



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 Additionally, sometimes the techniques or tools used in the analysis and design process are ineffective or outdated.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](#_Toc158653165)

[Acknowledgement iii](#_Toc158653166)

[List of Figures vi](#_Toc158653167)

[List of Tables vii](#_Toc158653168)

[List of Abbreviations viii](#_Toc158653169)

[1 Chapter 1: Introduction 1](#_Toc158653170)

[1.1 Introduction 1](#_Toc158653171)

[1.2 Problem Definition 1](#_Toc158653172)

[1.3 Project Aim and Objectives 2](#_Toc158653173)

[1.4 Project Methodology 2](#_Toc158653174)

[1.5 Project Timeline 4](#_Toc158653175)

[1.6 Document Organization 5](#_Toc158653176)

[1.7 Summary 6](#_Toc158653177)

[2 Chapter 2: Literature Review 7](#_Toc158653178)

[2.1 Introduction 7](#_Toc158653179)

[2.2 Research Methodology 7](#_Toc158653180)

[2.3 AI 9](#_Toc158653181)

[2.4 Machine Learning 10](#_Toc158653182)

[2.4.1 Neural Networks (NNS) 10](#_Toc158653183)

[2.4.2 Supervised Learning 11](#_Toc158653184)

[2.5 NLP 12](#_Toc158653185)

[2.6 System Development for AI 12](#_Toc158653186)

[2.6.1 NLP application Development 12](#_Toc158653187)

[2.6.1 ML application Development 14](#_Toc158653188)

[2.7 Similar Tools 15](#_Toc158653189)

[2.7.1 System 1 15](#_Toc158653190)

[2.7.2 System 2 15](#_Toc158653191)

[2.7.3 System 3 15](#_Toc158653192)

[2.8 Summary 24](#_Toc158653193)

[3 Chapter 3: System Analysis 26](#_Toc158653194)

[3.1 Introduction 26](#_Toc158653195)

[3.2 Methodology 26](#_Toc158653196)

[3.2.1 SDLC 26](#_Toc158653197)

[3.2.2 Analysis 27](#_Toc158653198)

[3.3 Analysis of Existing Systems 29](#_Toc158653199)

[3.4 Requirements Elicitation 30](#_Toc158653200)

[3.4.1 System Requirements 31](#_Toc158653201)

[3.4.2 Functional Requirements 31](#_Toc158653202)

[3.4.3 Non-Functional Requirements 32](#_Toc158653203)

[3.5 Requirements Specification 33](#_Toc158653204)

[3.6 Requirements Modeling 37](#_Toc158653205)

[3.6.1 Use case diagrams 37](#_Toc158653206)

[3.6.2 Class Diagram 39](#_Toc158653207)

[3.7 Summary 40](#_Toc158653208)

[4 Chapter 4: System Design 41](#_Toc158653209)

[4.1 Introduction 41](#_Toc158653210)

[4.2 Design Methodology 41](#_Toc158653211)

[4.3 Architectural Design 42](#_Toc158653212)

[4.4 Component Design 43](#_Toc158653213)

[4.5 Data Modeling Design 44](#_Toc158653214)

[4.6 User Interface Design 44](#_Toc158653215)

[4.7 Summary 46](#_Toc158653216)

[5 Chapter 5: Conclusion and Future Work 47](#_Toc158653217)

[5.1 Conclusion 47](#_Toc158653218)

[5.2 Goals Achieved 47](#_Toc158653219)

[5.3 Lessons Learnt 48](#_Toc158653220)

[5.4 Limitations and Future Work 48](#_Toc158653221)

[5.4.1 Limitations 48](#_Toc158653222)

[5.4.1 Future work 49](#_Toc158653223)

[6 References 50](#_Toc158653224)

# List of Figures

[Figure 1: Project Methodology 3](#_Toc158653113)

[Figure 2: Methodology Schema 9](#_Toc158653114)

[Figure 4: Neural networks architecture [12] 11](#_Toc158653115)

[Figure 5: Supervised learning process [13] 12](#_Toc158653116)

[Figure 5 Proposed Solution Design 16](#_Toc158653117)

[Figure 6 Process flow with NLP And ML Model 17](#_Toc158653118)

[Figure 8 Sample dataset 19](#_Toc158653119)

[*Figure 10 confusion matrix for the case of binary classification* 24](#_Toc158653120)

[Figure 11: waterfall for project ((\*): It will be worked on in the second term). 27](#_Toc158653121)

[Figure 12: Description of stages of analysis 28](#_Toc158653122)

[Figure 13:Devolper use-case diagram 37](#_Toc158653123)

