T.C.
ISTANBUL GEDİK UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

A PROPOSAL SOFTWARE PROJECT MANAGEMENT FOR A NEW ARABIC CODING SCHEME

MASTER'S THESIS

Sarah Abdukareem Oglah AL-BUSAEED

Engineering Management Master in English Program

MAY 2021

T.C.
ISTANBUL GEDİK UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

A PROPOSAL SOFTWARE PROJECT MANAGEMENT FOR A NEW ARABIC CODING SCHEME

MASTER'S THESIS

Sarah Abdukareem Oglah AL-BUSAEED
(191281009)

Engineering Management Master in English Program

Thesis Advisor: Asst. Prof. Dr. Umut Hulusi INAN

MAY 2021

T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz, Engineering Management Department İngilizce Tezli Yüksek Lisans Programı (191281009) numaralı öğrencisi Sarah Abdukareem Oglah AL-BUSAEED'in "A Proposal Software Project Management For A New Arabic Coding Scheme" adlı tez çalışması Enstitümüz Yönetim Kurulunun 10.05.2021 tarihli kararıyla oluşturulan jüri tarafından *Oy Birliği* ile Yüksek Lisans tezi olarak *Kabul* edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**

Tez Savunma Tarihi :10/05/2021

1)Tez Danışmanı: …….

2) Jüri Üyesi: …….

3) Jüri Üyesi: …….

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

DEDICATION

I dedicate this research to Allah Almighty my creator, my strong pillar, my source of inspiration, wisdom, knowledge and understanding. He has been the source of my strength throughout this program and on His wings only have I soared. To my role in life and my first beloved **(My Father: ABDULKAREEM)** who has always supported me with all his moral and material support throughout my education. To my dear mother dear **(Halima)**, I also dedicate this work to my husband; **(Emmad)** who has encouraged me all the way and whose encouragement has made sure that I give it all it takes to finish that which I have started. To my daughter **(Fatima Tulzahraa).** My sister and my dear brothers for my heart **(Sally, Ali, Hassan, Hamza)**. Thank you. My love can never be measured for all of you. Allah blesses you.

FOREWORD

This thesis is the end of my journey to obtain my M.Sc. It has been a challenging experience riddle of good and bad moments. However, I have felt supported during this time by several people. This thesis also belongs to them. I would like to express my gratitude to all of these people.

First and foremost, I would like to express my special thanks to my first teacher (Assistant Professor Dr. AbdulKareem Ibadi) who gave me the golden opportunity to do this wonderful project on the topic (A proposal Software Project Management for a New Arabic Coding Scheme) who contributed valuable expertise and excellent advice throughout all experimental and theoretical work in this thesis.

I am greatly indebted to my supervisor (Assist. Prof. Dr.Umut Hulusi INAN) for his support and guidance throughout my research work. It was all fruitful advice during my academic career.

I also appreciate the support of (Assist. Prof Dr. Mert Tolon).

It is with immense pleasure that I dedicate this thesis to my husband (EMMAD), who stood by me during my study and always offered his love, care, and support.

May 2021 **Sarah Abdukareem Oglah AL-BUSAEED**

TABLE OF CONTENT

Page

LIST OF ABBREVIATIONS

ASCII : American Standard Code for Information Interchange
ASMO : Arab Organization for Standard and Metrology
ATMs : Automatic Teller Machines
BinHex : Binary-to-Hexadecimal
BOCU : Binary Ordered Compression for Unicode
CAD/CAE : Computer-Aided Design/ Engineering
CASE : Computer-Aided Software Engineering
CCS : Coded Character Set.
CCSIDs : Code Charactere Set Identifeirs
CDRA : Character Data Representation Structure
CEF : Character Encoding Form
CES : Character Encoding Scheme
codec : Corresponding Coder-Decoder
DFD : Data Flow diagramming
GUI Design : graphical user interface Design
HTTP : Hypertext Transfer Protocol
IBM : International Business Machines Corporation
IEEE : Institute of Electrical and Electronics Engineers
ISO : International Organization for Standardization
KPAs : Key Process Areas
MIME : Multipurpose Internet Mail Extensions
MIME : Multipurpose Internet Mail Extensions
SCSU : Standard Compression Scheme for Unicode
SDLC : Software Development Life Cycle
U.S. english : United States English
UNIX : Is a Family of Multitasking, Multiuser Computer Operating Systems
UTF-16 : 16-bit Unicode Transformation Format
UTF-32 : 8-bit Unicode Transformation Format
UTF-8 : 8-bit Unicode Transformation Format
XML : Extensible Markup Language

LIST OF TABLES

Page

LIST OF FIGURES

Page

A PROPOSAL SOFTWARE PROJECT MANAGEMENT FOR A NEW ARABIC CODING SCHEME

ABSTRACT

The designers of computers tend to use the English language as a base for their design and build the keyboards and their coding scheme and many applications depending on the English features. The localization of applications performed by prebuilt fixed rules to make the Arabic characters as a convenient solution for the English-based keyboard. While there are a lot of problems left without a solution and that's what steal the pretty spirit of the Quran's language and make it boring and uncomfortable within computers. That's what lead most Arabic users to prefer the English language to be used on computers.

In this thesis we design a proposed software project management for a new Arabic coding scheme depend on the Arabic language features to solve a lot of Arabic coding problems especially those which concerned the formalization problems.

In the proposed coding scheme, we used one byte to store the Arabic character instead of two bytes. The proposed coding scheme can be used as a bilingual coding scheme instead of ASCII in an Arabic platform environment or as a text compression system.

Keyword: *ASCII, Arabic characters, Representation, Coding, Software engineering*

YENİ ARAPÇA KODLAMA ŞEMASI İÇİN BİR YAZILIM PROJE YÖNETİMİ ÖNERİSİ

ÖZET

Bilgisayar tasarımcıları, İngilizceyi tasarımlarında bir temel olarak kullanma eğilimindedir ve klavyeleri, kodlama şemalarını ve İngilizce özelliklere bağlı olarak birçok uygulamayı oluştururlar. Arapça karakterleri İngilizce tabanlı klavye için uygun bir çözüm haline getirmek için önceden oluşturulmuş sabit kurallarla gerçekleştirilen uygulamaların yerelleştirilmesi. Çözümsüz kalan pek çok sorun olsa da, Kuran dilinin güzel ruhunu çalan ve bilgisayarlarda sıkıcı ve rahatsız eden şey budur. Arap kullanıcılarının çoğunun bilgisayarlarda kullanılmak üzere İngilizceyi tercih etmesine neden olan şey budur.

Bu tezde, yeni bir Arapça kodlama şeması için önerilen bir yazılım projesi yönetimi tasarlıyoruz, özellikle de formalizasyon problemleriyle ilgili olan birçok Arapça kodlama problemini çözmek için Arapça dil özelliklerine bağlı.

Önerilen kodlama şemasında, Arapça karakteri iki bayt yerine saklamak için bir bayt kullandık. Önerilen kodlama şeması, bir Arapça platform ortamında ASCII yerine iki dilli bir kodlama şeması veya bir metin sıkıştırma sistemi olarak kullanılabilir.

Anahtar Kelime: *ASCII, Arapça karakterler, Temsil, Kodlama, Yazılım mühendisliği*

1. INTRODUCTION

1.1 Preface

Computers have become an effective and influential tool in the development of any sphere of life, but it has become the basis for measuring progress and development. The more computer applications in a particular area, the more theoretical research and studies to adapt the representation of information for storage and processing using computers [1]. However, when they created the computer, Americans and Europeans did not have a future vision of its development and spread because they designed it to suit the needs of their markets, culture, and languages, but the open structure of the personal computer and its rapid development are relatively easy to add software and associated technologies for cultures and languages. Arabic is one of these languages. Arab users suffered from difficulty dealing with computers until localization programs emerged in the 1980s that facilitated the processing of Arabic texts [2] These localization programs were the main reason for the current very normal handling in the areas of editing, printing, and the use of interfaces, databases, web pages, and e-mail in Arabic.

However, the possibilities remain nothing more than a direct localization of English programs or the application of global techniques that are not directly related to the Arabic language and do not go deeper and do not provide the tools to take full advantage of their wealth. The period of the 1990s saw the development of applications in the treatment of the Arabic language itself and provided solutions in the areas of machine translation, spelling, grammar, speech recognition, texting, text optical recognition, pure analysis, research, text retrieval, and automatic Arabic calligraphy [3]. However, the flaw in these solutions is that they are not widely used because they do not provide practical solutions suitable for use and cannot deliver the Arabic language and culture to the world.

The most important difficulties faced by the developers of these technologies are the lack of technical academic research related to Arabic, which is why they simulate applications based on European languages with the representation of an Arabic

craftsman. Although this simulation may succeed in solving one aspect of the Arabic language, it fails to build a complete processing system for it. The formation itself is one of the outstanding problems so far because the solutions used to process The Arabic language are linked to solutions based mainly on languages that are devoid of composition. The current treatment of The Arabic language considers the movement to be an object in itself and can only be dealt with as part of the letters of the word, but dealing with the language without formation is a treatment that lacks perfection and is not without ambiguity in understanding and here it should be noted that the correct and complete treatment of the Arabic language must be delayed Considering the composition as a prerequisite in understanding the Arabic language, i.e. the introduction of Arabic texts or the construction of processes to form texts subjectively, such programs exist, but they are still inaccurate and cannot be relied upon in practical applications. Machine translation to and from The Arabic language is one of the most complex tasks that can be faced by those working in the field of developing Arabic language techniques and accessing practical solutions in this field is one of the very vital images of the user and Arab companies, especially the presence of the Internet and the huge content of information available in it. In multiple languages, some hasty solutions have emerged in the field of translation, but they have not succeeded and developed because they are not relying on in-depth research in linguistics, it is nothing more than an improved version of electronic dictionaries [4]. It has become very clear that the subject of processing the Arabic language computer requires a radical technical revolution through theoretical research derived from the origin of the Arabic language and its rich heritage and based on a based understanding of the language, not adapting or improving what exists to accept many aspects, and this is why we have to enter into the topic of the representation of the language To store it in the computer as an important step ahead of the processing topic, the construction of any computer application requires first the way to represent the information and data of that application in the computer before preparing the methods and algorithms of the processing.

In this thesis, our study focused on finding an appropriate way to represent Arabic letters and their composition by returning to the basis of the Arabic language and its written letters in the past before drip and formation and we reached record results that serve the field of application of computer technologies and contribute to

reducing the volume of Arabic information and data Stored and transferred as an introduction to trying to develop a correct processing of Arabic language computer.

1.2 A Brief History

Shortly after the computer entered into working life in various sectors, the need to use Line Printers in languages that do not use Latin letters was required. At the end of the 1960s, belts were made of Arabic letters. These belts contained a series of Arabic letters, mostly one-shaped each letter (when the first letter was used in the first, middle or final word) due to the small number of places available on those belts and the need to contain English letters (large and small) next to Arabic characters. Reading the prints of those printers was difficult. This attempt required the inclusion of character code locations within the ASCII code tables, which were originally developed for English from the Standardization Institute in the United States of America. It was also necessary to place Arabic letter symbols separate from each other separated by small intervals that might be visible to the reader and may not be visible so that they appeared to be continuous writing.

The manufacturers of these printers have made decisions to determine the number and type of letters contained in the printers they make. But motivated by the marketing of its products, and is a Western company with no experience in Arabic, it started contacting Arab users (especially universities and large computer centers in the Arab world) to help improve the shape, number and distribution of the letters among The Latin characters and began to take place. Improvements to them.

After the widespread use of visual screens, these companies produced screens for Arab users that show Arabic characters in a range of bright spots. An important paradigm shift occurred in the mid-1970s when a company produced an Arabic screen that could show the Arabic character consisting of a luminous matrix so that its shape varies according to the location of the letter from the Arabic word itself. The same idea then moved to printers when dot Matrix Printers were deployed. The number of screen highlights or printer pins was limited first and the shape of the letters was poor. However, after increasing the accuracy of screens and printers, it is possible to print beautiful forms of Arabic letters in different locations, as well as to change the width of the character and add the composition. In the process, computers

had to be added limited intelligence to determine the shape of the letter (among the four forms: primary, intermediate, other, and separate) by its location of the word.

During this period, pressure has increased on the Arab authorities responsible for Arab standards, and discussions and deliberations have increased in Arab seminars and conferences on computers.

The Holy Qur'an is considered one of the first written Arabic book and it was read by the Arabs without ambiguity until the era of the Umayyad Caliph Abd al-Malik bin Marwan (AC685-756/ HAJ 65-86) [5], where diacritics appeared, so that non-Arabs read it correctly in a historical incident mentioned by history books which are out of the scope of this thesis, but rather in determining the outcome of the Arabic language after that incident, which is the emergence of Arabic letters in a clear new outfit. After diacritics, the number of characters is increased from 15 to 28 characters, where each character has become with a single pronunciation. Now, anyone can read Arabic words correctly even if he couldn't understand Arabic but knows how to draw those letters, and the way they are pronounced. Tables 1.1.a, shows the Arabic characters before diacritics where 11 characters (that has two or three pronunciations), Tables 1.1.b, shows the letters with single pronunciation.

Table 1.1: Arabic Letters before Punctuation

ح	د	ر	س	ص	ط	ع	ف	ل	ن	ي

a: Arabic characters with more pronunciations

ه	و	م	ا

b: Arabic characters with single pronunciation

1.3 Arabic Standard Specifications

In 1981, the Arab Organization for Standards and Standards, which was affiliated with the Arab League and based in Jordan, formed a committee to determine the Arabic standard for Arabic letters in the field of information. A series of these specifications were issued from 1981, most recently in November 1986, when the organization approved the standard specifications (ASMO 708) for the exchange of information on the computer in eight binary numbers and recorded globally under the number (ISO/8895-6). In this specification, it contains 120 characters, plus the eight

modulation tools, which are in succession: you intend to open, then you intend to annex, then you intend to break, and then the opening, the join, the fraction, the intensity, the stillness. The Arabic numbers in the (ASMO 708) English numerals occupied the same English numbers. This specification is characterized by [6]:

- The presence of a location for each of the 28 characters.
- The presence of six sites of al-Hamza in different forms أ إ ؤ ئ آ
- Having a tethered t-site and another for a thousand cabins.
- • The presence of eight formation sites is ً ٌ ٍ َ ُ ِ ّ ْ in the order shown.
- The existence of the hiving in the ASCII table position between the connected letters.

As this standard is more than 30 years late for the introduction of Arabic in computers, computer manufacturers have developed their specifications, making it difficult for them to undo them because of the high costs. Therefore, several sets of character systems continued to exist, depending on the computer manufacturers.

In according to the standard 708, several Arab groups were issued, including Arab Window Collections in Bahrain, Sakher in Saudi Arabia, IBM, and Microsoft Americas. The latter group began to spread more than others after the tyranny of windows and it is feared that it will remain dominant, overwhelming, and canceled in all or most other forms. It is noted among the localization systems that the question of the order of alphabets depends on the group used, since all these totals have considered the composition to be letters, so the order of letters at the writing is considered the composition of a letter to be taken into account in the order. It results in the group itself determining the order method. If the letters are scattered in the group table, the order will be problematic, especially if the character is taken in the form of the letter according to its location in the table, as in the group number 864 for I.B.M. The locations of the special letters (ء, ة, ى, ـ) all vary from group to group, generating a different order when using one and switching to the other.

