4100379

Mohammed Hadi Al-Harbi 4101704

Rami Ramadan Al-Mohammadi 4101758

**A project submitted in partial fulfilment of the requirements for the degree of Bachelor of Science Computer Science**

**Supervised by**

Dr. Mohammad M. Alsuraihi

1st Semester - Academic Year 1445 (2023/2024)

# Abstract

The project may face issues with excessive resource consumption, such as time or costs, and frequent changes in requirements can lead to project delays and increased expenses. Additionally, sometimes the techniques or tools used in the analysis and design process are ineffective or outdated. Software maintenance constitutes a pivotal stage within the software development lifecycle, encompassing a substantial portion, varying from 40% to 80% of the total expenses associated with software development. It's worth highlighting that a notable 60% of the overall maintenance expenditure is dedicated to the improvement of existing software functionalities. Consequently, it becomes imperative to meticulously prepare appropriate software documentation at each developmental phase in order to alleviate the financial burdens of maintenance. Using artificial intelligence (AI) to analyze requirements and generate drawings or models can be an effective solution for saving time and costs in the design and analysis processes. The crux of reducing maintenance costs lies in enhancing one's grasp of the software system, as understanding a software system accounts for roughly 50% of the time spent in the maintenance phase.

**Keywords** NLP; ML; AI-Powered; Requirements; Analysis; UML

# Acknowledgement

Alhamdulillah. First and foremost, all praise is due to Allah SWT the worthy of all the praises and compliments for giving us the strength and ability to complete this project. Next, we express gratitude to our parents and families. We would like to extend our thanks to our project supervisor, Dr. Mohammad M. Alsuraihi, who guided us in executing this project, providing invaluable advice, assisting us in challenging times, and significantly contributing to the project's completion.

We also want to express our appreciation to all the professors at Taibah University, College of Engineering and Computer Science, who have been part of our academic journey at the university.

Finally, and not least, we would like to thank those who worked on this project: Ali Talal Al-Ahmadi, Mohammed Hadi Al-Harbi, and Rami Ramadan Al-Mahmoudi, for their contributions to this project. I appreciate their hard work and dedication.

**Contents**

[Abstract ii](#_Toc153058440)

[Acknowledgement iii](#_Toc153058441)

[List of Figures vi](#_Toc153058442)

[List of Tables vii](#_Toc153058443)

[List of Abbreviations viii](#_Toc153058444)

[1 Chapter 1: Introduction 1](#_Toc153058445)

[1.1 Introduction 1](#_Toc153058446)

[1.2 Problem Definition 1](#_Toc153058447)

[1.3 Project Aim and Objectives 2](#_Toc153058448)

[1.4 Project Methodology 2](#_Toc153058449)

[1.5 Project Timeline 4](#_Toc153058450)

[1.6 Document Organization 5](#_Toc153058451)

[1.7 Summary 6](#_Toc153058452)

[2 Chapter 2: Literature Review 7](#_Toc153058453)

[2.1 Introduction 7](#_Toc153058454)

[2.2 Research Methodology 7](#_Toc153058455)

[2.3 AI 9](#_Toc153058456)

[2.4 Machine Learning 11](#_Toc153058457)

[2.4.1 Neural Networks (NNS) 12](#_Toc153058458)

[2.4.2 Supervised Learning 12](#_Toc153058459)

[2.5 NLP 13](#_Toc153058460)

[2.5.1 Named Entity Recognition 14](#_Toc153058461)

[2.5.2 Parsing 16](#_Toc153058462)

[2.5.3 Pronoun Resolution 18](#_Toc153058463)

[2.5.4 Semantic Analysis of Requirement Using NLP and ML 19](#_Toc153058464)

[2.6 System Development for AI 20](#_Toc153058465)

[2.6.1 NLP application Development 20](#_Toc153058466)

[2.6.1 ML application Development 22](#_Toc153058467)

[2.7 Similar Tools 24](#_Toc153058468)

[2.7.1 The lucid platform 24](#_Toc153058469)

[2.7.2 Visual paradigm 25](#_Toc153058470)

[2.7.3 StarUML 26](#_Toc153058471)

[2.7.4 Enterprise Architect 27](#_Toc153058472)

[2.7.5 MagicDraw 28](#_Toc153058473)

[2.7.6 ChatUML 29](#_Toc153058474)

[2.8 Summary 29](#_Toc153058475)

[3 Chapter 3: System Analysis 30](#_Toc153058476)

[3.1 Introduction 30](#_Toc153058477)

[3.2 Methodology 30](#_Toc153058478)

[3.2.1 SDLC 30](#_Toc153058479)

[3.2.2 Analysis 31](#_Toc153058480)

[3.3 Analysis of Existing Systems 33](#_Toc153058481)

[3.4 Requirements Elicitation 34](#_Toc153058482)

[3.4.1 System Requirements 35](#_Toc153058483)

[3.4.2 Functional Requirements 35](#_Toc153058484)

[3.4.3 Non-Functional Requirements 36](#_Toc153058485)

[3.5 Requirements Specification 37](#_Toc153058486)

[3.6 Requirements Modeling 41](#_Toc153058487)

[3.6.1 Use case diagrams 41](#_Toc153058488)

[3.6.2 Class Diagram 43](#_Toc153058489)

[3.7 Summary 44](#_Toc153058490)

[4 Chapter 4: System Design 45](#_Toc153058491)

[4.1 Introduction 45](#_Toc153058492)

[4.2 Design Methodology 45](#_Toc153058493)

[4.3 Architectural Design 46](#_Toc153058494)

[4.4 Component Design 47](#_Toc153058495)

[4.5 Data Modeling Design 48](#_Toc153058496)

[4.6 User Interface Design 48](#_Toc153058497)

[4.7 Summary 50](#_Toc153058498)

[5 Chapter 5: Conclusion and Future Work 51](#_Toc153058499)

[5.1 Conclusion 51](#_Toc153058500)

[5.2 Goals Achieved 51](#_Toc153058501)

[5.3 Lessons Learnt 52](#_Toc153058502)

[5.4 Limitations and Future Work 52](#_Toc153058503)

[5.4.1 Limitations 52](#_Toc153058504)

[5.4.1 Future work 53](#_Toc153058505)

[6 References 54](#_Toc153058506)

# List of Figures

[Figure 1: Project Methodology 3](#_Toc153058507)

[Figure 2: Methodology Schema 9](#_Toc153058508)

[Figure 3: AI branch [7] 11](#_Toc153058509)

[Figure 4: Neural networks architecture [12] 12](#_Toc153058510)

[Figure 5: Supervised learning process [13] 13](#_Toc153058511)

[Figure 6: Word2Vec architecture: (a) CBOW; and (b) skip-gram [15] 14](#_Toc153058512)

[Figure 7: RNN model architecture [15] 15](#_Toc153058513)

[Figure 8: Examples of experimental results: (a) original text; and (b) NER results [15] 16](#_Toc153058514)

[Figure 9: Constituency and dependency structures for the sentence ‘the man hit the ball’ [16] 17](#_Toc153058515)

[Figure 10: An example of a non-projective parse tree [16] 17](#_Toc153058516)

[Figure 11: difference between "original sentence" and "sentence with resolved Coreference" [18] 18](#_Toc153058517)

[Figure 12: identify potential spans [18] 19](#_Toc153058518)

[Figure 13: group spans [18] 19](#_Toc153058519)

[Figure 14: replace pronouns with real-world entities [18] 19](#_Toc153058520)

[Figure 15: waterfall for project ((\*): It will be worked on in the second term). 31](#_Toc153058521)

[Figure 16: Description of stages of analysis 32](#_Toc153058522)

[Figure 17:Devolper use-case diagram 41](#_Toc153058523)

