CHAPTER 4

NEW DIGITAL QURAN MODEL TO OPTIMIZE STORAGE

4.1 Introduction

In this chapter, the detailed algorithm development of the Digital Quran Model that will reduce the storage space will be discussed and presented; that integrates 3 main mechanisms: a hexadecimal representation for Quran text format, compressed digital Quran content structure and word duplication handling to manage the memory space efficiently. Figure 4.1 outlines the structure of this chapter.

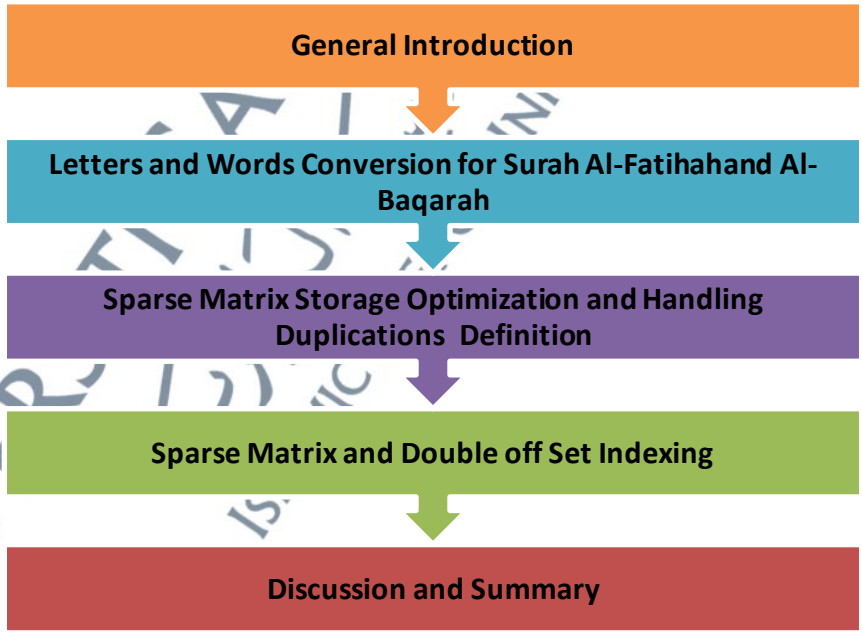

Figure 4.1: Outline the Structure of Chapter 4

4.2 Letters and Words Conversion

The new DQM technique is used to represent all words of the Holy Quran using Unicode. Space required for each word of the Holy Quran before and after the conversion to hexadecimal format for Arabic characters will be compared. Table 4.1 summarise statistics of the Holy Quran from Hammo et al. (2008). In this study, data from *qurananalysis.com* will be referred to which is part of a research initiative by the University of Leeds's Semantic Search and Intelligence System for the Quran.

It is important to note that the following techniques have been used by extant literature and various scholarly works relevant to the context of current research. This further supports the study's conduct while establishing a good and reliable approach for its conduct (i.e., Hakak et al., 2019; Almazrooie et al., 2020). Building on the resources as mentioned earlier, the following results are derived from the current analyses and investigation with the specifics of this study.

Table 4.1: Statistics of the Holy Quran

Number of pages	604
Number of chapters	114
Number of parts	30
Number of sections	60
Number of verses	6236
Number of words	77439
Number of distinct vowelised words	19273
Number of distinct non-vowelized words	14918
Number of distinct roots	1767
Number of letters	320015

Source: Hammo et al (2008)

A statistical account of certain Quranic expressions reveals a unique Quranic textual feature of 'numerical symmetry', which can be taken as an exciting feature of coherence in Quranic discourse. Table 4.2 presents examples of repetition of Quran words that appear in the Quran in the same number.