[Figure 14: user use-case diagram 38](#_Toc158653124)

[Figure 15: class diagram 39](#_Toc158653125)

[Figure 16:Design methodolgy ((\*): It will be worked on in the second term) 41](#_Toc158653126)

[Figure 17: Architectural Design 43](#_Toc158653127)

[Figure 18: Home Page 44](#_Toc158653128)

[Figure 19: Input Page 45](#_Toc158653129)

[Figure 20: Diagram Page 45](#_Toc158653130)

# 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

Nowadays, the field of software development is using the capabilities of artificial intelligence to improve the phases of this field, including prediction and decision-making assistance. Artificial intelligence has become instrumental in guaranteeing the high quality of software development phases. Additionally, it has the capacity to significantly reduce the time needed for software development, leading to increased overall productivity. Furthermore, AI can effectively detect and pinpoint errors within the development process, enabling timely rectification. It also contributes to the improvement of specific components within various phases of software development. [1]

In this project, we pursue to merge knowledge from two major disciplines of computer science: Artificial Intelligence and Software Engineering in order to facilitate software development. Particularly, employing the power of artificial intelligence to transform project analysis textual elements: scenarios and requirements into visual modeling representations (Usecases and Class Diagrams).

This chapter highlights the goal and objectives of this project, outlines our methodology to achieve these objectives, gives the plan for the project’s tasks, provides an overview of the subsequent chapters in this report, and finalizes with an overall summary of the chapter.

## Problem Definition

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. 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. To facilitate this understanding, various modeling languages have surfaced, enabling graphical representations that substantially contribute to an enhanced comprehension of software systems. [2]

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.

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.

## Project Aim and Objectives

The aim of the project is to complete the analysis, modeling, and data gathering for a system that converts textual requirements into use cases and class diagrams, with the aim of expediting and facilitating the work of systems modeling analysts and saving them time.

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

1. To review previous works that can be related to our project domain.
2. To finish the Analysis phase of the proposed system to be built.
3. To do the preliminary Design tasks of the suggested system.
4. To conclude with the lessons learnt and knowledge and experience gained from working on this project.

## Project Methodology

Reviewing previous works is a fundamental step in our project's research phase. By examining existing research and projects in our project domain, we can gain valuable insights into the state of the art, identify potential challenges, and build upon the knowledge and findings of others. This process allows us to ensure that our project is both innovative and informed by the latest developments in the field.

Completing the Analysis phase is a critical milestone in our project's development. During this phase, we thoroughly examine the project requirements, gather user needs, and analyze existing systems and processes. It provides the foundation for all subsequent project activities, ensuring that we have a clear understanding of what needs to be achieved and how to achieve it.

After completing the Analysis phase, we move on to the preliminary Design tasks. This phase involves translating the gathered requirements and insights into a comprehensive system design. We will define the system's architecture, user interfaces, data flows, and functionality. This step is pivotal in transforming our project concept into a concrete plan for implementation.

Concluding the project with a comprehensive review of lessons learned, knowledge gained, and experiences accumulated is essential for continuous improvement. This reflection allows us to identify areas of success and areas for improvement in project management, technical implementation, and collaboration. By documenting our insights and best practices, we can enhance our future projects and contribute to the growth of our team's expertise.



Figure 1: Project Methodology

## Project Timeline

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **December** | | | | **November** | | | | **October** | | | | **September** | | **Months**  **(2023)** |
| **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and identify the problem | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review related literature and existing systems |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Analysis and document system requirements |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Design the system structure and interfaces |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Documentation |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Presentation |

Table 1: Project plan v1

1. **Defining Project Objectives and Identifying the Problem (1 week):**

This time is needed to understand the project scope and specific objectives, as well as to analyze the problem thoroughly.

1. **Reviewing Relevant Literature and Theories (4 weeks):**

Allowing this time enables thorough research and review of literature related to the project, including studying relevant theories.

1. **Analyzing and Documenting System Requirements (4 weeks):**

More time is allocated to this stage as it involves a detailed analysis of functional and non-functional system requirements and their precise documentation.

1. **Designing System Structure and Interfaces (3 weeks):**

This time allows for the precise design of the system's structure and user interfaces, including working on their details.

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **December** | | | | **November** | | | | **October** | | | | **September** | | **Months**  **(2023)** |
| **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **2** | **1** | **4** | **3** | **Weeks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Define project objectives and identify the problem | **Tasks** |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Review related literature and existing systems |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Analysis and document system requirements |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Design the system structure and interfaces |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Documentation |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  | Presentation |

Table 2: Project plan v2

1. The presentation has been rescheduled to take place between the second and third weeks of December instead of the third and fourth weeks of December. This is the specified date from the university for the presentation.
2. The third task of the project will now commence from the third week of November until the first week of December, instead of from the fourth of November to the second of December. This adjustment is due to the modification in the presentation date.
3. In the third week of November, we will be nearing the completion of system analysis and will start the architecture design, connecting the analysis and design phases.