[Figure 18: user use-case diagram 42](#_Toc153058524)

[Figure 19: class diagram 43](#_Toc153058525)

[Figure 20:Design methodolgy ((\*): It will be worked on in the second term) 45](#_Toc153058526)

[Figure 21: Architectural Design 47](#_Toc153058527)

[Figure 22: Home Page 48](#_Toc153058528)

[Figure 23: Input Page 49](#_Toc153058529)

[Figure 24: Diagram Page 49](#_Toc153058530)

# List of Tables

[Table 1: Project plan v1 4](#_Toc153058531)

[Table 2: Project plan v2 5](#_Toc153058532)

[Table 3: Related Work Comparison 33](#_Toc153058533)

[Table 4:use-case (Upload Dataset) 37](#_Toc153058534)

[Table 5: use-case (Load Dataset) 37](#_Toc153058535)

[Table 6: use-case (train Model) 37](#_Toc153058536)

[Table 7:use-case (Analysis) 38](#_Toc153058537)

[Table 8:use-case (test Model) 38](#_Toc153058538)

[Table 9: use-case (Enter requirements Document) 39](#_Toc153058539)

[Table 10: use-case (analysis Document) 39](#_Toc153058540)

[Table 11: use-case (Create analysis model) 39](#_Toc153058541)

[Table 12: use-case (save diagram) 40](#_Toc153058542)

[Table 13: use-case (Draw use-case model) 40](#_Toc153058543)

[Table 14: use-case (Draw class model) 40](#_Toc153058544)

# List of Abbreviations

UML Unified Modeling Language

ML Machine Learning

NLP Natural Language Processing

AI Artificial Intelligence

UI User Interface

RNN Recurrent Neural Network

SDLC Software Development Life Cycle

NNS Neural Network Systems

NER Named Entity Recognition

# Chapter 1: Introduction

## Introduction

This project is a continuation of evaluated project 1. This continuation aims to develop and enhance the proposed solution based on the previous discussion and here some Notes:

* Focus on writing and know everything written in the project.
* Focus on clear, understandable, and simple project abstract writing.
* Research in scientific papers on the development of NLP applications.
* Use and benefit from the scikit-learn Library.
* Identifying the algorithms used in the project.
* There must be a system scenario that explains the system process.

## Project Aim and Objectives

The project’s goal is to deepen our understanding of the related work in terms of design and code, improve the system analysis process, complete the design phase with a focus on the data model and components, and program and test the system. All of this aims to speed up and facilitate the work of system modeling analysts and save their time.

To achieve this goal, we must achieve the following objectives:

1. Deepen the understanding of the related work in terms of design and code.
2. Improve the system analysis process.
3. Complete the design phase with a focus on the data model and components.
4. Program and test the system.
5. Conclude with the lessons learned and the knowledge and experience gained from working on this project.

## Project Methodology

## Project Timeline

This section presents the timeline plan versions of our project as the following:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **May** | | **April** | | | | **March** | | | | **February** | | | | **January** | **Months**  **(2024)** |
| **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and review discussion notes | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review existing systems |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Improve the system analysis process |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Complete design the system |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Program system and test |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Documentation |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | Presentation |

Table 1: Project plan v1

1. **Define project objectives and review discussion notes (1 week):**

We will define the main objectives of the project and review all the notes from previous discussions.

1. **Review** **existing systems (2 weeks):**

We will review the existing systems to know what languages were used in developing the previous systems and how they work.

1. **Improve the system analysis process (2 weeks):**

We will work on improving the system analysis process to ensure we get the most accurate results.

1. **Complete design the system (2 weeks):**

We will improve the system design and draw data models and components.

1. **Program system and test (6 weeks):**

We will program the system and test it to ensure it works as expected.

1. **Documentation:**

This task spans throughout the project's duration as it involves continuous documentation of the work.

1. **Project Presentation:**

The project should be presented during this time to showcase the results.

## Document Organization

* **Chapter 1.** In this chapter we discussed the plan and objectives and identified the problem that the project solves.
* **Chapter 2.** In this chapter, we reviewed existing systems.
* **Chapter 3.** Our focus in this chapter was on identifying functional and non-functional requirements using UML diagrams, as well as identifying user requirements and search methodologies.
* **Chapter 4.** In this chapter, we designed the system, identified the components of the system, modeled the data, and designed the user interfaces.
* **Chapter 5.** In this chapter, we programmed the system and conducted a series of tests to ensure that the system is working well.
* **Chapter 6.** A presentation of the project's conclusion was given, along with a proposal for new work to improve the current work, and a statement of whether the original objectives of the project had been met.

## Summary

In this chapter, we obtained a general idea of the challenges we face in analyzing project requirements to transform them into Use Case and Class Diagrams. We will build a model that relies on artificial intelligence to save time.

In the second chapter, we will delve into related work and similar solutions, discussing them in detail and compiling all relevant information.

# Chapter 2: Literature Review

## Introduction

In this chapter, we meticulously explore the current scholarly landscape within our field of study. Our primary objective is to conduct a precise examination of the existing literature and research pertaining to our subject matter. We will study and analyze the methodologies and techniques employed in this context.

Furthermore, we will undertake a comprehensive review and comparative analysis of systems and research akin to our project. Through these reviews and comparisons, we aim to extract valuable insights that contribute to understanding the historical and contemporary developments in our field, and elucidate the patterns, trends, and innovative approaches that have shaped the discourse and research.

By conducting a meticulous evaluation of prior works and engaging in a profound comparison with our research approach, we aim to direct our efforts towards delivering novel contributions and deepening our understanding of our study's domain.

## Research Methodology

Our data collection process involves using the Google Scholar search engine to find diverse scientific literature sources, such as research papers, conference papers, articles, and e-books that can assist us in understanding similar studies and identifying their strengths, weaknesses, similarities, and differences. For citation and referencing, we use digital libraries from IEEE, ACM, the university library, trusted websites (e.g., .org, .edu), and the Saudi Digital Library. Additionally, we use ChatGPT to assist us in translating and comprehending texts.

We will determine the Software Development Life Cycle (SDLC) model, whether we will use the Waterfall or Agile model, in Chapter 3 after researching topics related to our project and previous systems.

1. **IEEE** [3]**:** The IEEE style is a numerical style, where citations are numbered according to their appearance order. This number guides the reader to a complete reference in the reference list at the end of the work. The citation number should be placed within square brackets on the same line as the text, before any punctuation, with a space before the square brackets, such as: [x]. Once cited from a source, the same number is reused for all subsequent citations from the same source. [4]
2. **ACM Library Online** [5]**:** The ACM Digital Library is a research, discovery, and networking platform that offers full-text access to all ACM publications, including journals, conference proceedings, technical magazines, newsletters, and books. It also maintains a carefully curated collection of full-text publications from a limited number of publishers. The ACM Guide to Computer Literature is a searchable database dedicated solely to computing literature, encompassing a complex network of relationships among authors, works, institutions, and specialized communities.

After more than a week of researching topics related to our project, we have now compiled the main and subtopics as shown in the following figure.



Figure 2: Methodology Schema

## AI

Artificial intelligence (AI) is the branch of computer science concerned with making computers behave like humans (Computers with the ability to mimic or duplicate the functions of the human brain). [6]

Artificial intelligence systems encompass individuals, processes, hardware, software, data, and the essential knowledge required to develop computer and machine systems that exhibit intelligent characteristics. [6]

**AI conduct:**

* Learn from experience.
* Apply knowledge acquired from experience.
* Handle complex situations.
* Solve problems when important information is massing.
* Determine what is important.
* React quickly and correctly to a new situation.
* Process and manipulate symbols.