Table 4.2: Examples of Repetition of Quran Words

Word	Word Pair	Occurrence
(الدنيا - this world)	(الاخرة- the Hereafter)	115
(الملائكة - the angels)	(الشياطين - the devils)	88
(الموت - death)	(الحياة – life)	145
(الصيف والحر- summer)	(الشتاء - winter)	5
(السيئات- evil deeds)	(الصالحات - good deeds)	167
(الناس - people)	(الرسل - prophets)	241
(الكفر- disbelief)	(الايمان- faith)	17
الشهر- month)	i.e., the same number as the months of the year.	12
(اليوم - day)	i.e., the same number as the days of the year.	365
(الأيام - the days)	i.e., the same number as the days of the month.	30

4.2.1 Quran Words Conversion Algorithm

The word structure is a challenge in processing the digital Quran representation of Arabic letters. Figure 4.2 illustrate an example of a set of Arabic letters (ئ،ث،ذ،ع،ص،لا،هـ،ك) with its four expected position and its Unicode hexadecimal representation.

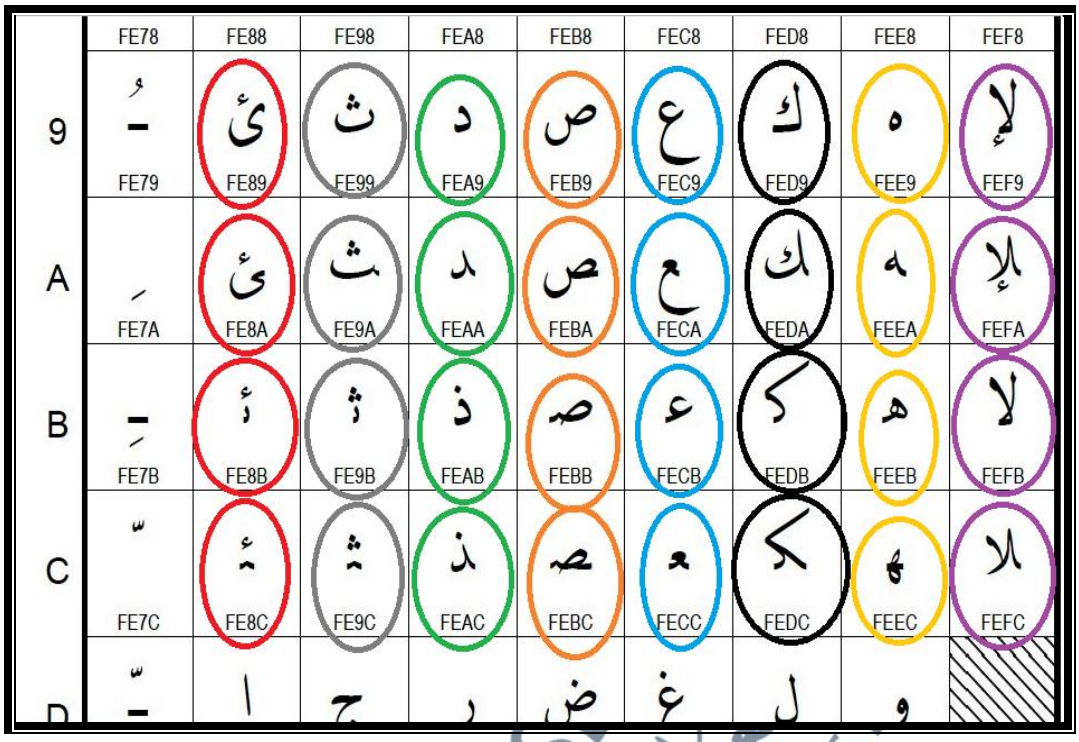

Figure 4.2: Arabic Letters (ى،ث،ذ،ع،ص،لا،هـ،ك) In Different Position With Their Hexadecimal Representation

For example, Surah Al-Fatihah contains 143 characters, 29, 19 unique words, and seven verses. First, based on each letter's hexadecimal representation, the words' conversion will be generated. The conversion algorithm into hexadecimal representation for the Quran word is shown in Figure 4.3, following the formulation in Equations 3.1 and 3.2.