1.4 The Proposed Represent for Arabic Diacritics Language

The Holy Qur’an is considered one of the first written Arabic books and it was read by the Arabs without ambiguity until the era of the Umayyad Caliph Abd al-Malik

bin Marwan, where punctuation and composition appeared, so that non-Arabs read it correctly in a historical incident mentioned by history books we are not in the process of mentioning them, but rather in determining the outcome of the Arabic language After that incident, before this historical incident the Arabic alphabet contains 15 letters only and most of them has more than one pronunciations, during the reign of the Caliph Abd al-Malik bin Marwan diacritics are proposed to let each character has single pronunciation. The number of Arabic characters became 28 characters.so, we make use of these extensions, before diacritics the ambiguous (has more than one pronunciation) characters are 10 characters shown in Table 1.1 while the rest of the letters are six letters. Note Tables 1.2

Table 1.2: Dictionary Letters

س	ن	ف	ح	ي	ع	ط	د	ر	ص

Table 1.3: The Answer Letters

ه	و	ك	م	ل	ا

1.5 Problem Definition

- The lack of technical academic research related to the Arabic language which led them to simulate applications based on European languages with Arabic literal representation. Although this simulation may succeed in solving an aspect of the Arabic language, it fails to build a complete treatment system for it.
- The subject of computer-based Arabic language processing requires a radical technical revolution through theoretical research drawn from the origin of the Arabic language and its rich heritage. This heritage must set up on a basis to understand the Arabic language. Not on adapting or improving what exists of application.
- As an important step that precedes the topic of treatment, the fact that building any computer application first requires a way to represent the information and data of that application in the computer before preparing methods and algorithms for processing.

1.6 Aim of Thesis

- Representing information for storing and processing it using a computer
- Representing the Arabic language for storing it in the computer as an important step that precedes the subject of processing, as building any computer application first requires a way to represent information and data of that application in the computer before preparing processing methods and algorithms.

1.7 Literature Review (Previous Studies)

1. Arabic Text Lossless Compression by Characters Encoding: The aim of this paper is to suggest a new sequential encoding technique that efficiently converts Arabic characters string from UTF-8 to ANSI characters coding. The encoding algorithm presented in this paper significantly reduces the file size. The decoding method transforms the encoded ANSI string back to its original format. Unlike the one-byte ANSI characters, Arabic alphabets are currently being stored in 2 bytes size which leads to inefficient space utilization. The newly developed sequential encoding technique reduces the space required for storage up to fifty percent. In addition, the proposed technique will retain the Arabic encoded text to its original form after decoding, which is leading to a lossless text compression. Thus, addressing the common concern of the currently available Arabic characters compression techniques [7].
2. Dynamic with Dictionary Technique for Arabic Text Compression: In this research paper we build a new, reliable, and sufficient algorithm for Arabic text language. The proposed algorithm should combine the features of the Huffman and Lempel Ziv algorithms, and is expected be able to reduce the general compression ratio. Our approach is different from Huffman algorithm in the sense that it assigns codes to n-gram symbols where n is a positive integer that is greater than or equal to one. Compared to Huffman algorithm, which assigns a code to each symbol individually, our approach is expected to assign codes to symbols in average. Our approach is different from Lempel Ziv algorithm in the sense that the size of dictionary that we build does not

grow in an uncontrolled manner. The size of the dictionary is fixed and its size can be expected prior to process the text files that are to be compressed. This is because the size of each word in the dictionary we build is fixed and is equal to n. So for example, given that the number of different symbols in the text file at hand is m and that n is 2, the total number of entries in the dictionary that we propose to build will be m*m in the worst case [8].

3. A new localization and compression system: Computer designers have deliberately used the English language as a basis in design, its features and characteristics, they put coding and character distribution systems on the keyboard and built all systems and applications. As for applications, localization processes are carried out using ready-made templates and rules to match the features and characteristics of the English language and did not solve the problems resulting from this compatibility as a problem Formation, which contributed to stealing the beautiful spirit of our beloved language and making it boring in computer applications and uncomfortable to use, and everyone preferred to use the English language in computer applications. In this paper, we devised a coding system for Arabic letters derived from the characteristics of the Arabic language, and we dealt with the issue of diacritics on the basis that they are characteristic of letters and not stand-alone letters. The coding method in this research is based on the consideration that the Arabic letter is the basis and the rest of the letters of other languages are secondary, or it is an expansion of the Arabic letters and this is because the Arabic language is wider and its letters more. In the new coding, the Arabic letter with its movement became stored in a single letter instead of two letters, and the unformatted Arabic letter was stored in half of the storage space required to store one letter approximately. The inferred system can be an alternative to the ASCII table currently used in character encoding as a bilingual encoding system. It can also be used as a multilingual encoding system (for more than two languages) or as a new localization system with a self-spelling correction or as a text compression system [6].

4. Multilayer Model for Arabic Text Compression: This article describes a multilayer model-based approach for text compression. It uses linguistic information to develop a multilayer decomposition model of the text in order

to achieve better compression. This new approach is illustrated for the case of the Arabic language, where the majority of words are generated according to the Semitic root-and-pattern scheme. Text is split into three linguistically homogeneous layers representing the three categories of words: derivative, non-derivative and functional words. A fourth layer, called the Mask, is introduced to aid with the reconstruction of the original text from the three layers in the decoding side. Suitable compression techniques are then applied to the different layers in order to maximize the compression ratio. The proposed method has been evaluated in terms of the rate of compression it provides and its time efficiency. Results are shown along with real texts to illustrate the performance of the new approach. The novelties of the compression technique presented in this article are that (1) the morphological structure of words may be used to support better compression and to improve the performances of traditional compression techniques; (2) search for words can be done on the compressed text directly through the appropriate one of its layers; and (3) applications such as text mining and document classification can be performed directly on the compressed texts [9].

5. Hybrid Technique for Arabic Text Compression: Arabic content on the Internet and other digital media is increasing exponentially, and the number of Arab users of these media has multiplied by more than 20 over the past five years. There is a real need to save allocated space for this content as well as allowing more efficient usage, searching, and retrieving information operations on this content. Using techniques borrowed from other languages or general data compression techniques, ignoring the proper features of Arabic has limited success in terms of compression ratio. In this paper, we present a hybrid technique that uses the linguistic features of Arabic language to improve the compression ratio of Arabic texts. This technique works in phases. In the first phase, the text file is split into four different files using a multilayer model-based approach. In the second phase, each one of these four files is compressed using the Burrows-Wheeler compression algorithm. [10]

6. WordCode using WordTrie: Computers work with text data by assigning a code for each character, called encoding. Characterencoding techniques emerged in the late 1960s, and a similar type of technique is still used to

encode text data. Computers can only understand alphabets, not words. In this article, we develop an approach that enables computers to understand words. We introduce a word-based encoding of text data named WordCode. WordCode encodes the most frequent set of characters (i.e., words) found in Internet directories with a dynamic code combination. Although some dictionary-encoding techniques have been proposed, we still tend to use character encoding, such as Unicode, to encode text data. Dictionary-encoding techniques have not been adopted due to the massive size of the code page and the complexity in accessing the code page. In this article, we introduce a customised trie named WordTrie to store words for faster encoding and decoding. We generate the code combination in such a way that the size of the WordCode for a word is always smaller than the total size of the character coding. Our experimental results from encoding text files from the Gutenberg corpus, Canterbury corpus, large corpus, Calgary corpus and Silesia corpus using WordCode show an up to 19.9% reduction in file size with respect to characterbased encoding. This smaller file size mean [11].

7. Dynamic with Dictionary Technique for Arabic Text Compression: In this research paper we build a new, reliable, and sufficient algorithm for Arabic text language. The proposed algorithm should combine the features of the Huffman and Lempel Ziv algorithms, and is expected be able to reduce the general compression ratio . Our approach is different from Huffman algorithm in the sense that it assigns codes to n-gram symbols where n is a positive integer that is greater than or equal to one. Compared to Huffman algorithm, which assigns a code to each symbol individually, our approach is expected to assign codes to symbols in average. Our approach is different from Lempel Ziv algorithm in the sense that the size of dictionary that we build does not grow in an uncontrolled manner. The size of the dictionary is fixed and its size can be expected prior to process the text files that are to be compressed. This is because the size of each word in the dictionary we build is fixed and is equal to n. So for example, given that the number of different symbols in the text file at hand is m and that n is 2, the total number of entries in the dictionary that we propose to build will be m*m in the worst case.[8]

8. Inclusion of Unicode Standard seamless characters to expand Arabic text steganography for secure individual uses: The need to protect confidential data stored on personal computers (PCs) or sent to other parties has increased significantly. Therefore, a secure, high-capacity strategy is needed to cover the data contained within media and make it difficult to detect. Due to the limited number of research studies that have achieved these ends through text steganography, we present an innovative approach for hiding secret bits in Arabic text with Unicode Standard seamless. Our method uses the contextual forms of Arabic characters to hide certain secret bits. Extra characters such as Zero-Width Joiners (ZWJ), Kashida, Medium Mathematical Spaces (MMSPs), and Zero-Width Non-Joiners (ZWNJ) are also used to further enhance the method's capacity without diminishing the data's security. Our experimental results show that this technique outperforms similar methods with an average improvement in capacity of more than 50%. This technique also has a higher security ratio than the other methods in this study, with the exception of Method 5, and is robust against electronic text modifications such as copying, pasting, and text formatting. Moreover, our algorithm can be widely adopted in related languages, such as Urdu and Farsi, due to its utilization of Unicode characters, which is the encoding standard used in most of the world's writing systems [2].

9. An improved algorithm for information hiding based on features of Arabic text: A Unicode approach: Steganography means how to hide secret information in a cover media, so that other individuals fail to realize their existence. Due to the lack of data redundancy in the text file in comparison with other carrier files, text steganography is a difficult problem to solve. In this paper, we proposed a new promised steganographic algorithm for Arabic text based on features of Arabic text. The focus is on more secure algorithm and high capacity of the carrier. Our extensive experiments using the proposed algorithm resulted in a high capacity of the carrier media. The embedding capacity rate ratio of the proposed algorithm is high. In addition, our algorithm can resist traditional attacking methods since it makes the changes in carrier text as minimum as possible. 2014 Production and hosting

by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University [12].

10. A Survey of Arabic Text Representation and Classification Methods: In this paper we have presented a brief current state of the Art for Arabic text representation and classification methods. First we describe some algorithms applied to classification on Arabic text. Secondly, we cite all major works when comparing classification algorithms applied on Arabic text, after this, we mention some authors who proposing new classification methods and finally we investigate the impact of preprocessing on Arabic TC[13].

1.8 Objectives

Finding an appropriate way to represent and form Arabic letters by returning to the basis of the Arabic language and its letters written in the past before diacritics, to gin standard results to serve the field of application of computer technologies and contribute to reducing the volume of Arab information stored and transferred as a prelude to trying to develop a correct treatment of the Arabic language on a computerized basis.

1.9 Thesis Outlines

This thesis is organized as follows:

1. **Chapter 1**: In this chapter, we discussed the following topics: Introduction, a brief history, Arabic standard specifications, and the proposed represent for Arabic diacritics language, problem definition, aim of thesis, Contributions, objectives.
2. **Chapter 2:** this chapter will contain the following sections: software engineering, goals of software engineering, software performance, process methods and tools, software process models, the linear sequential model, requirements engineering, the qualities of a specification, the requirements specification, The structure of a specification, functional requirements, data requirements, performance requirements, software reuse, encoding, character encoding, character rundown (theoretical characters gathering), character repertoire, Unicode encoding model, character sets character maps and code

pages, common character encodings, marketing management, marketing strategy, implementation planning, project process and vendor management, reporting measurement feedback and control systems, international marketing management, promotion strategy, benefit or profit, sales, market share (piece of the pie).

3. **Chapter 3:** the following section is depicted in this chapter: historical event for Arabic letters, representation of Arabic letters, representation of the diacritics characters, Arabic lettering program, representation of Arabic numbers and symbols, representation of Latin characters, change code, the general structure of the coding algorithm, compression ratio, marketing the new Arabic coding system.
4. **Chapter 4**: Conclusions and Recommendations.

2.THEORETİCAL BACKGROUND

2.1 Software Engineering

Although the creators have created individual meanings of programming designing, the definition proposed by Fritz Bauer at the principle gathering on this point despite everything fills in as a reason for conversation:

[Software Engineering] is the foundation and utilization of sound building standards to get dependable financial projects that work effectively on genuine equipment.

Pretty much every peruse will be enticed to add to this definition. It says small regarding the specialized parts of programming quality. It doesn't legitimately deliver the need to fulfill clients or convey the item in an opportune way; it excludes referencing the significance of estimation and measurements; it doesn't refer to the significance of a developed procedure. Nonetheless, Force's definition furnishes us with an essential line. What are the "standards of sound building" that could shape program advancement? How would we assemble programming "financially" to be "solid"? What is required to apply PC programs that work "productively" on one gadget yet on a wide range of "genuine machines"? These are the issues that keep on testing programming engineers. The IEEE has developed a dynamically exhaustive definition when it states:

Programming Building: (1) the utilization of a methodical, trained, quantifiable way to deal with the turn of events, activity, and upkeep of programming; that is, the use of designing to programming. (2) The investigation of approaches as in (1).

Programming building is tied in with making enormous bits of programming that comprise of thousands of lines of code and include numerous long periods of human exertion. One of the attractions of programming building is that there is nobody better approach to do this, but since it is so a wide range of approaches. The product engineer has to know various strategies and apparatuses. This assorted variety is a gem of programming building, and is delighted in by offering a scope of current advances and apparatuses. We will see that some product building strategies are very

much characterized while others are not all around characterized. Programming advancement forms are consistently being talked about

Software engineering with imagination and creativity is the process of creating something concrete from nothing. Software engineering methods have not yet been fully analyzed and organized. There still gets much room for imagination and creativity. Skill and taste is one of the joys of software engineering [14].

2.2 Goals of Software Engineering

Programming building is about the strategies, devices, and procedures required for programming improvement. He is especially inspired by the purposes behind the presence of a field of study called Programming Building, and the issues experienced in programming advancement. It additionally delineates an assortment of methods that endeavor to take care of issues and accomplish programming building objectives. Projects wherever encompass us in industrialized nations in home machines, correspondence and transportation frameworks, and business frameworks. The program comes in various shapes and sizes from the program on the cell phone to the program to structure another vehicle. In the order of programming, we can recognize two significant sorts:

1. System programming is a program that goes about as an apparatus to help make or bolster application programs. Models incorporate working frameworks, databases; arrange programming, and constructing agents.
2. System programming is a program that goes about as an instrument to help make or bolster application programs. Models incorporate working frameworks, databases; organize programming, and constructing agents.

Inside the classification of utilization programs, it might be valuable to characterize the accompanying system classes:

1. Games
2. Information frameworks: frameworks that store and access a lot of information, for instance, an aircraft seat reservation framework
3. Real-time frameworks: in which the PC must react rapidly, for instance, the control programming for a force station

4. Embedded frameworks: in which the PC assumes a smallish job inside a bigger framework, for instance, the product in a phone trade or a cell phone. Implanted frameworks are normally likewise constant frameworks
5. office programming: word processors, spreadsheets, email
6. Logical programming: completing figuring's, demonstrating, conjecture, for instance, climate estimating. Programming can either be off the rack (for example Microsoft Word) or customized for a particular application (for example programming for the Apollo moon shots).