**Techniques used in AI:**

1. **Knowledge Representation**: It is the science of translating real-world knowledge into a form that can be used by computers.
2. **Search:** A technique for selecting the best solution from all possible solutions.
3. **Automated Reasoning:** The process of achieving a specific goal based on prior knowledge.
4. **Planning:** The ability to make a good sequence of actions to achieve our objectives.

**AI branch:**



Figure 3: AI branch [7]

## Machine Learning

Machine science is a branch of artificial intelligence that aims to facilitate human life by replicating human behaviors. It is used to train the machine to train itself and to teach the machine to deal with data accurately. One of its uses is to predict future results based on known or previous data. Some of its types include supervised learning, unsupervised learning, and neural networks. [8] [9]

### Neural Networks (NNS)

It is a group of neurons that bind to each other, and its characteristics are self-learning It consists of three layers the input layer, the hidden layer, and the output layer. The input layer receives variables to connect to the hidden layer which could be more than one layer containing neurons. Convolutional Neural Networks (CNNs) are useful and powerful in computer vision and learning and are also a type of neural network. [10] [11]



Figure 4: Neural networks architecture [12]

### Supervised Learning

The standard formula for supervised learning is that the learner learns from several categories through the input and output examples given to him. It is the most common technique for solving classification problems. One of its algorithms is Naive Bayes, which solves prediction problems, and Linear regression, which trains models in the trained dataset, and then uses the model to predict new data or test data. [13]

**Supervised Learning Process:**

In this process, each dataset stored or sent to the learning algorithm has a label or name identifying this data. Each label has more than one feature. for example, If we take the dog, it is a label that the features are (big ear, big nose, etc..). [13]

Trained data with features and labels are entered as input to the learning algorithm. In addition, correct outputs are entered, and the algorithm is learned by comparing correct outputs to actual outputs to find errors. [13]



Figure 5: Supervised learning process [13]

## NLP

Natural Language Processing (NLP) is a branch of artificial intelligence that involves processing and analyzing text data and includes machine learning to understand and interact with human language. [14]

**How does a computer understand languages?**

Computers are emotionless machines, so we must convert natural languages into numbers. This numerical transformation allows the computer to perform mathematical operations on language data, enabling it to comprehend human language. [14]

**Text Tokenization:**

It is a process in which unnecessary words are removed, and the root (or lemmatization) is applied. Each sentence is divided into a sequence of words for analyzing each word separately. For example, "I like study" would be broken down into "I," "like," "study." [15]

**Word Embedding:**

Term Frequency (TF) is the simplest of its kind and involves counting how often each word appears in a document. For example, two sentences, “The Contractor should prepare” and “The Engineer should submit,” would be mapped to [1the, 1Contractor, 1should, 1prepare, 0Engineer, 0submit] and [1the, 0Contractor, 1should, 0prepare, 1Engineer, 1submit]. [15]

Term Frequency-Inverse Document Frequency (TF-IDF) analyzes the importance of prevalent and widely used words, such as("a," "an," "the"(. For example, the two preceding sentences would be mapped to [0.5the, 1Contractor, 0.5should, 1prepare, 0Engineer, 0submit] and [0.5the, 0Contractor, 0.5should, 0prepare, 1Engineer, 1submit]. [15]

The latest word embedding technique is Word2Vec, which assigns words used in a similar context to be close to each other in the vector space. For example, it places "contractor" and "engineer" in a nearby vector space if they are often used together. There are two types of Word2Vec: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW tries to predict the current word from its surrounding words, controlling the number of surrounding words. Skip-Gram, on the other hand, predicts the surrounding words from the current word. According to developers, CBOW is faster, but Skip-Gram provides better word predictions. [15]



Figure 6: Word2Vec architecture: (a) CBOW; and (b) skip-gram [15]

### Named Entity Recognition

Construction specifications are a document that outlines the necessary requirements for performing work during a project's construction phase. They typically consist of a description and requirements. Because construction specifications require precision in understanding and adherence during project development, failing to comprehend the project can result in wasting time, human resources, and an increase in costs. [15]

NER, a subfield of machine learning-based information extraction methodologies, categorizes specific words such as names, locations, and objects. It is divided into two sections: structural and semantic information. Each word's category can be easily determined. For instance, names typically begin with a capital letter, and words associated with locations follow prepositions like "in," "on," or "to." Objects are usually nouns. Semantic information stands out for its strength and expandability, automatically identifying usage patterns for each word and obtaining evidence from these patterns through machine learning algorithms. [15]

**Recurrent Neural Network for NER:**

It is a form of deep neural network where networks are connected in a sequential structure. Therefore, it can handle sequential input data that may carry multiple meanings, even within the same context. However, there is a critical issue called the vanishing gradient problem. This means that as the length of the sequence increases, the gradient becomes smaller, significantly reducing the model's learning capacity. [15]

To address this problem, two gates were introduced: the forget gate (ft) and the input gate (it). The forget gate's function is to discard previous information, while the input gate's role is to retain essential information for longer periods by enhancing crucial signals and forgetting the non-essential ones. [15]



Figure 7: RNN model architecture [15]

**Prediction Result of the NER Model:**

The results consist of automatically tagging each word from the original text, and each word is individually assigned to its category. Despite some incorrect classifications, out of 85 results, 75 appeared to match the scientific identifier. [15]



Figure 8: Examples of experimental results: (a) original text; and (b) NER results [15]

### Parsing

Describing the problems of structural linguistic analysis of sentences through parse trees, the analysis can be divided into two main types: dependency parsing and constituency parsing. Dependency parse trees rely on direct relationships between sentence structures, while constituency parse trees rely on the analysis of formal grammatical rules. [16]



Figure 9: Constituency and dependency structures for the sentence ‘the man hit the ball’ [16]

Dependency parsing is based on separating dependency relationships and linking different symbols directly. Each symbol depends on the main symbol it is associated with, except for the root symbol of the tree. There are two types of dependency trees: projective and non-projective. Projective trees have no crossing arcs, as shown in the previous figure. Non-projective trees, on the other hand, have intersecting arcs, as shown in the following figure: [16]



Figure 10: An example of a non-projective parse tree [16]

In the development of a natural language text analyzer capable of working with multiple languages, we face a challenge due to the varying linguistic structures between languages. We propose using the Bidirectional Long Short Term Memory (BiLSTM) neural network model to enhance the analyzer's performance across different languages, as this model can handle complexities arising from long-range linguistic structures. [16]

### Pronoun Resolution

Natural Language Processing (NLP) is one of the most challenging branches of artificial intelligence primarily because natural language is full of exceptions and ambiguities that make it difficult for computers to learn. One approach to simplify this process is to eliminate imprecise expressions that require context for understanding. This makes it easier for computers to learn. Examples of such expressions include pronouns like "he" or "she," which can be replaced with specific names referring to them.