## 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 the literature and learned about the techniques used. We reviewed and compared systems similar to ours.
* **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.** 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]

**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.

## 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]

## 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.

### 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.

1. **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.

1. **Feature Selection:**

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

1. **Data Splitting:**

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

1. **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.

1. **Performance Evaluation:**

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

1. **Improvement and Adjustment:**

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

1. **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

### System 1

### System 2

### System 3

This study will focus on providing solution to generate Use-Case diagram and Class diagram against the particular business requirement. system will read and understand the business requirement using Natural Language Processing and Machine learning and identifying entities and relationships on that for generate use case diagram. And also, classes and relationship between classes will also be identified to generate class diagram. Then system will show generated diagram with including user interacting feature where user can add additional element and add changes to use case or class diagram or edit existing element and its attribute or relations.



Figure 5 Proposed Solution Design

As showing in the Figure 6 proposed solution design, user can upload a text file containing a scenario or simply copy and paste the text inside. The system extracts word fragments and named entities using the NLP module. Subsequently, the ML module will identify and extract the usecases, actors, classes, associations according to the ML algorithms which have implemented. Plant UML modeling draws the use case and class diagram. The system allows editing of the generated diagrams and the user can modify and adjust the use cases, actors, classes, attributes and all types of relationships. Therefore, the output is a highly customizable output that the user will be able to achieve the desired output according to the business requirements provided.

**Methodology**

The text assure the importance of efficiently extracting information from text data, noting that manual text searching can be difficult and time-consuming, which may lead to important information being skipping. Therefore, providing an automated way to extract information from texts and present it in an organized manner can provide many benefits and save time spent browsing text documents. Information mining technology aims to achieve this goal by systematically analyzing texts and identifying relevant information. The example provided assure the importance of understanding the parts of speech in a sentence to accurately interpret the meaning.

Algorithmic design 

Figure 6 Process flow with NLP And ML Model

Figure 6 shows the process flow with Natural language processing and Machine learning model where several preprocessing stages has involved before proceeding to the ML based information extraction.

The important information in requirement texts usually follows certain patterns. When we look at the Parts of Speech tagging of a sentence, they follow specific patterns too. If we understand these patterns, we can easily get the information we need. This helps us create use case and class diagrams for the project.

In algorithmic design, we can choose any Parts of Speech tagger modules for English literature. We mark words in a story based on their meanings and how they're used in specific parts of speech.

Some POS tagging examples can be mention as: Coordinating Conjunction (CC), Cardinal Digit (CD), existential (EX), adjective (JJ), modal (MD) such as could, will etc., proper noun (NNP), predeterminer (PDT), possessive pronoun (PRP), etc. (Rachiele, G.2018) To assign grammatical information to each word in a sentence we used POS taggers

Most of the information in a requirement text is made up with verb, gerund/present (VBG), noun plural (NNS), TO, verb base (VB), noun (NN), preposition or subordinating conjunction with these pos taggers. But they come with different patterns.



Dataset would be like as Figure 7 shown below according to the above requirement text.



Figure 7 Sample dataset

For now, according to the given sample, there are 3 kinds of specific patterns that can identify:

• {VBG>NNS>TO>VB>NN}

• {NN>IN>DT>NN}

• {VB>DT>NNS>TO>VB}

The process involves finding the key terms for class or use case diagrams using machine learning, which makes this task a classification one. Data is divided into two sets: training and testing. The trained model uses features from the training instances to predict terms for unseen data. Comparing predicted and actual terms helps evaluate the model's performance. Statistical techniques rely on adjustments between classification results using probability and costs for decision-making.

Preprocess Functional requirement texts

The text discusses the process of text preprocessing, where ambiguity resulting from the use of multiple forms of a certain verb or the singular/plural form of a word is reduced. Additionally, common words like "a," "the," "of," and "is" do not contribute much information to our summarization goal. Below are several operations commonly used for document preprocessing.

**Document segmentation:**

A text is divided into several paragraphs to find where each sentence is placed in its respective paragraph.

**Stemming**:

We apply stemming to bring a word to its root or base form. The examples include use of a singular form rather than using a plural, or removal of ‘ing’ from a verb. To this end, Stanford NLP stemmer is employed in this paper. Paragraph segmentation. Paragraph segmentation divides a paragraph into sentences using sentence tags.