Steps	***Procedures*** **wordConversion**
1:	***Input***: *Quranic Word in Arabic*
2:	***Output* 1**: *Hexadecimal Code of Arabic Word*
3:	***Start***
4:	*Quranic Word* ← *W*
5:	*Tokenize Word into Letters*
6:	$W = \{h1, h2, \dots, h_n\}$
7:	*Check Look Up table for Hexadecimal Code*
8:	*Letter Value in Hexadecimal Code*
9:	*While* ($h_n != \emptyset$)
10:	$CHhex = Findunicode(ch \& "\_")$
11:	
12:	$Hex.Cod = \sum (h1, h2, \dots, h_n)$
13:	*Word Value in Hexadecimal Code*
14:	***End Loop***
15:	***End***

Figure 4.3: Word Conversion Algorithm into Hexadecimal Representation

For example, in the word (الرحيم) AL RAHIM, the new hexadecimal representation fives $5F893$ as discussed in Chapter 3 page 68 Eq., 3.3 which yield about 58.33% of reduction as summarized in Table 4.3 according to Equation 4.1.

$$Reduction\ \% = \frac{Size_{Arabic\ word} - Size_{Hexadecimal\ word}}{Size_{Arabic\ word}} \times 100$$
$$= \frac{12-5}{12} \times 100$$
$$= 58.33\% \quad (4.1)$$

Table 4.3: Storage Comparison Between words in Arabic and Hexadecimal

	Size (bytes)	**Size for a total of 148 Occurrences in Al-Quran**
الرحيم	12	1776
5*F*893	5	740
Reduction	**58.33%**	

4.2.2 Quran Verses Conversion Algorithm

Each verse in the Quran can then be represented using its Hexadecimal representation of each word in the verse. Based on the word conversion algorithm shown in Figure 4.3, the pseudocode to convert each verse is as presented in Figure 4.4.

Steps	***Procedures* Verse Conversion**
1:	***Input***: *Quranic Verse in Arabic*
2:	***Output* 1**: *Array of Hexadecimal Code of Words in the Verse*
3:	***Start***
4:	*Quranic Verse* $\leftarrow V$
5:	*Tokenize Verse into Words*
6:	$V = \{w1, w2, \ldots, w_n\}$
7:	*i=0;*
8:	*While not end of V*
9:	$whi = wordConversion\ (wi)$
10:	*List of words Wh=* $\{wh1, wh2, \ldots, wh_n\}$ *converted*
11:	***End***

Figure 4.4: Verse Conversion Algorithm into Hexadecimal Representation

For example, for Surah Al-Fatihah that has 7 verses, the verses conversion into Hexadecimal will be done 7 times accordingly. Let's say for the second verse, **الحمد لله رب العالمين** . When the method *verseConversion* (الحمد لله رب العالمين) been activated, the verse will be tokenized (Refer line 5 Figure 4.4) into 4 word i.e. **الحمد , لله , رب** and **العالمين.** Then, each word will be converted into a Hexadecimal representation by the the invocation of method *wordConversion*(الحمد**),** *wordConversion*(لله**),** *wordConversion*(رب**) and** *wordConversion*(العالمين**)** that will return the corresponding value of each word i.e *Wh* = {4F99C, 2FCAF, 1FD3C ,7F65F } based on Eq. 3.1 – 3.3.

4.3 Storage Optimization with Compressed Content Structure and Duplication Handling.

As discussed previously in Section 3.2.2-3.2.4, the Digital Quran content structure use table representation or matrix as adopted by Almazrooie et al. (2020) who used 6236 rows of elements to represent each verse in the Quran whereas Hakak et al. (2017) used 6234 rows. This study further proposed a compressed table representation due the sparse nature of the table since each verse in the Quran has various length from shortest with one letter to the longest with 129 words (https://quranalysis.com/analysis/basic-statistics.php) . Besides that, Quranic words is repeating itself throughout the Quran. From 77,797 words in the Quran, only 14,870 words are unique. Based on these facts, an index like listing with unique ID for each word make up a look up table that can further reduces the verses representation by only referring to the unique ID. Table 4.4 illustrate the look up table feature.