Coreference Resolution (CR) is the task of finding all linguistic expressions in a given text that refer to the same entity. We can address this problem by replacing pronouns with noun phrases after identifying these references and aggregating them. [17]

A screenshot of a phone

Description automatically generated

Figure 11: difference between "original sentence" and "sentence with resolved Coreference" [18]

Anaphora resolution (AR) is one of the cases that significantly differ from coreference resolution. An anaphora resolution occurs in the text when one term refers to another term and specifies the interpretation of another. [19]

Despite the distinctiveness of anaphora resolution from coreference resolution, coreference resolution is widely applicable and covers most cases. [17]

**Misleading pronominal references:**

Some cases can be misleading, such as cases where there is no relationship between the pronoun and other words in the text. Examples of these cases include derived sentences where the pronoun 'it' is redundant, and we can easily extract a sentence with the same meaning without using the pronoun 'it'. [20]

**Steps for Coreference Resolution:**



Figure 12: identify potential spans [18]

A white background with black text

Description automatically generated

Figure 13: group spans [18]



Figure 14: replace pronouns with real-world entities [18]

### Semantic Analysis of Requirement Using NLP and ML

During software development stages sometimes the quality of the analysis of the requirements is poor and can cause the failure of the software project. Measuring the quality of the analysis of the requirements is through, inter alia, understanding the requirements by making the terminology understood by different stakeholders and being clear is not ambiguous. [21]

**NLP activities:**

Activities that can be applied using NLP are the selection of requirements, elicitation of requirements, validation of requirements, creation of model, and discovery of ambiguity. [21]

**some of the categorizations used for NLP:**

Syntactic focuses on word analysis. Semantic, which focuses on the analysis of concepts and meaning of language programming using machine learning techniques. [21]

**Pre-processing approaches:**

Pre-processing is the process of preparing requirements by NLP techniques to convert and analyze requirements. [21] Includes 5 core techniques:

* **lexical analysis**: The first step in processing NL is performing a lexical analysis. The purpose of the lexical analysis is to interpret the meaning of individual words.
* **syntactic analysis**: The output of the lexical analysis serves as input to the syntactic analysis. The goal of the syntactic analysis is to uncover the grammatical structure of a sentence.
* **semantic analysis**: The purpose of the semantic analysis is to determine possible meanings of a sentence. Semantic analysis looks at the meanings of the different words in a sentence.
* **Categorization**
* **Pragmatic analysis**

## System Development for AI

In this section, we will discuss the system development for Artificial Intelligence applications and the stages followed in the development of Natural Language Processing (NLP) and Machine Learning (ML) applications, along with the languages used in development.

### NLP application Development

Developing Natural Language Processing (NLP) applications is crucial in the field of artificial intelligence. This work involves several sequential steps to ensure the creation of an effective application that leverages human language understanding. We will take a look at each step to comprehend it. [22]

1. **Data Acquisition:**

In this stage, we aim to acquire a dataset that reflects the linguistic diversity and complexity relevant to the application's subject. This step forms the foundation for understanding language and its varied usage.

1. **Text Cleaning:**

After obtaining the data, it's time to clean it from linguistic errors and distortions. This step is crucial to ensure accuracy and quality of results in later stages.

1. **Pre-processing:**

This part involves an initial analysis of the data and the application of basic natural language processing techniques. This step is fundamental to comprehend the language structure and prepare the data for training phases.

1. **Feature Engineering:**

This step focuses on utilizing features derived from the data in a way that contributes to the understanding of models and enhances their performance.

1. **Modeling:**

Modeling encompasses the training of models using processed data, refining them to achieve greater efficiency in understanding and processing language.

1. **Evaluation:**

After model training, their performance is evaluated using various tests to ensure their effectiveness and robustness in dealing with a wide range of scenarios.

1. **Deployment:**

Following model evaluation, the application is deployed for widespread use and application in real-world contexts.

1. **Monitoring and Model Updating:**

In this step, the application's performance is monitored post-deployment, and models can be updated based on changes in data or usage requirements.

**Using Python for Natural Language Processing (NLP):**

There are several options for tools and technologies when developing a natural language processing application, but we'll focus on Python code that utilizes the spaCy NLP library. [14]

There are many reasons to use Python and spaCy, including:

**Simplicity:** Python is an easy-to-learn language, making it suitable for newcomers to software development. It allows for writing straightforward code, especially compared to other languages.

**Community Support:** Python benefits from a large and active community. This means that solutions to problems can be found, educational resources are readily available, and assistance is just a question away.

**Libraries:** Python is renowned for its extensive libraries, particularly in the field of artificial intelligence. These libraries simplify the development of natural language processing applications (such: NLTK, spaCy, TextBlob, genism).

### ML application Development

Developing machine learning applications is an exciting challenge in the modern tech world. These steps represent the journey developers go through to build smart applications based on the learning capabilities of automated systems. we'll overview the main steps in developing these applications and how to achieve effective integration between the process components.

1. **Data Acquisition:**

Developers begin by examining and gathering data that the system will learn from. This data forms the foundation for understanding the application's context and training the model.

**Libraries in python:**

1. **NumPy:** A fundamental library for scientific computing and working with arrays.
2. **BeautifulSoup:** A library for extracting data from HTML and XML files.
3. **Data Analysis and Cleaning:**

Developing machine learning applications requires a thorough examination of the data, ensuring its cleanliness and freedom from errors to ensure quality training.

**Libraries in python:**

1. **Pandas:** A library for data analysis and cleaning.
2. **scikit-learn:** A machine learning library providing tools for data analysis and cleaning.
3. **Feature Selection:**

Identifying key features in the data that will contribute to the system's understanding of information and achieve outstanding performance.

**Libraries in python:**

1. **scikit-learn.**
2. **feature-engine**: A library for feature engineering in machine learning.
3. **Data Splitting:**

Dividing the data into sets for training and testing ensures an effective evaluation of the model and avoids overfitting problems.

**Libraries in python:**

1. **scikit-learn.**
2. **Model Selection:**

Modeling encompasses the training of models using processed data, refining them to achieve greater efficiency in understanding and processing language.

1. **Model Training:**

The model is trained using data, learning and benefiting from patterns and details within the data.

**Libraries in python:**

1. **scikit-learn.**
2. **ensorFlow:** A machine learning framework.
3. **Keras**: A high-level neural networks API.
4. **Performance Evaluation:**

The model's performance is assessed using test data to ensure its effectiveness and efficiency in dealing with new cases.

**Libraries in python:**

1. **scikit-learn.**
2. **Improvement and Adjustment:**

Based on the evaluation, developers adjust enhance the model's performance and efficiency in facing different challenges.

**Libraries in python:**

1. **scikit-learn.**
2. **TensorFlow:** A machine learning framework.
3. **Keras:** A high-level neural networks API.
4. **Application Deployment:**

After ensuring the model's quality, the application is deployed for public use and interaction with users.

1. **Performance Monitoring and Updating:**

Developers continue to monitor the application's performance after deployment, making necessary updates to keep up with changes and ensure continuous improvement.

## Similar Tools

### The lucid platform

The Lucid platform includes Lucidchart and Lucidspark products, working seamlessly to enhance productivity, clarity, and guidance at every stage of the workflow. [23]

The similarity in our program and the Lucid platform is that Lucid platform generates automatic maps from the data it possesses, while our program automatically draws UML diagrams from the user-inputted requirements.

It is a platform that helps teams build projects easily and collaboratively and has applications that help enhance productivity, clarity, and guidance at every stage of projects, including Lucidchart and Lucidspark. Lucid has released its own collaborative AI for its applications. [23]

Lucidspark is an application that helps teams with several activities, including brainstorming sessions and mentoring project teams. [23]

Lucidchart assists teams with several activities including building schemes such as UML, resource allocation for the project and the design of low-fidelity prototypes for user testing. [23]

**Collaborative AI:**

Collaborative AI enhances collaborative thinking processes, improves team innovation sessions, and supports design thinking. [23] Some uses of Collaborative AI include:

1. **Idea Generation:** Using artificial intelligence to generate new ideas from a set of keywords or a specific topic. [23]
2. **Idea Sorting:** Employing artificial intelligence to analyze a series of notes and categorize them. [23]
3. **Idea Summarization:** Using artificial intelligence to provide a summary of identified notes. [23]

### Visual paradigm

Visual Paradigm is a software application designed for software development teams, used for modeling business information systems, and managing development processes, but our project Draws it automatically and the user cannot modify the drawing.