The last is at times called a nitty-gritty program. These sorts of projects are conceivably data frameworks inside the purview of programming building. Data frameworks have an alternate history, and by and large, various advances are utilized to create them. The idea of the information (data) is frequently used to direct the structure of the program, so information investigation is a significant advance, which prompts the database plan of the application. This way to deal with programming improvement is past the extent of this book. Making programs is a troublesome undertaking, since programs are intricate. The recognize issues in programming improvement and the objectives that product advancement looks to accomplish are:

1. Meeting clients' needs
2. Low cost of creation
3. Superior
4. Versatility
5. Minimal effort of support
6. High unwavering quality
7. Conveyance on schedule.

Every objective is likewise accepted to be an issue since programming building was commonly ineffective. Presently we'll take a gander at every one of these objectives thusly. Later on, we'll take a gander at how objectives identify with one another. The improvement of explicit kinds of projects requires the utilization of exceptional advances; however, numerous improvement advances have a general pertinence [15].

2.3 Software performance

This is some of the time called effectiveness. These terms return to the days when the equipment cost and speed implied that each exertion was made to equipment for the most part memory and processor as precisely as could be expected under the circumstances. In the most recent week, a social change happened because of the speeding up PCs and the ease. At present, there is more spotlights on meeting individuals' prerequisites; we won't invest a lot of energy in execution. The presentation, however, can't frequently be disregarded, and we're worried about that:

1. An intuitive framework reacts inside a sensibly brief timeframe
2. A control signal is a yield to a plant in adequate time
3. A game runs adequately quick that the movement seems smooth
4. A cluster work isn't taking 12 hours when it should take one [16].

2.4 Process, Methods, and Tools

Programming designing is a layered innovation. Notice to Figure 2.1, any designing methodology (counting programming building) ought to be founded on an administrative pledge to quality. All out quality administration and comparable ways of thinking foster a culture of consistent procedure improvement, and this culture eventually prompts the advancement of progressively increasingly develop ways to deal with programming building.

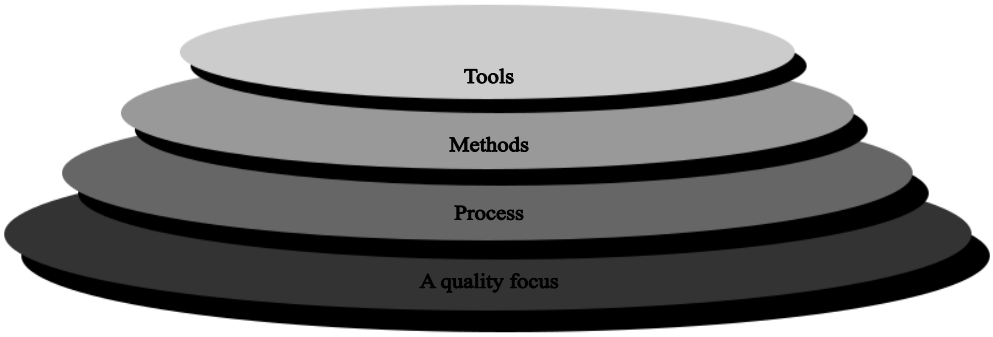

Figure 2.1: Software Engineering Layers

The reason for programming designing is centered on quality. The premise of programming building is the procedure layer. The product building process is the paste that associates the layers of innovation and permits the advancement of PC programs in a judicious and ideal way. The procedure characterizes a structure for a

lot of key procedure regions (KPAs) that must be set up for the compelling conveyance of programming building innovation. The fundamental regions of the procedure structure the reason for managerial control of program extends and decide the setting where specialized strategies, work items (structures, archives, forms, reports, models, and so on.) are applied, setting achievements, quality confirmation, and right change for men.

Programming designing strategies give specialized innovation to building programs. The techniques include a wide scope of assignments including prerequisites investigation, plan, program building, testing, and backing. Programming building strategies depend on a lot of essential rules that oversee each field of innovation utilizing displaying exercises and other illustrative methods.

Programming building devices give programmed or self-loader support for the procedure and techniques. At the point when apparatuses are consolidated with the goal that data utilized by an instrument can be utilized by another device, a framework for supporting programming advancement, called PC helped programming designing, is made. CASE joins programming and database designing programming (a vault containing significant data about the investigation, structure, program building, and testing) to ensure a product situation like computer-aided design/CAE (PC Supported Plan/Designing) equipment [17].

2.5 Software Process Models

To take care of genuine issues in the assembling condition, a product designer or gathering of architects must incorporate a significant application plan that incorporates layers of strategies, techniques, devices, and general stages that are frequently alluded to as this procedure model or programming building model. A procedure model for programming building is picked dependent on the idea of the venture and application, the strategies and apparatuses to be utilized, and the controls and conveyances required. In a charming paper on the idea of the product procedure, L.B.S S. Raccoon utilizes fractals as a premise speaking to the genuine idea of the production procedure.

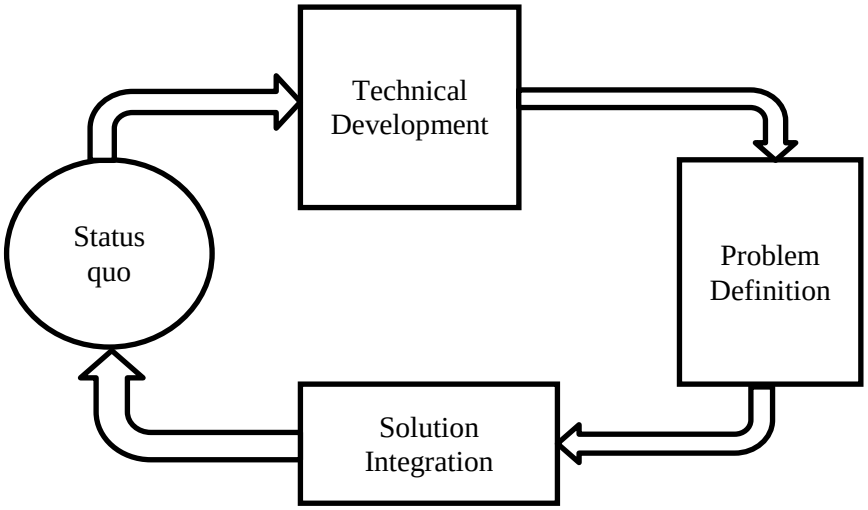

Figure 2.2: (a) The Phases of a Problem-Solving Loop

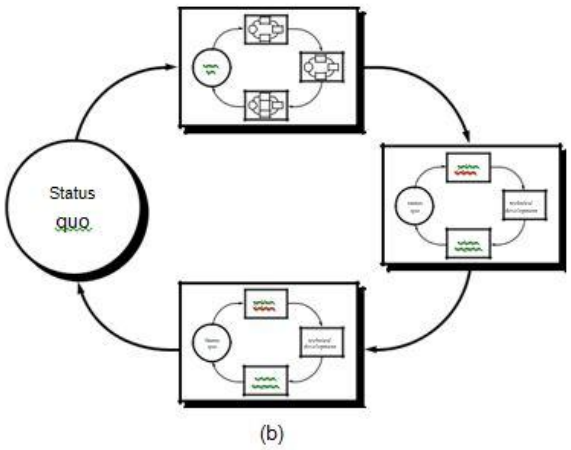

Figure 2.3: (b) The Phases Within Phases of The Problem-Solving Loop

All program improvement procedures can be depicted as a critical thinking ring (Figure 2.3a) where four unmistakable stages are experienced: the present circumstance, the issue, specialized turn of events, and incorporation of arrangements. The present state "speaks to the present state" . the meaning of the issue decides the difficult that decides its answer; specialized improvement takes care of the issue by applying a few advances, joining of arrangements, results (for instance, reports, programs, new business capacities, and new item) to the individuals who mentioned the arrangement in Fundamentally. The general advances and steps of general programming building are effectively good with these stages.

This critical thinking ring applies to programming designing work at various degrees of precision. It tends to be utilized at the total level when the application is thought of

and in the middle of the road level when the program segments are structured, even on the image level line. Along these lines, a fractal portrayal can be utilized to give an ideal perspective on the procedure. In Figure 2.3b, each phase in the critical thinking circle contains an indistinguishable critical thinking circle, which despite everything contains another critical thinking circle (this precedes to some reasonable cutoff points; for programs, a line of code).

All things considered, it is hard to compartmentalize exercises as flawlessly as Figure 2.3b suggests because cross-talk happens inside and across stages. However, this improved view prompts a significant thought: Paying little heed to the procedure model picked for a programming venture, all phases of the norm, issue definition, specialized turn of events, and arrangement joining all the while coincide at a given degree of detail. Given the iterative idea of Figure 2.3b, the four phases examined are similarly appropriate to finish the application examination and to the production of a little bit of code.

Raccoon recommends a "Tumult model" that portrays "programming advancement as a continuum from the client to the engineer to the innovation." As work advances toward a total framework, the stages are applied recursively to client needs and the designer's specialized determination of the product.

In the accompanying segments, a wide range of procedure models of programming designing are talked about. Each speaks to an endeavor to provide a request to a movement that is chaotic. Remember that each model is depicted in a manner that (in a perfect world) assists with controlling and arrange a genuine programming venture. But then, at their center, the entirety of the models displays qualities of the Bedlam model [17].

2.6 The Linear Sequential Model

The straight sequential model is some of the time called the exemplary lifecycle or cascade model, and a precise chain of command way to deal with program improvement starts at the framework level and advances through examination, plan, coding, testing, and backing. Figure 2.4 shows the straight sequential model of programming designing. Like the conventional designing course, the straight sequential model incorporates the accompanying exercises:

2.6.1 System/information engineering and modeling

Since programming is in every case some portion of a bigger framework (or business), work starts by setting up necessities for all framework components and afterward assigning some subset of these prerequisites to programming. This framework see is fundamental when the product ought to interface with different components, for example, gadgets, individuals, and databases. Frameworks designing and examination incorporate the assortment of necessities at the framework level with a modest quantity of the more significant level [18]. Structure and investigation. Data building incorporates gathering necessities at the vital business level and the business area level [18].

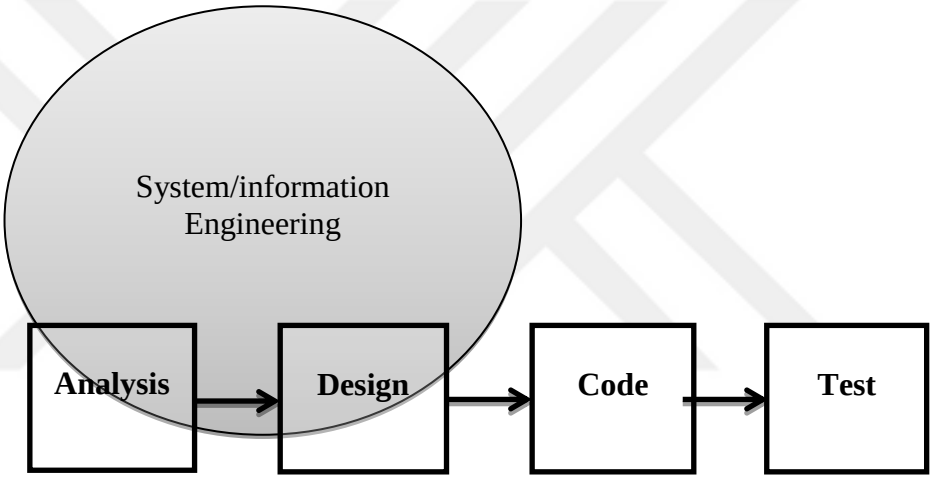

Figure 2.4: The Linear Sequential Model

2.6.2 Software requirements analysis

The way toward gathering necessities has been heightened and centers explicitly on programming. To comprehend the idea of the program (s) to be fabricated, the product engineer (the "analyst") must comprehend the program's data field, just as the usefulness, conduct, execution, and interface required. Framework and programming necessities are archived and assessed with the client [19].

2.6.3 Design

Programming configuration is a multi-step process that centers around four particular program highlights: information structure, programming design, interface portrayals, and procedural (calculation) subtleties. The plan procedure interprets the necessities

into a portrayal of the program that can be assessed for quality before coding starts. Like necessities, the plan is reported and turns out to be a piece of the product's synthesis [17].

2.6.4 Code creation

The design must be translated into a machine-readable form. The code creation step performs this task. If the design is implemented in a detailed way, the creation of a mechanical code can be accomplished [17].

2.6.5 Testing

When code has been created, program testing starts. The testing procedure centers on the intelligent inward pieces of the program, guaranteeing that all expressions are tried, and on utilitarian outer variables; for example performing tests to recognize blunders and guaranteeing that the predetermined sources of info will create real outcomes steady with the ideal outcomes [17].

2.6.6 Support

The program will without a doubt be liable to change after conveyance to the client (the conceivable special case is the product included). The change will occur due to a coincidence of errors, since the program must be adjusted to suit changes in its outside condition (for instance, the change required because of another working framework or new station), or on the grounds that the client needs useful or execution enhancements. Program support/upkeep restores every one of the past stages to a current program rather than another one.

The straight consecutive model is the most settled and for the most part used in programming structuring. Be that as it may, analysis of the model has made dynamic supporters question its viability. Among the issues that are now and then experienced while applying the straight sequential model:

Genuine tasks once in a while follow the recommended chain stream. Although the direct model can suit redundancy, it does so in a roundabout way. Therefore, the progressions can befuddle as the task group advances.

It is regularly hard for a client to expressly express all prerequisites. This requires a direct sequential model and thinks that it’s hard to retain the regular vulnerability found toward the beginning of numerous ventures. The client must have patience. A working copy of the program (s) will not be available until late in the project period. A fatal error, if not discovered until the work program is revised, can be disastrous.

In an intriguing investigation of the real undertakings Bardic [20] ,it was discovered that the straight idea of the great life cycle prompts "bans" as some venture colleagues need to sit tight for different individuals from the group to finish the needy assignments. The time invested holding up can surpass the energy spent in profitable work! Blocking will in general be increasingly predominant toward the start and end of a direct sequential procedure [19].

Every one of these issues is genuine. In any case, the great lifecycle model has a particular and significant spot in programming designing. It gives a model by which to create strategies for examination, plan, coding, testing, and backing. The great lifecycle stays a broadly utilized procedural model for programming building. While it has shortcomings, it is far superior to an arbitrary way to deal with programming advancement [19].

2.7 Requirements Engineering

Consistently, the main phase of programming improvement is to decisively characterize what the framework's clients need. The prevailing piece of this stage is correspondence between clients, designers, or specialists. At the point when specialists manage prerequisites, they are frequently alluded to as framework experts or essentially "investigators". This is the term we will utilize. As to, they are in some cases known as clients or customers. We will utilize the expression "client". The story starts when the client has a thought of another (or improved) framework. The investigator at that point contacts and collaborates to decide necessities particulars. The underlying thought of a client might be exceptionally ambiguous and uncertain, yet it is clear and once in a while explicit. It very well may be said that characterizing prerequisites is the absolute most significant movement in programming advancement. For the most part, it expends 33% of undertaking improvement endeavors. On the off chance that we can't decide precisely what is required, at that point it is futile to actualize it. In actuality, we can execute the loveliest projects on

the planet, yet on the off chance that not what was required, we fizzled. Regarding genuine programming advancement, there are solid signs that numerous frameworks don't explicitly address the issues of their clients because the requirements were not decisively distinguished in any case.