**Word normalization:**

Each sentence consists of multiple normalized words. Through normalization and lemmatization individual words become one common form, stemming down to their roots. Ambiguities are removed by Porter’s algorithm.

**Stop word filtering:**

Stop words can be filtered out after performing other preprocessing steps. There is no uniform rule for selecting a stop word because it depends on individual tasks. In this work, words such as a, is, in, the, of are selected as stop words and are filtered out from the document. In text mining applications stop word filtering is considered as a standard step.

Machine Learning Techniques to Identify Diagram Elements

1. Classification Model
2. Recurrent neural network Unit
3. RNN based Sequence to Sequence Model
4. Strategies of training

IMPLEMENTATION:

The initial dataset is divided into subsets containing functional requirement sentences for each element in the diagram, specifically actors, use cases, and classes. Each subset is further divided into validation, training, and test pairs. Separate machine learning models are built for each diagram element using the training subsets. During implementation, data preprocessing is conducted using Python programming, which involves cleaning and preparing the data for analysis.

check the words in the english dictionary and removing unused punctuation, removing stopwords etc. are the preprocessing stages done in the implementation.

Thereafter extracting feature task was perform where it checks existence of whole keyword and also existence of all parts of known keywords were performed.

The recurrent words such as propositions don’t participate as key terms. Calculating the term frequency and inverse document term frequency (TF X IDF), these expressions can dominate other meaningful expressions. Consequently, to deal with some challenging terms we can eradicate from script by allocating probability assessment of zero and also, numerals and nonalphanumeric characters will be removed.

Thereafter naïve based classifier has been implemented and as previous stages clean and tokenization tasks also perform before generate predicted tags based on decision rule

In the decision rule method, it takes the functional text, and the keywords from the early stage perform and also stopwords. First it gets scores of each whole keywords from the functional text provided. Then get scores for each known keywords with all parts in the requirement sentence. After that, three steps performed as.

1. adding tag if posterior probability is higher than or equal to 0.5
2. add highest scoring to TF-IDF keywords if not already added.
3. Add ‘c#’ if there no tags (‘c#’ just a special keyword to identify it clearly from other keywords)

**Identifying Use case and Class Diagram Relationships with NLP and ML techniques**

In this implementation our focus mainly on identifying relationship in both use case and class diagram related components separately. This is a combination of two models where one model will be identifying relationship involved entities. As an example of use case diagram first model will identify what are the actors and use cases involve in the relationship. Second model will be identifying the relationship type of that particular relationship. These two models need to build for both use case and class diagram separately. Two models per each diagram type to identify relationship involved in particular functional requirement text.

Similar to the previous implementation, in this iteration, we again divide the initial dataset into subsets containing functional requirement sentences for each diagram element. This time, we focus on use case relationships and class relationships. Each dataset is then separated into two subsets for the two models: one with entities involved in the relationship and the other with the relationship type along with the functional text. These subsets are divided into validation, training, and test pairs, with the same distribution as before. Two separate machine learning models are built for each diagram type using these training datasets.

**Draw diagram with identified diagram elements**

With this implementation we want to generate the usecase and class diagram according to identified elements through previous two implementation. Usecase and class diagram elements that has been identified from functional requirement text has outputted as JSON file format.

**Evaluation approach**

To evaluate the Model, we can use the term called Model Evaluation Metrics.

Model evaluation metrics are needed to measure model performance. The choice of evaluation parameters depends on the machine learning task provided. In this project that is classification

**Classification metrics**

When performing predictions through classification, four kinds of result that could occur as follow.

* True positives: are when we assume an observation belongs to a particular class and actually it is belonging to that exact class.
* True negatives: are when we assume an observation is not belonging to a particular class and that actually not belonging to that particular class as expected.
* False positives: occur when we predict our observation belongs to a one particular class but in reality, it is not belonging to that class.
* False negatives: occur when we assume an observation is not belonging to a particular class but actually it is belonging to that same class



*Figure 8 confusion matrix for the case of binary classification*

Accuracy, precision, and recall are the three main metrics that can be used to evaluate a classification model in generally.

1. Accuracy is the percentage of correctly predicted instances in the test data, calculated by dividing the number of correct predictions by the total number of predictions.
2. Precision is the fraction of true positives (relevant examples) among all examples predicted to belong to a particular class, indicating how many relevant items were predicted.
3. Recall is the fraction of test data predicted to belong to a particular class compared to all test data truly belonging to the class, indicating how many relevant items were predicted.

## 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 11: 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 12: 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 13: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 14: 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 15: 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 16: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 17: 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 18: Home Page



Figure 19: Input Page



Figure 20: 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. |