One of its AI-powered tools:

1. **code generation:**

This functionality generates code from UML class diagrams, facilitating the transformation of design into executable code. Additionally, you can reverse this process, allowing the conversion of code into a UML model for model extraction or design modification. [25]

It supports various programming languages such as Java, C++, C#, Python, PHP, and others. [25]

1. **Intelligent pattern recognition:**

Design pattern is considered a part of the diagram that can be reused in various diagrams to illustrate the Intelligent pattern recognition tool. First, we need to know what is meant by defining design pattern.

**defining design pattern:**

It is a tool like Intelligent Pattern Recognition, but it does not use artificial intelligence. It allows you to define a pattern and reuse it in your project or share it with your team. To apply the pattern, you need to manually define and save it as a ready-to-use pattern file. [26]

**The difference between Intelligent Pattern Recognition and defining design pattern:**

Defining design patterns involves manually defining and saving the pattern, while Intelligent Pattern Recognition utilizes artificial intelligence to automatically analyze and comprehend patterns without human intervention.

1. **Real-time Diagram validation:**

This is a process where changes in the diagram are immediately examined as the user makes any modifications. It checks the validity of the changes and guides the user to any potential errors in real-time.

### StarUML

It is an open-source software modeling application primarily used to support the Unified Modeling Language (UML). [27]

StarUML programs offer users the ability to create, draw, and design models using the Unified Modeling Language (UML), and It focuses primarily on manual UML design, providing users with the flexibility to precisely control the layout and arrangement of elements in their diagrams but our project Draws it automatically and The user cannot modify the drawing.

**Some features of StarUML:**

1. **Design Speed:**

Enables users to create elements and link them quickly and efficiently. [27]

1. **Integration with Programming Languages:**

Supports programming languages such as Java, C++, C#. [27]

1. **Versatile Usage:**

Considered useful in modeling diverse applications and systems. [27]

**Among the features that leverage artificial intelligence:**

1. **Creating Use Cases and Class Diagrams:**

Artificial intelligence is employed to guide users during the creation of use cases and class diagrams, providing guidance on relevant elements related to use cases and class diagrams.

1. **Suggesting Relevant Elements:**

Artificial intelligence is utilized to suggest relevant elements based on context.

1. **Detecting Errors and Inconsistencies:**

Artificial intelligence is employed to detect potential errors in the model, contributing to the enhancement of analysis quality by identifying and correcting any contradictions or errors in the design.

### Enterprise Architect

It is an application for software modeling used to design and build software systems, model business processes, depict systems, and current operations. [28]

Both our project and Enterprise Architect provide the capability to analyze requirements, in our project allows users to input requirements through writing or file analysis, this is a common feature with Enterprise Architect, which also supports inputting information in various forms, in our project it can Automated generation of drawings UML but in Enterprise But this feature is not available.

**Features include:**

1. Model Management
2. Project Management
3. Model Version Control
4. Generate RTF and HTML Documents (Reports)

**Among the features that leverage artificial intelligence:**

1. **Predictive Modeling:**

Utilizes artificial intelligence to analyze data and forecast future trends, aiding in guiding strategic decisions by understanding the potential future of the project.

1. **Risk Assessment:**

Uses artificial intelligence to analyze information and estimate potential risks, assisting in identifying and mitigating potential risks before they impact the project.

1. **Automated Testing:**

Uses artificial intelligence to execute tests automatically, ensuring quality and continuity.

### MagicDraw

MagicDraw facilitates the execution of the software development life cycle, unlike other modeling and architectural environments that rely on UML, [29] In our system, we'll use Conflict Analysis, similar to MagicDraw, to analyze requirements and create a model with minimal contradictions.

It allows for a focus on business and process modeling, gathering requirements, and design, providing independence from any specific software development process. [29]

**Among the features that leverage artificial intelligence:**

1. **Smart Requirements Management:**

Artificial intelligence is used to automatically link requirements with UML diagrams, ensuring an accurate and updated representation of project requirements.

1. **Conflict Analysis:**

MagicDraw can analyze models and identify conflicts between requirements and different models.

1. **Process Coordination Improvement:**

By using artificial intelligence-supported features, teams can enhance the coordination of development processes, contribute to improving model quality, and ensure compliance with requirements.

### ChatUML

chatUML is a chat system between the user and artificial intelligence that can take requirements and convert them into UML diagrams. You can also modify the diagrams by requesting changes through the chat, making it similar to our system to a great extent.

Some of its features include:

1. Easily generating UML diagrams.
2. Conveniently editing diagrams.

## Summary

In this chapter thoroughly examines the research context, starting with a detailed review of existing literature. The emphasis is on a meticulous analysis of Very precise. The research methodology, which includes Google Scholar, digital libraries for data collection, is outlined succinctly.

The chapter then delves into the domains of Artificial Intelligence (AI), covering its definition, conduct, and techniques such as knowledge representation, search, automated reasoning, and planning. Machine Learning (ML) is introduced as a branch of AI, focusing on supervised learning, neural networks, and their applications like NLP.

The discussion on NLP includes text tokenization, word embedding, and techniques like TF, TF-IDF, and Word2Vec. Named Entity Recognition (NER), parsing, pronoun resolution, and semantic analysis are explored in the context of NLP.

The chapter transitions to system development for AI, outlining the stages for developing NLP and ML applications. Python and various libraries are highlighted for development. Similar tools in the field, such as Lucidchart, Visual Paradigm, StarUML, Enterprise Architect, and MagicDraw, are introduced, emphasizing their AI-powered features.

After investigating these topics, we will be prepared in Chapter 3 to analyze the system and find both functional and non-functional requirements.

# Chapter 3: System Analysis

## Introduction

In this chapter, we will focus on defining the system requirements using simple diagrams. We will use UML tools to explain how the system operates and its requirements. We will specify both functional and non-functional requirements and discuss the development methodology. We will also analyze an existing system.

## Methodology

### SDLC

In our project, we used the waterfall methodology for project execution. In the first step, the project plan. In the second step Literature Review. in the third step, analyzed the system. in the fourth step, system design.

Stability of Requirements: The Waterfall model requires defining project requirements early. It fits well when requirements are stable and undergo minimal changes.

Path Determination: The Waterfall model follows a linear and organized approach, allowing the team to clearly outline the workflow and task sequence. This structure is beneficial for projects requiring precision and strict timelines.



Figure 15: waterfall for project ((\*): It will be worked on in the second term).

### Analysis

In the analysis phase we have identified 4 basic activities to fully analyze the components of the system. Before starting with system design in general we must collect the requirements and analyze them correctly. During the analysis phase of the system, we used this existing strategy in Figure 16.



Figure 16: Description of stages of analysis

In the process of systems analysis existing, we will compare work related to our system and identify points of difference between them and mention the advantages of each work like our system.

In the process of requirements elicitation, we will describe system requirements and functional and non-functional requirements in the natural language, and we will Elicit them through brainstorming, use similar systems requirements, use the Lucid platform tool to generate ideas for the requirements.

In the process of requirements specification, we will construct tables to describe both functional and non-functional requirements.

In the process of requirements modeling, we will draw use cases that show all possible interactions with the system and write the description for them.

## Analysis of Existing Systems

At this stage, we compared our system with similar systems through features. Then we used some of the features of similar systems at the stage of the elicitation requirements.