Deciding the product framework prerequisites is the initial phase in attempting to guarantee that the framework does what potential clients need. This undertaking proceeds all through the improvement of the program and is called check. Prerequisites particulars have a second essential job. This is the benchmark for assessing whether a program is working appropriately and blunder free. The mission to endeavor to guarantee programs are liberated from mistakes is a tedious and troublesome procedure all through the improvement procedure. It is called confirmation. Particulars blunders can contribute incredibly to testing and upkeep costs. The expense of fixing a blunder during the test can be multiple times the expense of fixing it during the particulars. It is evaluated that something like half of the imperfection is brought about by poor particulars. Treatment is to identify (or forestall) the imperfection early, that is, during the particular.

Simple to compose terrible program details; hard to compose great particulars. In this section, we will read the rules for composing determinations. Recollect that details are typically not composed once and afterward solidified. All the more normally, necessities change during program advancement as clients' prerequisites are explained and adjusted [17].

2.8 The Qualities of a Specification

We have seen that, in a perfect world, the details ought to be restricted to what is required. We presently offer a rundown of attractive particulars for the determinations. Great determinations ought to explain the following accompanying attributes:

- Implementation free – what is required, not how this is accomplished
- Complete – there is nothing missing
- Consistent – no individual necessity negates some other
- Unambiguous – every prerequisite has a solitary translation

- Concise – every prerequisite is expressed once just, without duplication
- Minimal – there are no pointless fixings
- Understandable – by both the customers and the designers
- Attainable – the prerequisite is doable
- Testable – it tends to be shown that the necessities have been met. This rundown of alluring highlights can be utilized as an agenda when a detail is drawn up. Moreover, it very well may be utilized as an agenda to look at and improve a current particular.

Prerequisites details ought to likewise have the option to give clear direction on the best way to check that the framework addresses the issues of its clients. In the previously mentioned train details, there is a great deal of quantitative data that permits target judgment of the achievement of the train utilizing estimation instruments, for example, stop hours. We will concentrate on some normal deficiencies in specifications. We have seen that the specifications of the tractor have the following positive properties:

1. It specifies requirements, not implementation
2. It is testable
3. It is clear.

Be that as it may, the details experience the ill effects of in any event one lack: they are inadequate. For instance, there is no notice of the expense or cutoff time. Presently we should investigate the prerequisites detail for a little segment of the program: Compose a Java applet to give an individual telephone directory. Ought to perform employments to scan for a number and enter another telephone number. The program ought to give a cordial UI. Apply the agenda above. On the execution issue, the detail says that the program will be written in Java, which is identified with the "how" of usage. Second, the determinations don't give any insights regarding the two employment subtleties; they are inadequate. The necessity is regularly hazy or subject to elective translations, and this obviously might be because of the utilization of common language in detail. Equivocalness is a typical issue. The necessities for giving a simple to-utilize interface are consequently pitifully ambiguous, making the determinations deficient and untestable. A few words are unclear and in this manner

ought to be maintained a strategic distance from inside the determinations. Some common models are the words "adaptable", "shortcoming open-minded", "quick", "proper" and "simple to utilize". In some cases the necessities strife with one another, as in these two: the information is put away on an attractive stripe the framework will react in under one second.

Since the attractive strip can't give one-second reaction time. It very well may be hard to decide oversights or inadequacy.

A run of the mill spec field is the way mistakes are dealt with, for instance, framework client input blunders. All in all, making fruitful specs is a difficult movement that necessities reliable discernment. Needs powerful correspondence between the customer and the engineer. It needs the most precise utilization of the characteristic language. A survey of determinations by a few people can help improve them [21]

2.9 The Requirements Specification

The last item for evoking prerequisites and breaking down them is the necessities detail. It is a crucial bit of documentation that is basic to the achievement of any product advancement venture. If we can't decide precisely what the framework ought to do, how might we build up the program with certainty, and how would we want to ensure that the finished result addresses its issues? It is the reference report by which every resulting improvement is assessed. There are three significant variables to consider:

1. The degree of detail.
2. To whom the record is tended to.
3. The documentation utilized.

The primary factor identifies with the need to limit determinations however much as could be expected to figure out what the framework ought to do as opposed to how it ought to be finished. As we've seen, the particulars ought to in a perfect world be the clients' perspective on the framework as opposed to anything about how the framework is executed. The subsequent factor emerges because the particulars must be comprehended by two unique gatherings of clients - clients and designers. The individuals in these two gatherings have various foundations, encounters, and phrasing. They share a shared objective

Depict what the system should do, yet they will as a rule use a substitute language. Customers will tend non-specific depictions conveyed in the trademark language. Amazingly, while the trademark language is unimaginable for refrain and love messages, it is a horrible technique to achieve definite, dependable, and unambiguous points of interest. On the other hand, given their specific bearing, inspectors may need to use precise documentation (possibly numerical) documentation to choose the structure. This carries us to the topic of the documentation. A few documentations are accessible for composing details:

- Informal, writing in characteristic language, utilized as unmistakably and cautiously as could be expected under the circumstances. In this part, we will focus on this methodology.
- Formal, utilizing scientific documentation, with thoroughness and brevity. This methodology is outside the extent of this book. Formal techniques will in general be utilized uniquely in security basic frameworks.
- Semiformal, utilizing a blend of regular language along with different diagrammatic and plain documentations.

Most of these symbols have their origins in software design methods, that is, in software implementation methods. Hence a potential problem is to include information about implementation. These images will be talked about later in this book and incorporate false images, information stream graphs, and class charts. Right now, most prerequisites details are written in characteristic language, helped by use case graphs. One methodology is to draw up two reports:

1. A prerequisites determination composed fundamentally for clients, depicting the clients' perspective on the framework and communicated in normal language. This is the substance of the agreement between the clients and the engineers.
2. A specialized particular that is utilized principally by designers, communicated in some increasingly formal documentation, and portraying just a piece of the data in the full necessities detail. On the off chance that this methodology is received, there is then the issue of guaranteeing that the two archives are perfect [17].

2.10 The Structure of a Specification

Given that necessities particular will normally be written in common language, it is valuable to design the general structure of the determination and to recognize its segment parts. We can likewise distinguish those fixings that, maybe, ought not to be incorporated by any stretch of the imagination, since they are worried about the usage instead of the prerequisite. The rest of this segment presents one method of organizing details. One way to deal with giving an unmistakable structure to a determination is to parcel it into parts. Programming comprises of the blend of information and activities. In determinations, the comparing components are called useful and information prerequisites. One of the significant discussions in processing is about which of these two fundamental components – information or capacity – is essential. A few ways to deal with advancement, outstandingly the article situated methodology, are all-encompassing, rewarding capacity and information with equivalent significance. Be that as it may, our anxiety here is with the determination, not with advancement draws near. Be that as it may, the organization of a particular will in general mirror the framework improvement strategy being utilized [17]. A checklist for the contents of a requirements specification is:

1. The utilitarian necessities
2. The information necessities
3. Performance necessities
4. Constraints
5. Guidelines.

We will currently take a gander at these thus.

2.11 Functional requirements

The utilitarian prerequisites are the genuine substance of a necessities determination. They state what the framework ought to do. Models are: The framework will show the titles of the considerable number of books composed by the predetermined writer. The framework will ceaselessly show the temperatures of the considerable number of machines. Practical necessities are described by action words that perform activities [22].

2.12 Data requirements

Information prerequisites have three parts:

1. User's information is a contribution to or yield from the framework using a screen, console, or mouse.
2. Data that is put away inside the framework, as a rule in documents on the plate, for instance, data about the books held in an open library.
3. Information went to or from another PC framework, for instance, to a server. [23].

2.13 Collect System Requirements

This stage is the most important stage in the development process, and this stage begins with identifying the users and those involved in the production of the program and identifying their requirements accurately and clearly. There are several techniques and methods that help in the process of gathering requirements such as:

1. Interview: This method is used with the most important people involved in the production of the program, especially if the number of people involved in the production of the program is large, and when the requirements gathering team wants specific answers.
2. Workshop technique: In this way, the requirements gathering team gathers all those involved in the production of the program and all the people whose work the program may affect in one place in order to listen to all opinions. It is important in this technique that the workshop is managed in a way that guarantees listening to everyone and also guarantees not losing control of the workshop.
3. Brainstorming technique: This technique is used on the work team and also on the users. In this technique, ideas are proposed about the program, taking into account that everyone has the right to suggest their ideas even if they are strange.
4. The use of prototypes and experimental designs: This technique is used by the work team to inform the customer of work findings and take his opinions, and it is also used to obtain approval from the customer that these

requirements are correct. This technique also helps users to express their requirements clearly and concretely. And this technology continues to be used in many stages of software development.

5. Questionnaires technique: This method is used when some of the persons involved in the production of the program are in another place by sending them questionnaires or talking to them by phone. This method is used to question specific things.

2.14 Performance Requirements

These are proportions of execution, some of which are quantitative, and some of which can be utilized as a major aspect of testing[17]. Models are:

1. Cost
2. Delivery date
3. Response occasions (for example the framework will react to client demands inside one second.)
4. Data volumes (for example the framework must have the option to store data on 10,000 workers.)
5. Loading levels to be adapted to (for example the framework must have the option to adapt to 100 exchanges for each moment from the retail location terminals).
6. Reliability prerequisites (for example the framework must make some mean memories between disappointments of a half year.)
7. Security necessities.

2.15 Software Reuse

A significant programming building method is to reuse programming parts from a library or a previous venture. This abstains from rehashing an already solved problem, and can spare tremendous exertion. Moreover, reusable segments are normally altogether tried. It has for some time been a fantasy of programming specialists to choose and utilize helpful segments, similarly as an electronic designer counsels a list and chooses instant, attempted and-tried electronic parts. Be that as it

may, a part can only with significant effort be reused if it is associated in some unpredictable manner to different segments in a current framework. A heart transplant starting with one person then onto the next would be incomprehensible if there were such a large number of corridors, veins, and nerves to be cut off and reconnected.

There are in these manner three necessities for a reusable segment [24]:

1. It offers valuable assistance.
2. It plays out a solitary capacity.
3. It has the base of associations (in a perfect world no associations) with other components.

2.16 Encoding

Encoding is the way toward changing over information into an organization required for a few data preparing needs, including:

- Program assembling and execution
- Data transmission, stockpiling, and pressure/decompression
- Application information preparing, for example, record change

Encoding can have two implications:

- In PC innovation, encoding is the way toward applying a particular code, for example, letters, images, and numbers, to information for transformation into a comparable figure.
- In gadgets, encoding alludes to simple to computerized transformation.

Encoding incorporates the usage of a code to change remarkable data into a structure that can be used by an outside technique. [25]

The sort of code used for changing over characters is known as the American Standard Code for Data Trade (ASCII), the most consistently used encoding plan for records that contain content. ASCII contains printable and nonprintable characters that address promoted and lowercase letters, pictures, diacritics checks, and numbers. An unprecedented number is given out to specific characters [26].

The standard ASCII plot has only zero to 127 character positions; 128 through 255 are dubious. The issue of indistinct characters is comprehended by Unicode

encoding, which permits some of each character used the world over. Various types of codes join BinHex, Uuencode (UNIX to UNIX encoding), and Multipurpose WebMail Expansions (Emulate).

Encoding is also used to diminish the size of sound and video records. Each solid and video record position has a contrasting coder-decoder (codec) program that is used to code it into the fitting arrangement and a short time later deciphers for playback.

Encoding should not be confused with encryption, which disguises content. The two techniques are used comprehensively in the frameworks organization, programming, remote correspondence, and limit fields.[27]

American Standard for Data Exchange (ASCII) is a methodology for encoding characters that rely upon the solicitation for alphabetic characters in the English language.

ASCII entire number depictions have printable and nonprintable subsets. Printable characters are common characters, and nonprintable characters can't avoid being characters used to address comfort keys, e.g., erase, eradicate, and return. [28]

ASCII is 7-piece addressing only 128 characters (0-127). The range 0-31 are control characters, with 32-127 addressing successive characters from start to finish, numerals from 0 to 9, and diacritics marks (anyway not in a particular request). ASCII could very well be used to encode U.S. English.

A couple of individuals dumbfound codes more than 128-255 to be ASCII, yet really, they are most certainly not. As PCs propelled, it got normal to use an 8-piece byte. This last character thought about extra 128 characters, which is known as widened ASCII. Different systems complete out of nowhere widened ASCII, so there are likenesses gives that aren't knowledgeable about the underlying 128 characters [28].

2.17 Character Encoding

Character encoding is utilized to speak to a collection of characters by an encoding framework. Contingent upon the deliberation level and setting, relating code focuses and the subsequent code space might be viewed as bit designs, octets, regular numbers, electrical heartbeats, and so on. A character encoding is utilized in the calculation, information stockpiling, and transmission of literary information.

"Character set", "character map", "codeset" and "code page" are connected, yet not indistinguishable, terms [29].

Early character codes related to the optical or electrical message could just speak to a subset of the characters utilized in composed dialects, once in a while confined to capitalized letters, numerals, and some accentuation as it were. The minimal effort of the computerized portrayal of information in present-day PC frameworks permits progressively expand character codes, (for example, Unicode) which speak to the greater part of the characters utilized in many composed dialects. Character encoding utilizing globally acknowledged gauges licenses the overall exchange of text in electronic structure. Terms related to character encoding [27]:

- The character is the base unit of a substance that has a semantic worth.
- The character set is a great deal of characters that can be used by different tongues. Model: The Latin character set is used by English and most European lingos, yet simply the Greek character set is used by the Greek language.
- The coded character set is a great deal of characters wherein each letter facilitates an exceptional number.
- The code point for an encoded character set is any permitted a motivation in the character set or code space.
- Code partition is a ton of numbers whose code centers are code regards.
- Code unit is a piece course of action that is used to encode each letter of a reference as a specific encoding [27].

2.18 Character Rundown (Theoretical Characters Gathering)

The character collection is a theoretical arrangement of more than one million characters found in a wide assortment of contents including Latin, Cyrillic, Chinese, Korean, Japanese, Hebrew, and Aramaic.

Different images, for example, melodic documentation are likewise remembered for the character collection. Both the Unicode and GB18030 measures have a character collection. As new characters are added to one norm, the other standard additionally includes those characters, to look after equality.

The code unit size is proportional to the bit estimation for the specific encoding:

- The US-ASCII code unit is 7 bits.
- The UTF-8, EBCDIC, and GB18030 code unit is 8 bits.
- The UTF-16 code unit consists of 16 bits.
- The UTF-32 code unit consists of 32 bits [30].

2.19 Unicode Encoding Model

Unicode and its equivalent standard, the ISO/IEC 10646 Widespread Character Set, together involve a forefront, bound together character encoding. Instead of mapping characters genuinely to octets (bytes), they autonomously portray what characters are available, contrasting ordinary numbers (code centers), how are these numbers encoded as a movement of customary amounts of fixed size (code units), in conclusion how are these units coded as a movement of octets [31]. The inspiration driving this examination is to cause an overall plan of characters that can be encoded in a collection of ways. Adequately delineating this structure requires more careful terms than "character set" and "character encoding" [4]. It follows the terms used in the propelled model:

2.19.1 Character reference

A character reference it is a full arrangement of theoretical characters upheld by the framework. The reference might be shut, for example, augmentations are not permitted without making another norm (likewise with ASCII and a large portion of the ISO-8859 arrangement), or it might be open, permitting increases (as with Unicode and to a constrained degree Windows code pages. The characters in a specific reference mirror the choices made about how to partition composing frameworks into fundamental data units. The essential factors of the Latin, Greek, and Cyrillic letter set can be separated into letters, numbers, diacritics imprints, and some uncommon characters, for example, zone, which would all be able to be masterminded in a straightforward direct grouping that is shown in a similar request. Peruse. Yet, even with these letters in order, diacritics structure a multifaceted nature: they can be considered either as a major aspect of a solitary character that contains a letter and diacritics (known as the reconfigured character), or as

independent letters. The first takes into account a lot less complex book control framework, yet the last permits the utilization of any mix of letters/emphasizes in the content. Ligatures cause comparable issues. Other composing frameworks, for example, Arabic and Hebrew, are spoken to by increasingly complex character sets because of the need to suit such things as bi-directional content and images that are connected in various manners to various circumstances [25].

2.19.2 Coded character set (CCS)

A coded character set (CCS) a limit that gives out characters to code centers (each picture point addresses one letter). For example, in a particular reference, the promoted "An" in the Latin letter set can be addressed by the picture point 65, the letter "B" to 66, and so forth. Different coded character sets may have a comparable set; for example, ISO/IEC 8859-1 and IBM 037 and 500 code pages all spread a comparative reference yet set it to different code centers. [25].

2.19.3 Character encoding form (CEF)

A character encoding structure (CEF) The assignment of code focuses on code units to encourage capacity in a framework that speaks to numbers as fixed-length bit arrangements (for example essentially any PC framework). For instance, a framework that stores numeric data in 16 bits can straightforwardly speak to 0 to 65535 code focuses per unit, however bigger code focuses (for instance, 65536 to 1.4 million) can be spoken to utilizing various 16 bits. This correspondence is characterized by CEF [25]. Next one,

2.19.4 Character encoding scheme (CES)

A character encoding plan (CES) is the interconnection of code units in an arrangement of octets to encourage capacity on an octal record framework or transmission over an octal system. Basic character encoding plans incorporate UTF-8, UTF-16BE, UTF-32BE, UTF-16LE or UTF-32LE; Compound character encoding plans, for example, UTF-16, UTF-32 and ISO/IEC 2022 switch between numerous outlines Basic utilizing byte request labels or departure arrangements; pressure frameworks attempt to decrease the number of bytes utilized per code unit, (for example, SCSU, BOCU, and Punycode) [25].

Even though UTF-32BE is the easiest CES, most frameworks that work with Unicode use either UTF-8, which is in reverse perfect with fixed-width ASCII and doles out Unicode code focuses to variable-width successions of octets or UTF-16BE, It is in reverse good with fixed-width UCS-2BE and maps Unicode code maps to 16-piece variable-width arrangements. See a correlation of Unicode encodings for a point by point conversation [25].

At last, there may be a more elevated level convention that gives extra data to characterize the private variable for a Unicode character, particularly when there are provincial factors "normalized" in Unicode as a solitary character. A case of this is the XML: Lang trait.

The Unicode model uses the term sanction term for chronicled frameworks that allot a grouping of characters straightforwardly to the arrangement of bytes, covering all layers of CCS, CEF, and CES [25].

2.20 Character Sets, Character Map and Code Pages

Truly, the expressions "character encoding", "character map", "character set" and "code page" have been interchangeable in software engineering, where a similar standard will characterize a gathering of characters and how they will be encoded in a surge of code units - typically one letter for every unit the code. However, the phrasing presently has related yet particular implications,because of the endeavors of measures bodies to utilize exact wording when expounding on and binding together a wide range of coding frameworks. In any case, the terms are as yet utilized conversely, with the character set all over the place.

A "code page" as a rule implies a byte-to-byte encoding, yet for some arrangement of encodings (covering various contents), numerous characters in similar codes share most of these code pages. Notable code page bunches are "Windows" (given Windows-1252) and "IBM"/"DOS" (contingent upon code page 437), see Windows Code Page for subtleties. Most, yet not all, of the documentation indicated by code pages, are single-byte encodings (yet observe the 8-piece byte size). It sets the Character Information Portrayal Structure (CDRA) from IBM with encoded character set identifiers (CCSIDs) and they are each called in an unexpected way "charset", "character set", "code page" or "charmap".

The expression "code page" doesn't show up in UNIX or Linux as "charmap" is liked, and is generally in the more extensive setting of nearby dialects.

As opposed to the CCS above, "character encoding" is a guide of unique characters to code words. The character set in HTTP (and Emulate) is like character encoding (dislike CCS) [32].

"Old coding" is a term some of the time used to depict coding's of old characters, however with vagueness insignificance. The greater part of its utilization is with regards to Unicodification, where it alludes to documentations that neglect to cover all Unicode code focuses, or when all is said in done, utilize a to some degree diverse character set: a few image focuses speak to one Unicode character, or the other way around (see for instance page Images 437). A few sources allude to coding as heritage simply because it goes before Unicode. All Windows code pages are typically alluded to as obsolete, because they are Unicode-related and because they can't speak to every one of the 221 potential Unicode code focuses.

2.21 ISO 8859-6 Arabic

One of the most common character encoding is ISO/IEC 8859-6:1999, Information innovation 8-piece single-byte coded realistic character sets — Part 6: Latin/Arabic letters in order, is a piece of the ISO/IEC 8859 arrangement of ASCII-based standard character encodings, first release distributed in 1987. It is casually alluded to as Latin/Arabic. It was intended to cover Arabic. Just ostensible letters are encoded, no reshaped types of the letters, so forming handling is required for show [33]. It does exclude the additional letters expected to compose most Arabic-content dialects other than Arabic itself, (for example, Persian, Urdu, etc.). ISO-8859-6 is the IANA preferred charset name for this standard when supplemented with the C0 and C1 control codes from ISO/IEC 6429. The text is in a logical order, so BiDi processing is required for display. Nominally ISO-8859-6 (code page 28596) is for "visual order", and ISO-8859-6-I (code page 38596) is for logical order. But in practice, and required for HTML and XML documents, ISO-8859-6 also stands for logical order text. There is also ISO-8859-6-E which supposedly requires directionality to be explicitly specified with special control characters; this latter variant is in practice unused. IBM has assigned code page 1089 to ISO 8859-6 [34]. It is an emulation for their AIX operating system. Unicode is preferred over ISO-8859-6 in modern

applications, especially on the Internet; meaning the dominant UTF-8 encoding for web pages (see also Arabic script in Unicode, for complete coverage, unlike e.g. ISO-8859-6 or Windows 1256 that do not cover extras). 0.1% of all web pages use ISO-8859-6.

2.22 Code Chart [35]

Table 2.1: Code Values Are Set To 0xEB - 0xF2 to Merge Characters

ISO/IEC 8859-6																
	_0	**_1**	**_2**	**_3**	**_4**	**_5**	**_6**	**_7**	**_8**	**_9**	**_A**	**_B**	**_C**	**_D**	**_E**	**_F**
2_ 32	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	، 002C	- 002D	. 002E	/ 002F
3_ 48	0/٠ 0030	1/١ 0031	2/٢ 0032	3/٣ 0033	4/٤ 0034	5/٥ 0035	6/٦ 0036	7/٧ 0037	8/٨ 0038	9/٩ 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ 64	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ 80	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6_ 96	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ 112	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	
A_ 160	NBSP 00A0				¤ 00A4								، 060C	SHY 00AD		
B_ 176												؛ 061B				؟ 061F
C_ 192		ء 0621	آ 0622	أ 0623	ؤ 0624	إ 0625	ئ 0626	ا 0627	ب 0628	ة 0629	ت 062A	ث 062B	ج 062C	ح 062D	خ 062E	د 062F
D_ 208	ذ 0630	ر 0631	ز 0632	س 0633	ش 0634	ص 0635	ض 0636	ط 0637	ظ 0638	ع 0639	غ 063A					
E_ 224	ـ 0640	ف 0641	ق 0642	ك 0643	ل 0644	م 0645	ن 0646	ه 0647	و 0648	ى 0649	ي 064A	ً 064B	ٌ 064C	ٍ 064D	َ 064E	ُ 064F
F_ 240	ِ 0650	ّ 0651	ْ 0652													

2.23 Windows-1256

Windows-1256 is a code page used to make Arabic (and conceivably some other language that uses Arabic substance, for instance, Persian and Urdu) inside Microsoft Windows. This code page isn't acceptable with ISO 8859-6 and MacArabic encodings [36].

The coding of each single letter is without the fundamental Arabic letter set, and just one out of each odd generous visual sort of the variables of the letter shape is separated, beginning, widely appealing, last, or headed (for instance coding letters, not glyphs). The Arabic letters in the C0-FF pack are in a steady progression all together sorted out in Arabic, yet some Latin letters are punctuated. These are a segment of the Latin letters Windows-1252 used in the French language, as this European language has some chronicled criticalness in the past French areas in North Africa, for instance, Morocco and Algeria [36].

This allowed French and Arabic substances to be mixed while using Windows 1256 without trading the code page (regardless, uppercase letters are avoided with diacritics) [36].

Unicode is supported on Windows 1256 in present-day applications, especially on the Web; this suggests the predominant UTF-8 encoding for website pages (see in like manner Arabic substance in Unicode, for full consideration, not at all like Windows 1256 or ISO-8859-6 that doesn't cover modules). Under 0.1% of all site pages use Windows-1256 in September 2019.

2.24 Character Set

Since the first code page left 9 characteristics (bytes) set apart as "unused" in the principal points of interest, this byte was later used to secure additional characters required for the Persian-Arabic substance (for Persian and Urdu), despite Euro sign. The going with table delineates the widely inclusive version of Windows-1256 [37]. Each character appears with Unicode indistinguishable and its decimal picture. Here each Arabic letter appears in a different structure. The genuine kinds of letters are presented inside the Arabic words through a great deal of programming rules and sponsorship for the fitting content style. [35]

Table 2.2: Windows 1256 or ISO-8859-6 That Does Not Cover Plugins [36]

Windows-1256																
	_0	**_1**	**_2**	**_3**	**_4**	**_5**	**_6**	**_7**	**_8**	**_9**	**_A**	**_B**	**_C**	**_D**	**_E**	**_F**
0_ **0**	NUL 0000	SOH 0001	STX 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
1_ **16**	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
2_ **32**	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	، 002C	- 002D	. 002E	/ 002F
3_ **48**	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4_ **64**	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5_ **80**	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6_ **96**	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7_ **112**	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
8_	€ 20AC	پ 067E	‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	ٹ 0679	‹ 2039	Œ 0152	چ 0686	ژ 0698	ڈ 0688

128	20AC	067E	201A	0192	201E	2026	2020	2021	02C6	2030	0679	2039	0152	0686	0698	0688
9_	گ	‘	’	“	”	•	–	—	ک	™	ڑ	›	œ	ZWNJ	ZWJ	ں
144	06AF	2018	2019	201C	201D	2022	2013	2014	06A9	2122	0691	203A	0153	200C	200D	06BA
A_	NBSP	،	¢	£	¤	¥	¦	§	¨	©	ھ	«	¬	SHY	®	¯
160	00A0	060C	00A2	00A3	00A4	00A5	00A6	00A7	00A8	00A9	06BE	00AB	00AC	00AD	00AE	00AF
B_	°	±	²	³	´	µ	¶	·	¸	¹	؛	»	¼	½	¾	؟
176	00B0	00B1	00B2	00B3	00B4	00B5	00B6	00B7	00B8	00B9	061B	00BB	00BC	00BD	00BE	061F
C_	ہ	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
192	06C1	0621	0622	0623	0624	0625	0626	0627	0628	0629	062A	062B	062C	062D	062E	062F
D_	ذ	ر	ز	س	ش	ص	ض	×	ط	ظ	ع	غ	ـ	ف	ق	ك
208	0630	0631	0632	0633	0634	0635	0636	00D7	0637	0638	0639	063A	0640	0641	0642	0643
E_	à	ل	â	م	ن	ه	و	ç	è	é	ê	ë	ى	ي	î	ï
224	00E0	0644	00E2	0645	0646	0647	0648	00E7	00E8	00E9	00EA	00EB	0649	064A	00EE	00EF
F_	ً	ٌ	ٍ	َ	ô	ُ	ٓ	÷	ّ	ù	ْ	û	ü	LRM	RLM	ے
240	064B	064C	064D	064E	00F4	064F	0650	00F7	0651	00F9	0652	00FB	00FC	200E	200F	06D2

2.25 Software Project Management [38]

In the software project management stage, it depends on the following two paragraphs:

- Project management:

  It is the process of planning and controlling the development of the system during a specified period of time and with the lowest possible costs

- Project manager:

  He is the person responsible for managing hundreds of tasks that need to be coordinated

2.26 System Development Stages [39]

1. Collect system requirements:

A statement that includes the required functions of the new system and the characteristics it has. Here we can visualize the interfaces of the system in order to help us visualize the next work that we must do. The requirements can be modified during the next stages.

2.27 The Requirements Have Two Types

- **Functional requirements**: include the operations that the system must perform,
- **Non-functional requirements:** properties or tasks that is not present in the system but which it needs: operational, performance, security, etc.

This stage ends with the issuance of a requirements definition report, which is a text file that contains the required requirements, within priorities, the aim of which is to issue the so-called project scope [39].

2.28 Methods for Gathering Requirements

Interviews: The most common method, its stages: selecting the interviewees, setting the interview questions, preparing for the interview, conducting the interview, employing those who grew up in the interview [39].

2.29 Planning

Here we can use project management techniques such as creating network diagrams and Gantt diagrams [39].

2.30 Analysis

Study the way the system works conceptually (high level close to human understanding)

The aim is to understand the requirements of the new system and to develop a system compatible with it. Include system description [39].

2.31 Design

Study the way the system works in a physical way [39]

It includes designing system interfaces

- Building Algorithms
- Programming

2.32 Tests

A good test includes a test for the exceptional cases and possible incorrect entries

A successful test is one by which we can find errors [39].

2.33 Maintenance [40]

System characterization methods:

2.33.1 Use case

A group of activities that produce a specific result, describing how the system interacts with the event it is related to. This concept is useful when the situation is complex [40].

2.33.2 Process model

A basic way to show how the system operates, visualizing the activities that are being performed and how the data moves between these activities

- **Data Flow diagramming (DFD)**

Is a common technique for creating process models and is essential for creating basic system process descriptions.

It has levels:

- **Context diagram:**

Represents the whole system within a single process and shows the external entities and their connection with the system

1. **Level 0:** we include operations briefly and very broadly
2. **Level 1:** we put the operations contained in the operations of the previous level (i.e. We go into more details)
3. **Level 2:** we put the operations contained in the operations of the previous level (i.e. We go into more details)

Balancing: is the process of ensuring that information displayed at one level of dfd is more accurately displayed at the next level. [40]

Note: The DFD diagram at one level must contain between 3 - 9 operations (this is in ideal cases)

- **Context diagram:**

It is the initial DFD diagram, meaning that it contains the first level, which in turn describes the system in a brief and very general way, and shows all external entities (i.e., the objects that deal with the system)

Note: The systems analyst must be proficient in DFD and USE CASE charts, as these are essential skills. [40]

External Entity: Any person, organization or system that deals externally with the system for which we want to create a DFD scheme

2.34 Program Design Document [41]

It includes diagrams that define the architecture and details programmers need to get started

1- **Data model:**

A set of fundamentals that describe the database and the restrictions that must be tracked

2- **Meta data:**

The stored information that describes the contents of the data model.