The checkmark  represents that the advantage will be achieved in the system whereas the  shows that it will not be achieved in the system. “**?**” it shows that it is possible to try to make a feature in the system. The differences between our system and other comparable work are shown in Table 3.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| System  Feature | Lucid platform | Visual paradigm | StarUML | Enterprise Architect | MagicDraw | ChatUML | Our project |
| Create UML diagrams |  |  |  |  |  |  |  |
| Code Generation |  |  |  |  |  |  | ? |
| Smart Requirements Management |  |  |  |  |  |  |  |
| Automated generation of drawings. |  |  |  |  |  |  |  |
| Intelligent pattern recognition |  |  |  |  |  |  |  |
| Real-time Diagram validation |  |  |  |  |  |  |  |
| Suggesting Relevant Elements |  |  |  |  |  |  |  |
| Detecting Errors |  |  |  |  |  |  |  |
| Predictive Modeling |  |  |  |  |  |  |  |
| Conflict Analysis |  |  |  |  |  |  |  |
| Idea Generation |  |  |  |  |  |  |  |
| Idea Sorting |  |  |  |  |  |  |  |
| Idea Summarization |  |  |  |  |  |  |  |

Table 3: Related Work Comparison

We have identified some features that will not be achieved in the system for many reasons for each feature:

* Intelligent pattern recognition: because we have identified that the system analyzes the text or document through the context of the text through Named Entity Recognition. It classifies specific words such as names, locations and objects. It is divided into two sections: structural and semantic information.
* Real-time Diagram validation: because our system allows the user to enter text by typing and by upload file and at upload file, we do not need to process changes in real-time.
* Detecting Errors: because our system allows a certain percentage of requirements errors to be ignored by the user.
* Predictive Modeling: because our system does not predict future data and trends.
* Conflict Analysis: because our system does not identify inconsistencies between requirements and models but only generates them and displays them to the user.
* Idea Generation and Idea Sorting and Idea Summarization: because the system does not have the functions of generating ideas and sorting the ideas generated and not summarizing them.

## Requirements Elicitation

Requirements engineering is the process of discovering all system services, restrictions imposed on it, and services that meet user needs, analyzing, documenting, and verifying these services [31]. It includes high-level activities, including elicitation, analysis, and other activities. Requirements elicitation is the collection of all requirements, including system requirements and user requirements [31]. System requirements describe all the requirements that we want the system to perform [31]. After the elicitation process, we extracted the system requirements, and from them we identified the functional and non-functional requirements through brainstorming, using similar requirements tools, and using the Lucid platform tool.

### System Requirements

A structured document setting out detailed descriptions of the system’s services and operational constraints [33].

Defines what should be implemented so may be part of a contract between client and contractor [33].

* 1. To build an AI website system to receive software development requirements texts and documents and use NLP techniques to analyse them.
  2. To extract classes, functions (methods / procedure) and relations between classes and function.
  3. To generate UML diagrams: use-cases and class diagram.

### Functional Requirements

The requirements determine the system's behavior, what it should and shouldn't provide, and how it interacts with inputs [31].

1. To build an AI website system to receive software development requirements texts and documents and use NLP techniques to analyses them.
   1. The system shall allow the user to enter requirements either by typing directly or by uploading a document.
   2. The system shall allow the user to create analysis model.
   3. The system shall allow the user to choose type of UML diagram such as (class, use case).
   4. The system shall prepare to clean and organize the initial data to perform the text analysis process.
   5. The system should analyze the text by doing Tokenization, Parsing and Relation Extraction.
   6. The system should allow the user to export the diagram by png format.
2. To extract classes, functions (methods / procedure) and relations between classes and function.
   1. The system shall analyses the requirements context using NLP techniques.
3. To generate UML diagrams: use-cases and class diagram.
   1. The system shall draw the diagram after generated by NLP techniques.

### Non-Functional Requirements

It describes the characteristics of the system and the constraints imposed on the services provided by the system such as time constraints and other constraints. It also describes specific standards for some of the system's services [31].

1. To build an AI website system to receive software development requirements texts and documents and use NLP techniques to analyses them.
2. The system must be fast, ensuring that the processing of requirements and drawing (use case or class diagram) does not exceed (write the time).
3. The resulting diagrams in the system should have an accuracy of no less than (write the percentage).
4. It should be able to handle (write the range of users) simultaneously.
5. The system should be user-friendly and adaptable to various devices to provide a seamless experience.
6. The system should perform its functions with minimal steps.
7. To extract classes, functions (methods / procedure) and relations between classes and function.
8. The system should be able to recognize and analyze a minimum of (write the percentage) of user requirements entered, with a permissible margin for error not exceeding (write the percentage).

## Requirements Specification

|  |  |
| --- | --- |
| Upload Dataset | Use case |
| Developer, DBMS | Actor |
| 1. the user uploading a dataset into the system. 2. After the user uploads successfully the data uploads to Training Dataset and Testing Dataset.  * Training Dataset: The training dataset is used to train the machine learning model. * Testing Dataset: The testing dataset is a separate set of examples that the model has not seen during training. | Description |
| The system should be in a state ready to receive the dataset, with sufficient storage and processing resources available. | Pre-condition |
| The dataset is successfully uploaded and stored in the system. | Postcondition |

Table 4:use-case (Upload Dataset)

|  |  |
| --- | --- |
| load Dataset | Use case |
| DBMS | Actor |
| 1. After the Training Dataset and Testing Dataset 2. The data load to use to train Model and test model and Prepare data | Description |
| The data loading system is set up and accessible to users. | Pre-condition |
| After load Dataset can use the loaded dataset to generate train Model and test model and Prepare data | Postcondition |

Table 5: use-case (Load Dataset)

|  |  |
| --- | --- |
| train Model | Use case |
| Developer | Actor |
| 1. After load data from training dataset 2. The train model Training involves the model learning patterns and relationships within the data, which is a form of analysis. 3. After the training, the developer analyzes how well the model is learning from the data. 4. The developer assesses the performance of the trained model using a separate testing dataset. | Description |
| A labeled dataset is available for training. | Pre-condition |
| The machine learning model has completed the training process successfully. | Postcondition |

Table 6: use-case (train Model)

|  |  |
| --- | --- |
| Analysis | Use case |
| Developer | Actor |
| 1. After training model 2. Prepare data from load dataset perform initial cleaning steps.  * Tokenization: tokenization is the process of breaking the text into individual units for analysis. * Clean Data: This step is crucial to ensure accuracy and quality of results in later stages.  1. Extraction NERs; typically involves identifying and classifying entities, such as names of people. 2. Relation Extraction: module identifies and classifies relationships between entities in unstructured text data. 3. Extraction Event: A module extracts events from text, offering insights into specific occurrences. 4. After analysis is completed generate Diagram Specs get all the Analysis specifications like number of the classes and the relation between them and then draw | Description |
| A dataset has been successfully loaded into the data processing system | Pre-condition |
| the analysis is completed and ready to draw | Postcondition |

Table 7:use-case (Analysis)

|  |  |
| --- | --- |
| test Model | Use case |
| Developer | Actor |
| 1. The Developer loads data from testing dataset Use a separate dataset not used during training Model. 2. After loading data start representing real-world scenarios and cover the kind of cases the model may encounter. 3. The developer applies the loaded model to the testing dataset to make predictions. 4. The developer analyzes the model's predictions and evaluates its performance using metrics such as accuracy, precision, recall, and F1 score. | Description |
| A testing dataset, distinct from the training data, is available. | Pre-condition |
| The model's performance is evaluated based on the testing dataset. | Postcondition |

Table 8:use-case (test Model)

|  |  |
| --- | --- |
| Use case | Enter requirements Document |
| Actor | User |
| Description | 1. The user clicks on the text box. 2. The user enters requirements by typing.   Or The user clicks on the select doc. |
| Pre-condition | The user enters the website. |
| Post-condition | The requirements are entered successfully |