3- **Design Strategies:**

a. **In-house:**

It means that programming takes place within the organization itself

Advantages:

- It allows flexibility and creativity in system production
- Compatible with the technology and standards used
- It develops the skills of the team work in the organization

Disadvantages:

- It takes time and effort that may not exist now
- May require skills that are not currently available
- Often it costs more
- Usually it requires more time than planned
- The risk that the project will fail

b. **Purchase a ready-made system and modify it as desired:**

Can be compatible with many existing organizations and businesses

Tried and guaranteed, this saves us time and cost, rarely does it fit fully into an organization's requirements, may require specific experience

c. **Outsourcing:**

That is, to make a foreign party program the system, this can reduce or increase costs

Risk:

- We lose confidentiality of information in the organization
- We won't have the opportunity to develop the system in the future
- We will not have the opportunity to learn

3. THE PROJECT MANAGEMENT FOR A PROPOSED ARABIC CODING SCHEME

3.1 Introduction

In software engineering, building a software system is not just writing a code, but rather a production process that has several basic and necessary stages to obtain the product. The ultimate goal of any program is to obtain the best possible efficiency at the lowest possible cost. (This standard applies to any engineering design process and in all fields). These stages are called the Software Life Cycle, or sometimes known as the "Software Development Life Cycle (SDLC: Software Development Life Cycle)".

The software life cycle consists of the following stages:

1. Requirement Analysis: During this stage, the team responsible for developing the program determines the requirements and goals they want to reach through the program. Usually these requirements are specific to the customer, and here issues of cost and quality must be taken into consideration.
2. Design: In the design stage, the team develops ideas for how to design the program. Here discussion is made about what is the best programming language or usable programming environment in order to implement the required program with the specific requirements.
3. Implementation: After defining the requirements and objectives, agreeing on the programming language and establishing the basic outlines of the program and its architecture, the team implements a "prototype" of the program.
4. Testing: In the testing phase, the team tests their program and makes sure that it performs all the functions required of it, and within the established standards and requirements.
5. Evolution: After completing the testing phase, and ensuring that the program is effective, and performs the required function of it with the best efficiency

and lowest cost, the team works to search for ways to develop the program, add new features to it, and make it more effective and more efficient.

The stages of developing and building a software system, or programs, according to the classic "waterfall" model.

The previous stages are one of the most famous models for building the software system, which is the "Waterfall Model", which is considered one of the classic models in building the software system. This model and other classic models were developed, where the concept of flexible systems or processes appeared, Fragile Process, which gives up the fixed model of the classic system in order to obtain more flexibility and comfort in how and the mechanism of implementing the project and the software system.

3.2 Coding System Requirements

The proposed representation should be part of any application that deals with Arabic language texts implicitly, so the user of these applications does not have to have full knowledge of its details. However, the system analyst of this representation must have full knowledge of the systems of representation of data in general and the date of the origin of Arabic writing on the other hand. In doing so, we will need to collect and extract information in the following ways:

1- Reading several books on the subject of Arabic writing history. By reading these books we initiate an idea about the original letters of Arabic language.

2- Brainstorming technique: by this technique we proposed using the derived Arabic letters by using diacritics.

3.1.2 Requirement collecting analysis

This stage results in a set of written files that contain some tables and drawings that clearly and directly specify the requirements. Among the conditions that must be met in these files are:

1- That it contains all the requirements and that these requirements be written in a clear, direct and accurate manner that does not allow any other interpretation.

2- These files should be up-to-date and contain the date for each update, the author of these requirements and the source of these requirements.

3- It should be taken into account that the requirements gathering team is not the design team. Therefore, the requirements must be clearly and understandably written in order to enable the design team to start the design process based on these files without the help of the requirements gathering team. Therefore, any error or ambiguity in these files may result. To problems and difficulties in advanced stages, which often leads to material loss and delay in the delivery of the program.

In this thesis the designer is the one who in charge of gathering the requirements of the program and used the brainstorming and prototyping methods for gathering the program requirements, so the prototypes itself is the collected information.

3.2 Representation of Arabic letters

Before using diacritics, the Arabic alphabet contains 15 letters, four of them having single pronounce, nine having two pronounces, and 2 having three pronounces for each the total is 28 after using diacritics to use each character with a single pronunciation.

Four bits are needed to represent 16 letters (15 original letters plus the space). To accomplish this, we divided the letters into two groups: the original letters and the derived letters by adding punctuation to the original letters (and the designation based on the method of drawing the letter) as shown in Table No. 3.1.

Table 3.1: The Original and Derived Letters

Original	Derived	Original	Derived	Original	Derived
الألف		ر	ز	ف	ق
ب	ي	س	ش	ل	ك
ن	ت، ث	ص	ض	م	
ح	ج، خ	ط	ظ	هـ	
د	ذ	ع	غ	و	

From Table 3.2, it can be noted that there are letters that do not have a derivative and letters with more than one derivative such as 'هـ' and 'ن'. To make the derivative is only one for "ح" and "ن", the second derivative of "ن" is to be transferred, and the

second derivative for "ح", which is "خ" moved to the location in the table that correspond to the original letters with no derivative.

The number of entries in Table 3.2 is fifteen entries and it is quite sufficient to represent the original letters using four binaries, but there is a letter that must be included in the representation because of its presence in the text and it is a void and it needs to empty one of the entries of Table No. 3.2 i.e. transfer of one of the original letters and make it a derivative of another original letter This case is proportional to the letter "ء", which can be considered to be derived from the letter "أ", and there is the letter "ة", added as derived for the letter "هـ" and the result can be shown in Figure No. 3.2.

Table 3.2: Original and Derivative Arabic Letters

Original	Derivative	Original	Derivative	Original	Derivative
ا	ء	ر	ز	ف	ق
ب	ي	س	ش	ل	ك
ن	ت	ص	ض	م	ث
ح	ج	ط	ظ	هـ	ة
د	ذ	ع	غ	و	خ

The original letters in Table 3.3can be represented using four bits as shown in Figure 3.3.

Table 3.3: Quartet Representation Of Original Letters

Char.	Rep.	Char.	Rep.	Char.	Rep.	Char.	Rep.
ا	0000	د	0100	ط	1000	م	1100
ب	0001	ر	0101	ع	1001	هـ	1101
ن	0010	س	0110	ف	1010	و	1110
ح	0011	ص	0111	ل	1011	Space	1111

To design a correct representation for non-diacritic Arabic letters, a fifth binary bit is added with a value of (0) if the letter is original and (1) if the letter is a derivative, for

example (00011) represents the character "ح" and (10011) represents the character "ج" and (01101) represents the letter "هـ", (11101) represents the letter "ة" and so on for the rest of the letters in Table 3.4. For operational necessities, the symbol (11111) is considered the symbol for changing to ASCII representations (it will be explained later), and the space is represented by (01111).

The quintet is a sufficient representation for all Arabic letters as shown in Figure 3.5. Arabic letters contain two groups: the connected and the separated letters according to the formulation and the spelling rules that depend sometimes on the diacritics character (will be explained later), the editor program will draw the letters on the screen or on the paper with its suitable form depending on the formulation and spelling rules.

By now, non-diacritic Arabic letters are represented by five bits instead of the eight-bit representation currently in use, thus three bits are reduced.

Table 3.4: Quintet Representation of the Letters of the Arabic Language

B_0							
1	**0**		Line	B_1	B_2	B_3	B_4
ء	**ا**	16	**0**	0	0	0	0
ي	**ب**	17	**1**	0	0	0	1
ت	**ن**	18	**2**	0	0	1	0
ج	**ح**	19	**3**	0	0	1	1
ذ	**د**	20	**4**	0	1	0	0
ز	**ر**	21	**5**	0	1	0	1
ش	**س**	22	**6**	0	1	1	0
ض	**ص**	23	**7**	0	1	1	1
ظ	**ط**	24	**8**	0	0	0	0
غ	**ع**	25	**9**	1	0	0	1
ق	**ف**	26	**10**	1	0	1	0
ك	**ل**	27	**11**	1	0	1	1
ث	**م**	28	**12**	1	1	0	0
ة	**هـ**	29	**13**	1	1	0	1
خ	**و**	30	**14**	1	1	1	0
chng	**space**	31	**15**	1	1	1	1

3.3 Representation of the Diacritics Characters

To represent the diacritics in the Arabic language, we have noticed that the diacritics includes Sokon, Fatha, Dhama, Kasrah, and Tanween each of the previous, which are four diacritics in addition to the Tanween and Shaddah, and we divided the formation into two original and derived groups, given that the Tanween of each diacritic as shown in Table 3.6.

Table 3.5: The Original and Derived From Movements

Original	Diacritics shape	Derivative
Sokon	**Non-diacritic**	◌ْ
Fatha	◌َ	◌ً
Dhama	◌ُ	◌ٌ
Kasrah	◌ِ	◌ٍ

In Table 3.6 we put the Sokon as original diacritics with no derivative. In the Arabic language, there are letters without diacritics characters that are written in words to indicate the permissibility of more than one possible diacritic character, so we need to initiate a no-diacritics character in the table which must be a default diacritics character. To represent such a case, we will consider that Sokon is diacritics derived from "no- diacritics ". In the same way as in the previous paragraph, we will add three bits to represent the diacritics character as shown in Table 3.6.

Table 3.6: Representation of Diacritics Characters

Diacritic	Representation	Diacritic	Representation
No diacritics	000	Sokon	100
Fatha	001	Tanween Alfatha	101
Dhama	010	Tanween Aldham	110
Kasrah	011	Tanween Alkasar	111

In Arabic letters, there is another diacritics character called "shadda" (◌ّ) used if a word contains a repeated character consequently, provided that the first letter is with Sokon diacritic, shadda is eliminated in the proposed representation.

Here it should be noted that the letter (ا) is without a diacritic and the letter (أ) is the same letter with Sokon diacritic and the letter (آ) can be written as double "ا" without a diacritic, while 'ى' is written 'ا' with 'ً' diacritic. The letter "ء" has four other forms written according to the spelling rules that depend on diacritics character precede or followed, they are (أ, إ, ؤ, ئـ).

Thus, we were able to represent the Arabic letter and its diacritics using one byte instead of two, and accordingly, the size of the Arabic texts was reduced into the half. Table 3.7, shows the representations of sample of two different letters followed by all possible diacritics.

Table 3.7: "ص" and "ك", With Diacritic Representations

Character	8-bit coding	Decimal value	Note
ص	00000111	7	Without diacritics
صَ	00100111	39	7+32
صُ	01000111	71	7+64
صِ	01100111	103	7+96
صْ	10000111	135	7+128
صً	10100111	167	39+128
صٌ	11000111	199	71+128
صٍ	11100111	231	103+128
ك	00011011	27	Without diacritics
كَ	00111011	59	27+32
كُ	01011011	91	27+64
كِ	01111011	123	27+96
كْ	10011011	155	27+128
كً	10111011	187	59+128
كٌ	11011011	219	91+128
كٍ	11111011	251	123+128

3.4 The Software Design Stage

The design stage of a program is an analytical process for the requirements of the program to choose and build the structure of the program and its parts and how they

relate to each other. This results in a set of files, models and graphs from which it is possible to program and write the program code "completely".

The design phase is divided into several stages, the most important of which are:

1- Architectural design
2- Designing software models using the UML - Unified Modeling Language graphic language
3- GUI - Graphical User Interface design

This is a brief explanation of each stage as follows:

3.4.1 Architectural design

Just as building a building requires first determining the shape of its structure, so does building the program. A program structure is the process of arranging parts of a program in a specific way Arrange and organize the connection of these parts with each other. Often this does not require the innovation stage, as there are known and specific types of structures. All that is required at this stage or choosing the appropriate structure for the program. The choice of structure depends on the type of program and on studying other important aspects such as performance, protection, security and safety, and ease of maintenance. The following sections the architectural design is depicted in detail.

3.5 Arabic Lettering Program

From the foregoing, it is clear that we have allocated a storage value for each of the Arabic letters, and the truth is that the letters stored in the computer storage do not resemble the letters on the screen or on the printer, and the truth of this difference is from the way letters are written within Arabic texts because most Arabic letters are not written in one form within the words but rather most letters have more forms that differ according to their position in the word.

In general, each letter has three forms, which are: at the beginning of the word (frontal), at the end of the word (final), and in the middle of the word (middle). In addition, the letters of the Arabic language are divided into three groups as shown in table 3.8, they are:

1- 8 letters have the same form in the three positions they are :(ا,د,ذ,ر,ز,و,ط,ظ).

2- 17 letters have the same form in frontal and middle and differ from the final position; they are (**س, ش, ص, ض, بـ, تـ, ثـ, نـ, يـ, حـ, جـ, خـ, فـ, ق, ك, ل, م**).

3- 3 letters have different form in each position they are (**ع, غ, هـ**).

The first group is considered as the separated letters, while the second and the third groups are the connected letters which written connected to the previous or next letter or connected with both.

Table 3.8: Forms of Arabic Letters

Arabic character	Frontal	Middle	Final
ا,د,ذ,ر,ز,و,ط,ظ	ا,د, ذ, ر, ز,و,ط,ظ	ا,د, ذ, ر, ز,و,ط,ظ	ا,د, ذ, ر, ز,و,ط,ظ
بـ ,تـ ,ثـ ,نـ ,يـ ,حـ, جـ, خـ س, ش, ص, ض فـ, قـ, كـ, لـ, مـ	بـ,تـ ,ثـ ,نـ ,يـ, حـ, جـ, خـ سـ, شـ, صـ, ضـ فـ, قـ, كـ, لـ,مـ	بـ، تـ،ثـ ,نـ,يـ, حـ, جـ, خـ سـ، شـ, صـ, ضـ فـ, قـ, كـ, لـ,مـ	ب ,ت ,ث ,ن,ي, ح, ج, خ س, ش, ص, ض ف, ق, ك, ل, م
ع, غ, هـ	عـ, غـ ,هـ	عـ, غـ, هـ	ع,غ or ع, غ ه or ـه

3.6 chng code

We have agreed previously to name the letter which represents (11111) as the change code (chng) as it appears in Table No. 3.4. The Arabic text maybe contains any other characters or symbols or numbers which are not an Arabic letters or diacritics and can be represented by ASCII table provided that each must be preceded by chng code.

3.7 General Structure of the Encoding and Decoding Algorithm

The main structure of the coding algorithm is depicted in Figures 3.1, 3.2 explain the flowchart of the proposed coding and encoding algorithm respectively. The application uses the Arabic alphabet as a string called alphabet = **' ابنحدرسصطعفلمهو ءيتجذزشضظغقكثةخ',** which is ordered in special sequence proposed previously in this thesis.

Arabic language is right justified; therefore, the most right letter is in position 1 and so on the last letter is the most left character. The diacritics characters are stored in string variable called diacritic="ًٌٍَُِّْ" in the coding program.

The algorithm provides two options: coding and encoding. The coding procedure work on merging the Arabic letter with its diacritic code in a single representation code, while the encoding procedure used to separate them for writing.

Python 3.8.5 is used to build the coding and encoding algorithms. Algorithm 1, Algorithm 2 depicted the essential steps to perform the encoding and decoding algorithms.

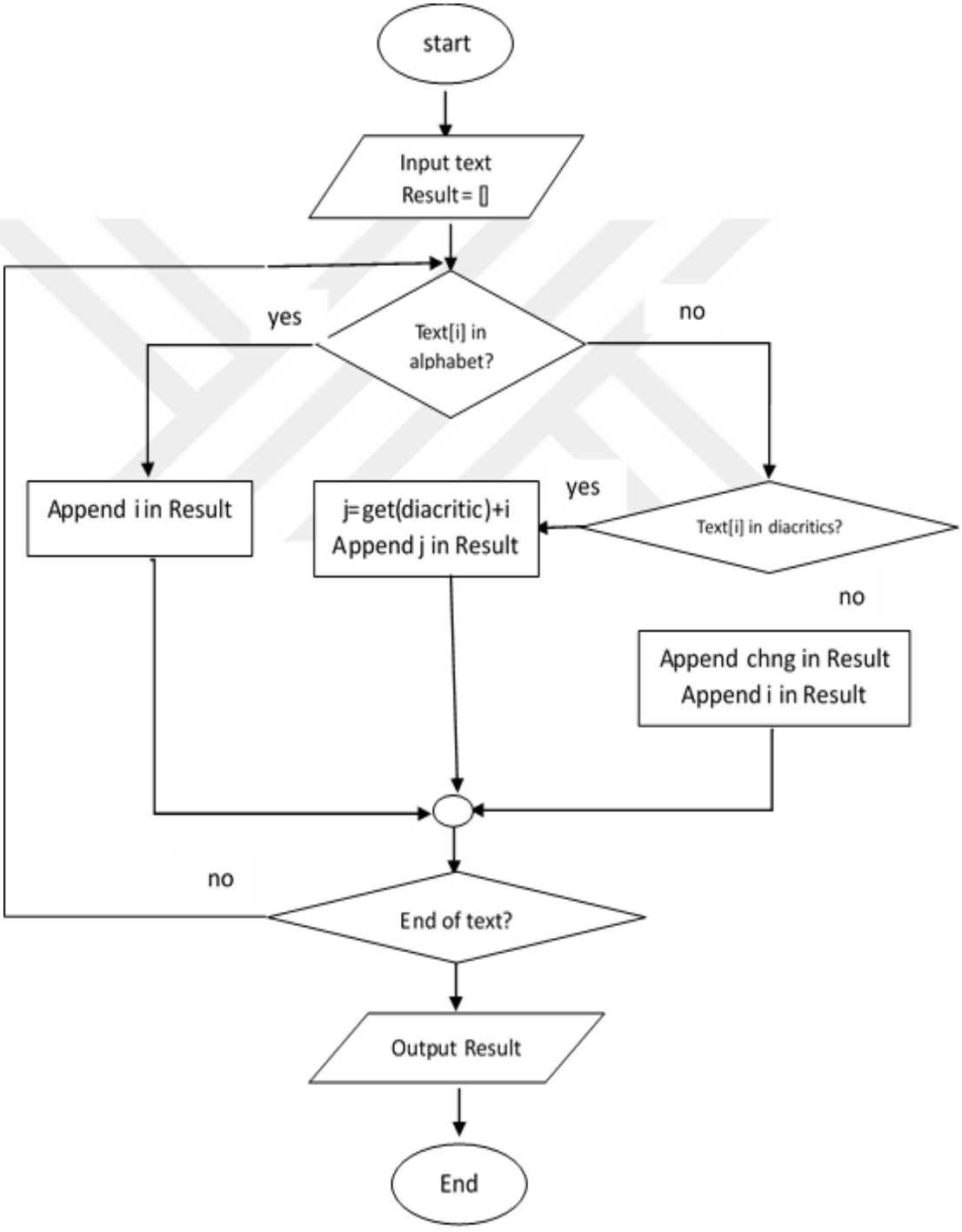

Figure 3.1: Flow Chart of the Encoding Algorithm

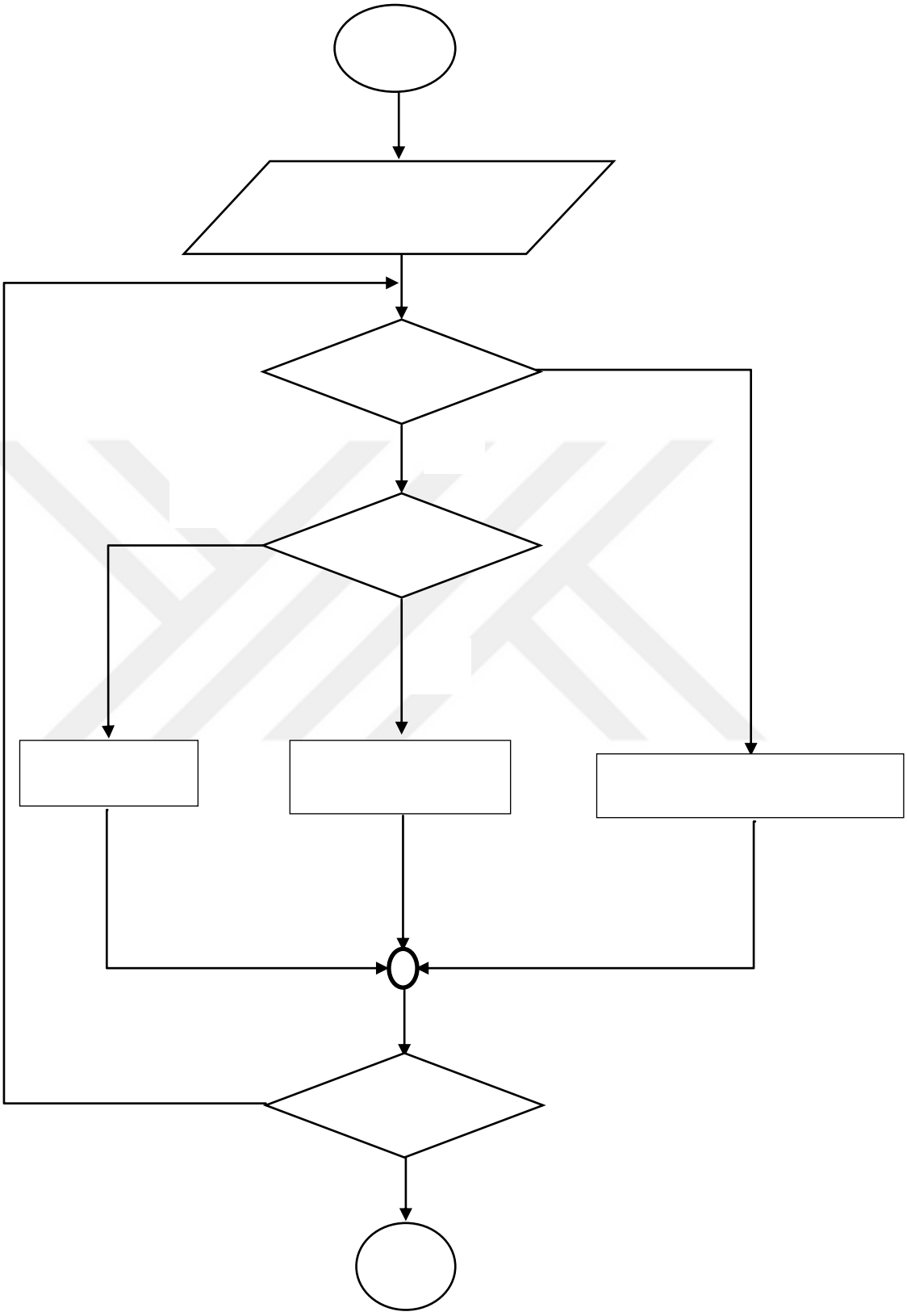

Figure 3.2: Flow Chart of the Decoding Algorithm

Algorithm 1:

Input: Text (string)

Output: Result (list of numbers)

alphabet = 'ابنحدرسصطعفلمهوءيتجذزشضظغقكثةخ'

hamza = 'أإئؤ'

alf = 'آاأى'

diacritics = {'َ': 32, 'ُ': 64, 'ِ': 96, 'ْ': 128, 'ً': 160,'ٌ': 192,'ٍ': 224}

- Repeat for each ch in Text
- if ch in alphabet then:
  i = index of ch in alphabet
  append(i) to Result
- if ch is Hamza list:
  append (16) to Result
- if ch in alf list:
  i = index of ch in alf list
  if i = 0:
  append (0,0) to Result
  if i=1:
  append (0,128) to Result
  if i=2:
  append (0,192) to Result
- else if ch in diacritics:
  j = get the last code in Result
  delete the last code in Result
  i = index of ch in diacritics + j
  append(i) to Result
- else if ch = 'ّ'
  j = get the last code in Result
  delete the last code in Result
- append (j + 128) to Result
- append(j) to Result
- else:
  append (31) to Result
  append (ASCII (ch)) to Result

Algorithm 2:

Input: List (list of numbers)

Output: Text (string)

Alphabet ='ابنحدرسصطعفلمهوءيتجذزشضظغقكثةخ'

Hamza = 'أإئؤ'

Alf = 'اآأى'

diacritics = {'ً': 32, 'ٌ': 64, 'ٍ': 96, 'ْ': 128, 'ّ': 160,'ٓ': 192,'ٔ': 224}

- Repeat for each i in List
- If i = 31 append (List[i+1]) to Text
- If i>31 subtract the value of diacritic(j) from i
    - Ch=diacritic[i]
- If i=16 select j from Hamza
    - Append j+ch to Text
- If i=0 select j from Alf
    - Append j+ch to Text
- Else append i+ch to Text

3.8 GUI Design

The user interface is the interface of the program that the user interacts with. It mainly includes the visual part, which usually consists of windows, buttons, writing fields, background color, font color ... etc.

This section relates to the technical and aesthetic part of the program. How much he cares about the responsiveness of the interface and its ability to give clear and sufficient information to the user. Python 3.8.5, is used to design the program interface.

Figure 3.3 shows the program interface of the conversion from ASCII to the proposed representation with two display text boxes one for interior representation and the second for normal representation.

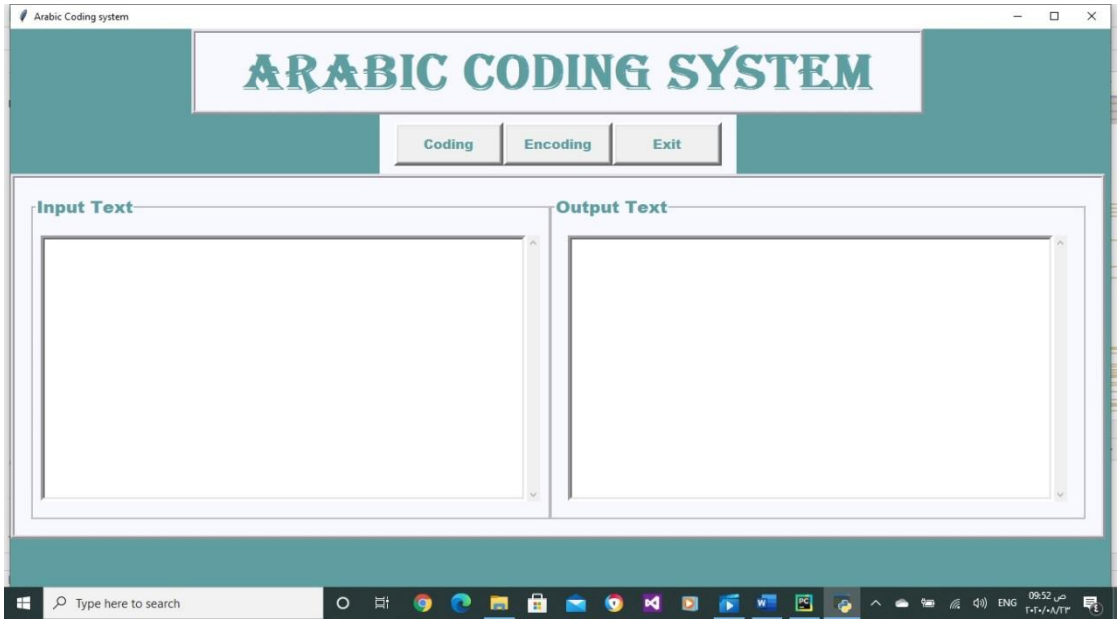

Figure 3.3: Application Interface

3.9 Implementation Stage

Implementation Software Engineering, in this chapter, we will examine programming methods, and challenges in implementing programs.

In the coding process, the increases of the size of the program which becomes impossible to remember the program flow is avoided by using structured programming because if we forgets how basic programs, files, and procedures are created, then it becomes very difficult to share, debug and modify the program. It encourages us to use subroutines and loops instead of using simple hops in the code, thus achieving clarity in the code and improving its efficiency as structured programming helps to reduce coding time and organize the code properly.

By using structured programming modular programming is performed to broke down the program into a smaller set of instructions. These groups are known as modules, procedures, or subprograms.

In procedural programming, a procedure can produce different results when it is called with the same argument, as the same program can be in a different state while communicating with it. This is a characteristic as well as a disadvantage of procedural programming, in which the sequence or timing of the execution of an action becomes important.

3.10 Programming style

The programming method is set from coding rules followed by all programmers to write code. When multiple programmers are working on the same software project, they often have to work with the program code written by some other developer. This becomes boring or impossible sometimes if not all developers follow some standard programming patterns for coding a program.

The programming style of python with its indentation characteristic makes the program code legible and understandable by everyone, which in turn makes debugging and troubleshooting easier.

3.11 Compression Ratio

Running the application on some samples of the holy Quran with different sizes shows that the text has been compressed with ratio in the range 66% to 44% which

obtained from the input file size and the output file size depending on the using of diacritics and numbers and symbols in the input text. Figure 3.4, shows the application interface with compression ratio.

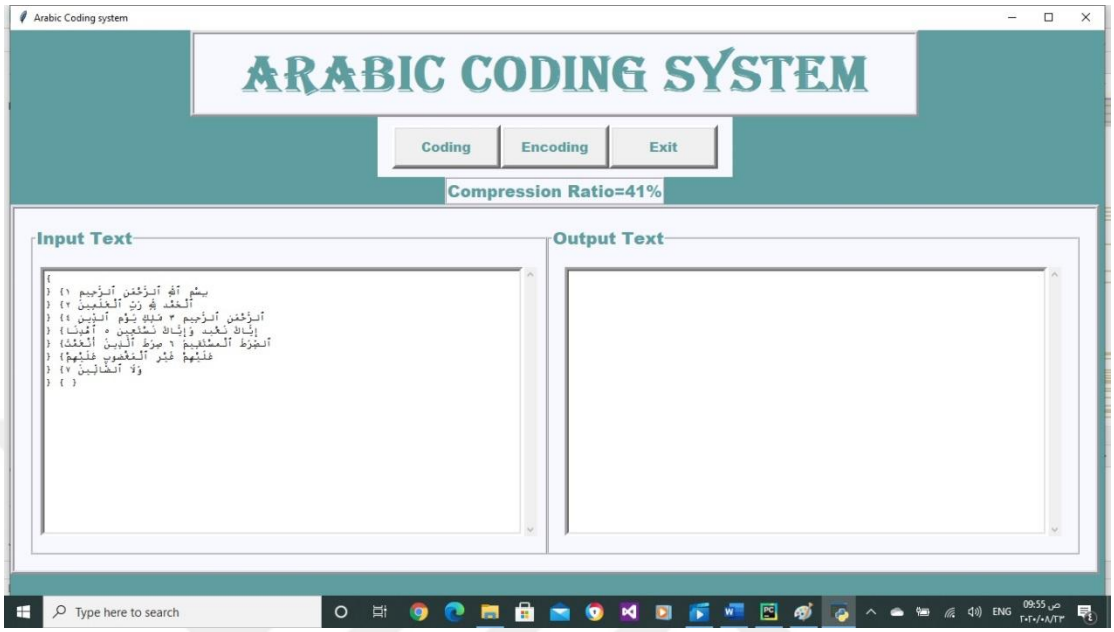

Figure 3.4: Compression Ratio in Applications Interface

The size of the output file may equal the size of input file when the Arabic text contains no diacritics

3.12 Performance Test

The program passes several tests, as follows:

1. Unit Test: In which a specific unit of the program is tested, such as, class, function.
2. Test a group of units working together.
3. Test part of the system or program.
4. Test the program in full.