Table 9: use-case (Enter requirements Document)

|  |  |
| --- | --- |
| Use case | Analysis |
| Actor | User |
| Description | 1. After the user entered the requirements. 2. Preparation: Analysts clean and structure raw data for analysis  * Tokenization: Resumes are tokenized into individual words or phrases * Clean Data: This step is crucial to ensure accuracy and quality of results in later stages.  1. Extraction NERs; typically involves identifying and classifying entities, such as names of people. 2. Relation Extraction: module identifies and classifies relationships between entities in unstructured text data. 3. information Extraction: system gathers structured information from various sources, preparing it in a usable format for analysis. 4. Event Extraction: A module extracts events from text, offering insights into specific occurrences. 5. After analysis is completed generate Diagram Specs get the analysis like number the class and the relation between them and then draw |
| Pre-condition | Data is available for analysis. |
| Post-condition | the analysis is completed |

Table 10: use-case (analysis Document)

|  |  |
| --- | --- |
| Use case | Draw analysis model |
| Actor | User |
| Description | 1. The user clicks on select type. 2. The system shows to type of model (class model, use-case model)   If the user chooses class model   1. The system shows class model on screen after the user start generate.   If the user chooses class model   1. The system shows use-case model on screen after the user start generate |
| Pre-condition | The system will determine whether it is the context or Document to Draw the Diagram. |
| Post-condition | The selected model type is displayed on the screen. |

Table 11: use-case (Create analysis model)

|  |  |
| --- | --- |
| Use case | save diagram |
| Actor | User, DBMS |
| Description | 1. After the system draw the model (class, use-case). 2. The user click save diagram. 3. The system prompts the user to choose a location and provide a name for the saved diagram. 4. The user selects the destination folder and enters a name for the diagram file. 5. The system determines the type of model (e.g., class diagram, use case diagram) based on the previously chosen model type. 6. The system saves the diagram data to the designated location using the DBMS for storage. 7. The system displays a confirmation message indicating that the diagram has been successfully saved. |
| Pre-condition | The selected model type is displayed on the screen. |
| Post-condition | The model is successfully saved to the specified location, and the user receives a confirmation message. |

Table 12: use-case (save diagram)

|  |  |
| --- | --- |
| Use case | use-case diagram |
| Actor | User |
| Description | 1. After analysis completed the system draws a use-case model after the requirements. 2. The system displays a use-case model on screen. |
| Pre-condition | The system is ready and prepared to start drawing the diagram |
| Post-condition | The diagram has been successfully drawn by the system. |

Table 13: use-case (Draw use-case model)

|  |  |
| --- | --- |
| Use case | class diagram |
| Actor | User |
| Description | 1. The system draws a class model after the analysis requirements are completed. 2. The system displays class models on screen. |
| Pre-condition | The system is ready and prepared to start drawing the diagram |
| Post-condition | The diagram has been successfully drawn by the system. |

Table 14: use-case (Draw class model)

## Requirements Modeling

### Use case diagrams



Figure 17:Devolper use-case diagram

The developer upload dataset and then the data uploading to Training Dataset and Testing Dataset The data load to use to train Model and test model and Prepare data the train Model load data from training dataset, the training model learning patterns, and relationships within the data, which is a form of analysis. the developer analyzes how well the model is learning from the data and assesses the performance of the trained model using a separate testing dataset after the Analysis It contains Prepare and Tokenization, Clean Data Extraction NERs Relation Extraction Event and then After analysis is completed generate Diagram Specs get all the Analysis specifications and draw analysis model uses this specifications to draw the diagram, test model it use separate dataset not used during training Model represent real-world scenarios, The developer applies the loaded model to the testing dataset to make predictions and analyzes the model's predictions evaluates its performance using such as accuracy, precision, recall, and F1 score,



Figure 18: user use-case diagram

The user can enter requirements and then Upload data and then load data to analyze by Prepare and Tokenization, Clean Data Extraction NERs Relation Extraction Event and then After analysis is completed generate Diagram Specs get all the Analysis specifications and draw analysis model uses these specifications to draw the diagram whither it is Usecases diagram, class diagram depends on the user what need

### Class Diagram



Figure 19: class diagram

The GenerateDiagramSpecs class identifies the characteristics of diagram generator we take these properties from class analyze and use class diagram.

The class analyze consists of five classes:

1. ExtractNERs class:

This class identifying and classifying entities from the text.

1. ExtractEvent class:

This class is about finding events from the text.

1. ExtractRelations class:

This class identifies and classifies relationships between entities.

1. DataPreparation class:

This class does clean data and Tokenization for text.

1. ExtractRelations class:

This class identifies and classifies relationships between entities.

Class diagram take name and the boundary of diagram and drawing the diagram such as class digarm and use case .

User class it allows to user to entre requirement.

Developer class it allows to train and test the model and he can upload the data set.

DBMS class it can opload and load the data .

Model class it can evaluate the dataset after train and test the data

Trin class train and test do testing training for data.

## Summary

In this chapter, we analyzed the system, identified the methodology of analysis and development, identified the functional and non-functional requirements, and displayed them in the format of tables and UML diagrams such as use case and class diagram.

After analyzing the system, we will be ready in chapter 4 to design the system and determine the architecture of the system and design the user interface.

# Chapter 4: System Design

## Introduction

This chapter covers system design, including structural design, component design, data modelling, and user interface design. Structural design focuses on organizing the system's structure to ensure efficient performance, while component design addresses the interaction of software components to achieve functional goals. Data modelling design deals with efficiently organizing and storing data, and user interface design highlights improving the user experience through an effective and attractive interface.

## Design Methodology



Figure 20:Design methodolgy ((\*): It will be worked on in the second term)

**Architectural Design:**

The process of determining the overall structure and organization of a software system, involving high-level decisions on system configuration, key component specification, and their interactions.

**Component Design:**

The process of dividing a system into manageable units, with each unit representing a specific module or function within the software.

**Data Modeling Design:**

The process of defining and organizing data requirements for a system, creating a conceptual representation of data, defining relationships between entities, and specifying data storage and access.

**User Interface Design:**

The process of creating an easily understandable and visually appealing interface for users to interact with software, including the design of structure, navigation, and visual elements. Goals include enhancing user experience through user-friendly and efficient interface design.

## Architectural Design

The architectural design is concerned with understanding how a system is organized and designing the overall structure of the system, it involves identifying major system components and their communications.



Figure 21: Architectural Design

We opted for the client-server architecture because it allows for the distribution of tasks between the client and the server. The client requests services or data from the server, and the server fulfils these requests. In our system, we will employ the 3-Tier client-server model.

The second tier consists of a layered model with six layers, representing a sequence of steps that can be gradually executed on the data sent from the client. This division facilitates a comprehensive and organized examination, improving the quality of the transmitted data.

In our system, the client in the first tier sends requirements. In the second tier, these requirements undergo examination and analysis before reaching the third tier, which is our system responsible for fulfilling the client's requests.

## Component Design

Our system is divided into two main components: document analysis and drawing diagram. Each component has several classes. The component document analysis includes preparing Data, Analyse, training the model, testing the model, and uploading the Dataset. The component diagram drawing includes a draw analysis model, use case diagram and class diagram.

This phase will be executed in the second stage of the project due to time constraints.

## Data Modeling Design

Designing a data model involves creating a structure to organize data in the system for efficient storage and management. We'll divide the data into two parts: one for training the model and another for testing it. The aim is to use the training part to teach the model using available data, adjusting its settings to improve performance. On the flip side, we use the testing part to see how well the model works on new data. This helps by giving the model new data and checking how accurately it can make predictions. This split lets us check the model's efficiency and gives a precise idea of how well it handles new information. It all contributes to making machine learning better and evaluating statistical models.

This phase will be executed in the second stage of the project due to time constraints.

## User Interface Design



Figure 22: Home Page



Figure 23: Input Page



Figure 24: Diagram Page

## Summary

In this chapter, we have created the Architectural design and utilized the client-server pattern. The component and data modeling design will be developed in the next phase. Finally, we have initiated the design of the user interface.