The purpose of this test is to test that the program performs the required functions, and test its performance and effectiveness in terms of speed and space used, and there are other objectives such as measuring the program's ability to repair and change. Two main types of tests are used, they are:

1. White box: In which, the structure and code of each unit is studied and tested to ensure their validity. This test is conducted in the initial stages.
2. Black box: In which, the inputs are done with the unit and program. This test is done after the white box test.

3.13 Experimental Results Analysis

The main goal of designing and building an Arabic letters encoding scheme is to use a method derived from understanding the letter and diacritics in the Arabic language to find a suitable representation of the Arabic letter and its diacritic, and we have come to create a representation of the character and its diacritics in the size of one byte, while they were represented by the size of two bytes. Thus, we obtained a compression ratio equivalent to 50 percent. Whereas, if Arabic texts were used without diacritics, the text would not be compressed. Table No. 3.9 showing the compression ratio for four files with the same text, the first file has a 100 percent diacritical proportion and the second file contains the same text, but at a ratio of 60 percent of diacritics The third file contains the same text at a ratio of 30 percent of diacritics and the fourth file contains the same text without diacritics.

Table 3.9: Decoding Results

Arabic Text File	Original File Size (KB)	Encoded File Size (KB)	Compression Percentage
File1.txt	10.0KB	3.4KB	66%
File2.txt	8.28KB	3.3KB	61%
File3.txt	5.69KB	3.2KB	44%

3.14 Arabic Coding Scheme Management

Figure 3.5 shows the Arabic coding scheme management as a software project management developing stages.

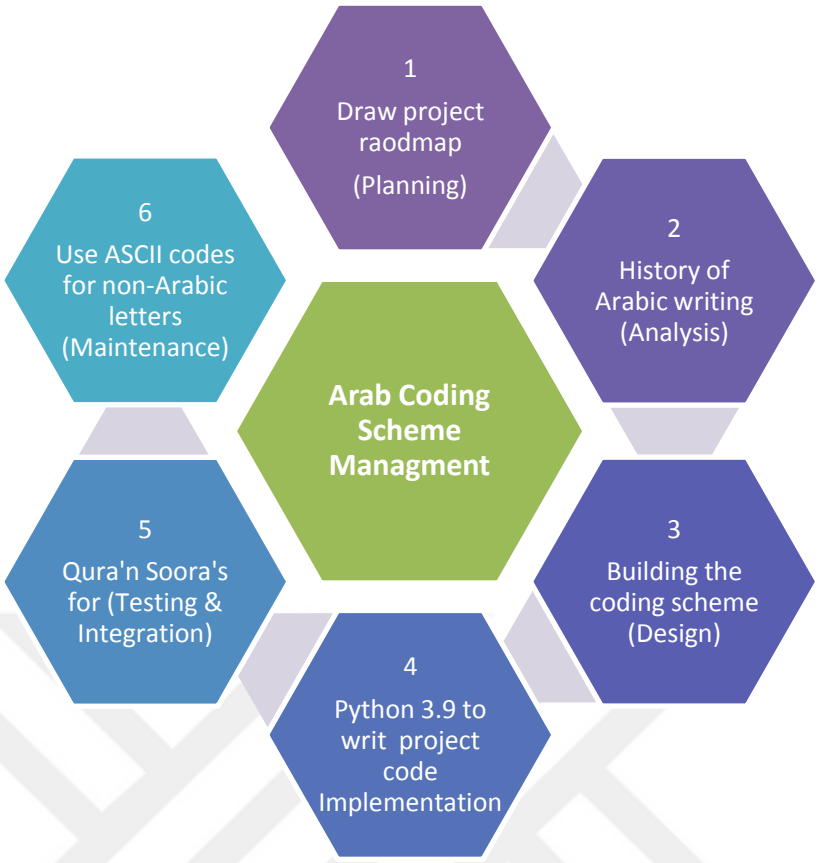

Figure 3.5: Arab Coding Scheme Management

4. CONCLUSIONS AND RECOMMENDATIONS

4.1 Conclusions

We have focused on this research to develop a new compressed coding system for Arabic letters as an alternative to the ASCII system in an Arab operating environment and it is an important essential part to start building Arab operating system. Otherwise, the new representation system can be used in other applications such as an Arabicization system or a text compression system. From this thesis we can conclude the following:

- We can develop a good coding scheme by studying the characteristics of the Arab language as shown in this thesis.
- The proposed coding scheme is a compressed scheme with a ratio of 69%.
- The proposed coding scheme is a bilingual coding scheme for the Arabic language with other languages like English.
- The proposed coding scheme can be used as a compression application or can be used as a part of the Arabic editor system Microsoft Word and others.
- The proposed coding scheme needs to redesign a new keyboard suitable for the new representation method with the distribution of a new letter on the keyboard to be used as an editor system.

4.2 Recommendations

During developing the coding application a new approaches jump in mind to be developed, some of them are followed:

- Studying the most frequented letters in the Arabic language to redistribute them on the keyboard or redesign a new bilingual keyboard depending on the characteristics of the Arabic language.

- Adding the Arabic dictation rules to the application to correct the wrong Arabic words in the text.
- Add an option to select non- diacritics Arabic text with an expected compression ratio of 60%.
- Design a new bilingual keyboard suitable for the proposed coding scheme.

4.3 Contributions

- One of the most contributions of this article is to build a representation based on a deep understanding of the Arabic language and not as it was in the past, where the Arabic letters were represented in a similar way to represent the Latin language.
- The second contribution of this article is in the field of data compression, we creating a compressed representation of at least 66%.

REFERENCES

[1] **N. A.** (2016). Gordon, "Issues in retention and attainment in Computer Science," no. March.

[2] **N. Alanazi, E. Khan, and A. Gutub**, "Inclusion of Unicode Standard seamless characters to expand Arabic text steganography for secure individual uses," *J. King Saud Univ. - Comput. Inf. Sci.*, no. xxxx, 2020, doi: 10.1016/j.jksuci.2020.04.011.

[3] **W. Helali, Z. Hajaiej, and A. Cherif,** "Arabic corpus implementation: Application to speech recognition," *2018 Int. Conf. Adv. Syst. Electr. Technol. IC_ASET 2018*, pp. 50–53, 2018, doi: 10.1109/ASET.2018.8379833.

[4] **M. Johnson** *et al.*, "Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 339–351, 2017, doi: 10.1162/tacl_a_00065.

[5] **A. Al-Ameer,** (1969). "University of London," *Univ. London*, vol. 5, no. August,.

[6] **A. Ibady**, (2010). "A new localization and compression system Dr. Abdulkareem Ibady Baghdad college of economic science 2010"

[7] **T. A. Hilal and H. A. Hilal,** "Arabic text lossless compression by characters encoding," *Procedia Comput. Sci.*, vol. 155, no. 2018, pp. 618–623, 2019, doi: 10.1016/j.procs.2019.08.087.

[8] **F. Thaher,** "Dynamic with Dictionary Technique for Arabic Text Compression," *Int. J. Comput. Appl.*, vol. 135, no. 9, pp. 4–9, 2016, doi: 10.5120/ijca2016908299.

[9] **A. Awajan,** (2011). "Multilayer model for Arabic text compression," *Int. Arab J. Inf. Technol.*, vol. 8, no. 2, pp. 188–196.

[10] **A. Awajan and E. A. Jrai**, (2015). "By Arafat Awajan & Enas Abu Jrai HybridTechniqueforArabicTextCompression," no. August.

[11] **R. Sakthi Murugan and V. S. Ananthanarayana**, "WordCode using WordTrie," *J. King Saud Univ. - Comput. Inf. Sci.*, no. xxxx, 2019, doi: 10.1016/j.jksuci.2019.05.011.

[12] **A. A. Mohamed**, "An improved algorithm for information hiding based on features of Arabic text: A Unicode approach," *Egypt. Informatics J.*, vol. 15, no. 2, pp. 79–87, 2014, doi: 10.1016/j.eij.2014.04.002.

[13] **R. Ayadi, M. Maraoui, and M. Zrigui**, (2016). “A Survey of Arabic Text Representation and Classification Methods,” *Res. Comput. Sci.*, vol. 117, no. 1, pp. 51–62, doi: 10.13053/rcs-117-1-4.

[14] **J. M. Reddy, S. V. A. V Prasad, and B. V. R. Murthy**, (2017). “Significance of Software Layered Technology on Size of Projects,” *J. Sci. Technol.*, vol. 2, no. 1, pp. 46–58, [Online]. Available: www.jst.org.in.

[15] **A. Bacchelli and C. Bird**, (2013). “Expectations, outcomes, and challenges of modern code review,” *Proc. - Int. Conf. Softw. Eng.*, pp. 712–721, 2013, doi: 10.1109/ICSE.6606617.

[16] **J. F. Perez and G. Casale**, (2017). “Line: Evaluating Software Applications in Unreliable Environments,” *IEEE Trans. Reliab.*, vol. 66, no. 3, pp. 837–853, doi: 10.1109/TR.2017.2655505.

[17] **M. Christensen and R. H. Thayer**, (2015). “Software Requirements Specification,” *Proj. Manag. Guid. to Softw. Eng. Best Pract.*, doi: 10.1109/9781118156629.ch3.

[18] **A. M. Davis and P. Sitaram,** (1994). “A concurrent process model of software development,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 19, no. 2, pp. 38–51, , doi: 10.1145/181628.181637.

[19] **M. J. M. Davis,** (1995). “Process and Product : Dichotomy or Duality ? The Maturity Movement and Acedia .,” vol. 20, no. 2, pp. 17–18.

[20] **D. E. Perry and L. G. Votta**, (1994). “Prototyping a Process Monitoring Experiment,” *IEEE Trans. Softw. Eng.*, vol. 20, no. 10, pp. 774–784, doi: 10.1109/32.328994.

[21] **P. Heck and A. Zaidman**, (2018). *A systematic literature review on quality criteria for agile requirements specifications*, vol. 26, no. 1. Springer US.

[22] **Z. Kurtanovic and W. Maalej**, (2017). “Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning,” *Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. RE 2017*, pp. 490–495, doi: 10.1109/RE.2017.82.

[23] **D. Arruda and N. H. Madhavji**, (2017). “Towards a requirements engineering artefact model in the context of big data software development projects: Research in progress,” *Proc. - 2017 IEEE Int. Conf. Big Data, Big Data 2017*, vol. 2018-Janua, no. v, pp. 2314–2319, 2017, doi: 10.1109/BigData.2017.8258185.

[24] **N. H. Bakar, Z. M. Kasirun, N. Salleh, and A. H. A.Halim**, (2017). “Extracting Software Features From Online Reviews to Demonstrate Requirements Reuse in Software Engineering,” *6th Int. Conf. Comput. Informatics, ICOCI 2017*, no. 157, pp. 184–190.

[25] **P. Update**, (2008). “Unicode character encoding model,” pp. 1–23.

[26] **M. Mehrnoush, B. J. Belzer, K. Sivakumar, and R. Wood,** (2017). "EXIT Chart-Based IRA Code Design for TDMR Turbo-Equalization System," *IEEE Trans. Commun.*, vol. 65, no. 4, pp. 1762–1774, doi: 10.1109/TCOMM.2017.2662003.

[27] **T. Henning, P. Plitzner, A. Güntsch, W. G. Berendsohn, A. Müller, and N. Kilian**, (2018). "Building compatible and dynamic character matrices–Current and future use of specimen-based character data," *Bot. Lett.*, vol. 165, no. 3–4, pp. 352–360, doi: 10.1080/23818107.2018.1452791.

[28] **S. K. Mukhopadhyay, M. O. Ahmad, and M. N. S. Swamy,** (2018). "SVD and ASCII Character Encoding-Based Compression of Multiple Biosignals for Remote Healthcare Systems," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 137–150, doi: 10.1109/TBCAS.2017.2760298.

[29] **S. Mahato, D. K. Yadav, and D. A. Khan**, (2020). "Personal characters to bits mapping using Dot Pattern Character Encoding Scheme (DPCES)," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 32, no. 2, pp. 197–207, doi: 10.1016/j.jksuci.2017.08.003.

[30] **J. Newmarch**, (2017). *Network Programming with Go*.

[31] **J. Stewart and J. Uckelman,** (2013). "Unicode search of dirty data, or: How I learned to stop worrying and love unicode technical standard #18," *Proc. Digit. Forensic Res. Conf. DFRWS 2013 USA*, vol. 10, pp. S116–S125, doi: 10.1016/j.diin.2013.06.013.

[32] **S. Rosmorduc**, "The Thot Sign List (TSL) & hieroglyphic encoding."

[33] **I. S. Alkhazi and W. J. Teahan**, (2019). "Compression-Based Parts-of-Speech Tagger for The Arabic Language," no. 2, pp. 1–15.

[34] **X. Wang, H. Pham, P. Arthur, and G. Neubig,** (2019). "Multilingual neural machine translation with soft decoupled encoding," *7th Int. Conf. Learn. Represent. ICLR 2019*, pp. 1–13.

[35] **T. U. Standard,** (2020). "Arabic Mathematical Alphabetic Symbols".

[36] **P. du Gay, T. Lopdrup-Hjorth, K. Z. Pedersen, and A. O. Roelsgaard,** (2019) "Character and organization," *J. Cult. Econ.*, vol. 12, no. 1, pp. 36–53, doi: 10.1080/17530350.2018.1481879.

[37] **S. Ecma,** (2000). "8-Bit Single-Byte Coded Graphic Character Sets Latin / Arabic Alphabet," no. December.

[38] **I. Stamelos,** (2010). "Software project management anti-patterns," *J. Syst. Softw.*, vol. 83, no. 1, pp. 52–59, doi: 10.1016/j.jss.2009.09.016.

[39] **D. Pandey, U. Suman, and A. K. Ramani**, (2010). "An effective requirement engineering process model for software development and requirements management," *Proc. - 2nd Int. Conf. Adv. Recent Technol. Commun. Comput. ARTCom 2010*, pp. 287–291, doi: 10.1109/ARTCom.2010.24.

[40] **R. Banker, S. Datar, C. Kemerer, and D. Zweig,** (2002). “Software Errors and Software Maintenance Management,” *Inf. Technol. Manag.*, vol. 3, no. 1/2, pp. 25–41, doi: 10.1023/A:1013156608583.

[41] **D. P. Voorhees**, *Texts in Computer Science Guide to Efficient Software Design*. .

RESUME

EDUCATION:

Bachelor : 2010, Baghdad college of economic science, software engineering, Baghdad, Iraq

SKILLS:

- Prolog, C++, Pascal programmer and Visual Basic.net, HTML, python.
- Lecturer for the following subjects, computer graphic, compiler.
- Microsoft Office Application.
- Microsoft IC3.
- Photoshop.
- AutoCAD.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- 'A New Arabic Coding Scheme', International Journal of Engineering and Natural Sciences (IJENS's), Vol.2. No.3.