In the upcoming chapter, we will summarize everything we have accomplished since the beginning of the project. We will discuss how we achieved our goals, the lessons Learnt, and future work.

# Chapter 5: Conclusion and Future Work

## Conclusion

In this section, we will summarize all the work we have done from the beginning to the end of the project. We started by identifying the problem and how we would solve it, then defined the project's goals, methodology, and project plan.

In the second chapter, we researched topics related to the problem we are solving, including AI, ML, and NLP, and how to develop a system for artificial intelligence. We also explored similar systems.

In the third chapter, we analyzed existing systems, identified requirements from these systems, conducted brainstorming sessions, and then created the use-case diagram. We described the use cases and drew the class diagram.

In the fourth chapter, we developed the architectural design and user interface design.

## Goals Achieved

We have successfully achieved our project goals. Here's a overview of the objectives and how we accomplished them.

1. **To review previous works that can be related to our project domain:**

We conducted research on technologies related to our project and similar systems.

1. **To finish the Analysis phase of the proposed system to be built:**

We identified requirements after analyzing similar systems and used brainstorming techniques. We also created a use-case and described them.

1. **To do the preliminary Design tasks of the suggested system:**

We developed initial designs for the project, including class diagrams, architectural engineering design, and the user interface.

1. **To conclude with the lessons learnt and knowledge and experience gained from working on this project:**

We documented any new knowledge and experience gained from the beginning to the end of the project.

## Lessons Learnt

1. **Working as a team-** This project helped us to work together, learn and benefit from each other when we needed each other.
2. **Project planning**- involves setting the goals of the project, creating a task list for each goal, and developing a weekly schedule. Tasks are reviewed and assigned every weekend, and completed work is presented to the supervisor for feedback and approval.
3. **Learn NLP**- NLP is a new system for us, and we did not study anything about it in our university courses.
4. **Improve writing skills-** One of the most important things we learned in our project is to improve writing skill.
5. **Learning System Development for AI (NLP, ML)**- taught us the development steps in the fields of Natural Language Processing (NLP) and Machine Learning (ML).
6. **Researched and Read papers**- Acquired knowledge of methods for searching for reliable references and extracting information from them.
7. **Writing References**- Acquired knowledge of how to write references in the IEEE style.
8. **Every software project has different types of requirements**.

## Limitations and Future Work

### Limitations

The work we haven't done yet is component design and data modeling design due to time constraints. We will implement them in the second phase of the project.

### Future work

The tasks we will undertake in the second phase of the project include:

1. Improving analysis and design, which we worked on in this phase of the project.
2. Component design
3. Data modeling design
4. Project development
5. Project testing

# References

|  |  |
| --- | --- |
| [1] | H. Sofian, N. A. M. Yunus and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," *IEEE Access,* vol. 10, pp. 51021-51040, 2022. |
| [2] | R. S. e. Bashir, "UML models consistency management: Guidelines for software quality manager.," *International Journal of Information Management,* vol. 36, no. 6, pp. 883-899, 2016. |
| [3] | T. w. l. t. p. o. f. t. a. o. technology, "The world's largest technical professional organization for the advancement of technology," IEEE. |
| [4] | university of BATH, 2017. [Online]. Available: https://www.bath.ac.uk/publications/library-guides-to-citing-referencing/attachments/ieee-style-guide.pdf. |
| [5] | ". R. M. A. D. Library. [Online]. Available: https://dl.acm.org/. |
| [6] | G. F. Luger., Artificial intelligence: structures and strategies for complex problem solving., Addison-Wesley Publishing CompanyUnited States, 2008. |
| [7] | M. O. I. Bashir, "ResearchGate," [Online]. Available: https://www.researchgate.net/figure/Different-branches-of-AI-13\_fig1\_357512563. |
| [8] | 2. W. A. Hany H Ammar1, Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems, 2013. |
| [9] | S. Shafiq, A. Mashkoor, C. Mayr-Dorn and A. Egyed, "A Literature Review of Using Machine Learning in Software Development Life Cycle Stages," *IEEE Access,* vol. 9, pp. 140896-140920, 2021. |
| [10] | H. a. Y. N. A. M. a. A. R. Sofian, "Systematic mapping: Artificial intelligence techniques in software engineering," *IEEE Access,* vol. 10, pp. 51021-51040, 2022. |
| [11] | Z. a. L. F. a. Y. W. a. P. S. a. Z. J. Li, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems,* 2021. |
| [12] | JayeshBapuAhire, "Data Science Central," [Online]. Available: https://www.datasciencecentral.com/the-artificial-neural-networks-handbook-part-1/. |
| [13] | V. Nasteski, "An overview of the supervised machine learning methods," *Horizons. b,* vol. 4, pp. 51-62, 2017. |
| [14] | Y. Vasiliev, Natural language processing with Python and spaCy: A practical introduction., No Starch Press , 2020. |
| [15] | G. L. S. C. &. H. O. S. Moon, "Automated construction specification review with named entity recognition using natural language processing," *Journal of Construction Engineering and Management,* vol. 147, no. 1, p. 04020147, 2021. |
| [16] | S. &. C. C. Jaf, "Deep learning for natural language parsing," *IEEE Access,* vol. 7, pp. 131363-131373, 2019. |
| [17] | R. P. S. C. E. &. T. R. Sukthanker, "Anaphora and coreference resolution: A review," *Information Fusion,,* vol. 59, pp. 139-162, 2020. |
| [18] | P. M. Marta Maślankowska, "neurosys," [Online]. Available: https://neurosys.com/blog/intro-to-coreference-resolution-in-nlp. |
| [19] | C. Manning, "Stanford lecture (CS224n) by Christopher Manning," 2019. [Online]. Available: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1162/handouts/cs224n-lecture10-coreference.pdf. |
| [20] | S. G. L. &. H. C. Loáiciga, "What is it? Disambiguating the different readings of the pronoun 'it.," *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* pp. 1325-1331, 2017. |
| [21] | B. Arendse, "A thorough comparison of NLP tools for requirements quality improvement," 2016. |
| [22] | S. Vajjala, B. Majumder, A. Gupta and H. Surana, "Practical Natural Language Processing: A Comprehensive Guide to Building Real-world NLP Systems.," O'Reilly Media, 2020. |
| [23] | "lucid collaborative ai," innovation training, [Online]. Available: https://www.innovationtraining.org/lucid-collaborative-ai/. |
| [24] | "lucidchart vs lucidspark," innovation training, [Online]. Available: https://www.innovationtraining.org/lucidchart-vs-lucidspark-when-why-and-how-to-use-them-both/. |
| [25] | "UML/Code Generation Tool," visual paradigm, [Online]. Available: https://www.visual-paradigm.com/features/code-engineering-tools/. |
| [26] | "Defining design pattern," visual paradigm, [Online]. Available: https://www.visual-paradigm.com/support/documents/vpuserguide/26/36/6246\_definingdesi.html. |
| [27] | "starUML in 2022," Pat research, [Online]. Available: https://www.predictiveanalyticstoday.com/staruml/#content-anchor. |
| [28] | "Introduction to Enterprise Architect," sparx systems, [Online]. Available: https://sparxsystems.com/enterprise\_architect\_user\_guide/13.0/. |
| [29] | "MagicDraw," Dassault Systemes, [Online]. Available: https://www.3ds.com/products-services/catia/products/no-magic/magicdraw/. |
| [30] | S. Ian, Software Engineering, 11th ed., Addison-Wesley,, 2015. |
| [31] | R. J. Leach, Introduction to Software Engineering, 2016. |
| [32] | J. T. Catanio, "Requirements analysis: A review," *Advances in Systems, Computing Sciences and Software Engineering,* pp. 411-418, 2006. |