Table 4.4: Lookup Table to Calculate Words with Total Letters Count, representation in Hexadecimal and Unique ID

Kalimah	Count	New Hex	ID
اليهود	6	5F8E4	1766
بنورهم	6	5F8E3	1767
نفعهما	6	5F8E5	1768
يكذبون	6	5F8E0	1769

This approach will lead to memory reduction thus saving space and time due to faster access at the presentation layer in a bit form. Figure 4.5 illustrate the process in creating the digital Quran content structure with look up table to handle repeating words with sparse matrix.

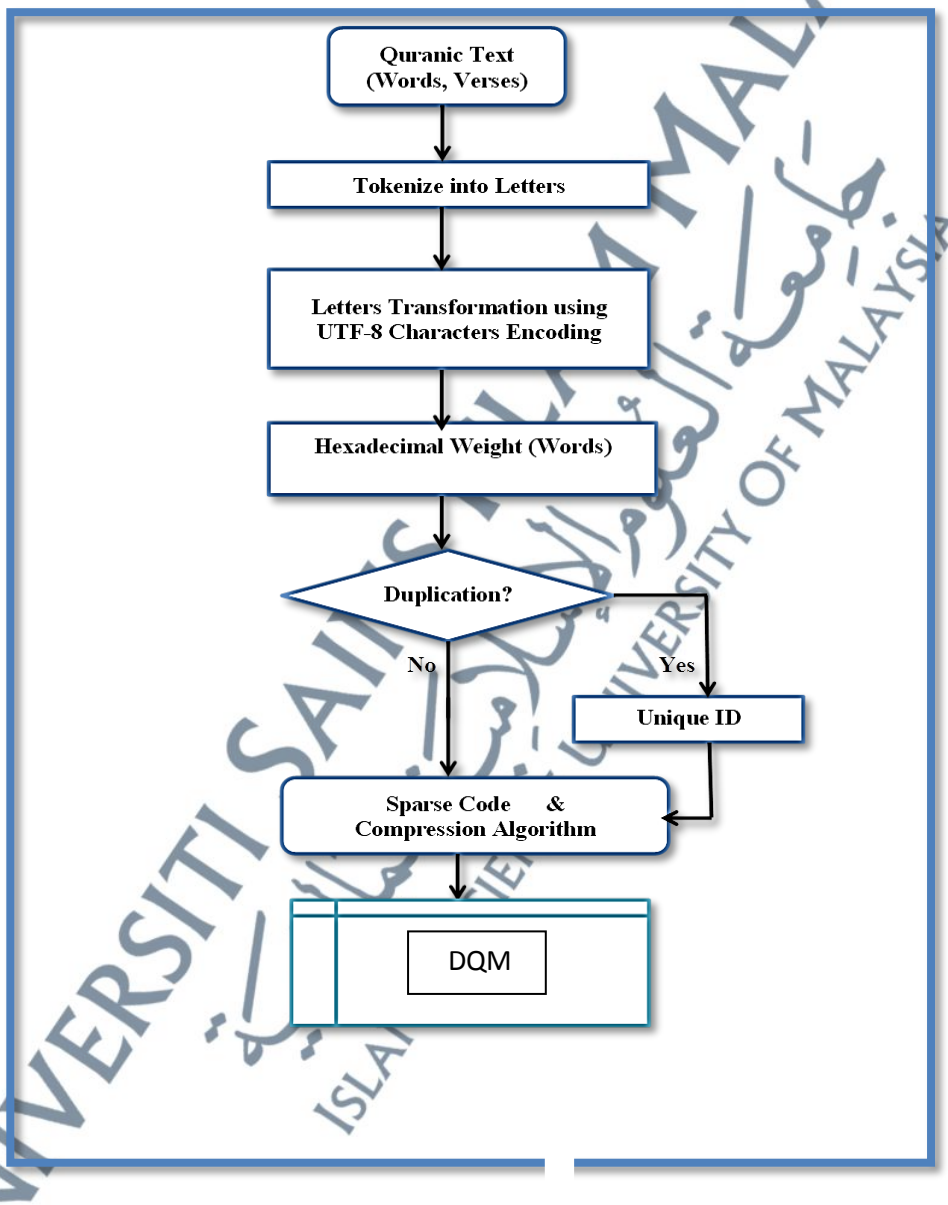

Figure 4.5: DQM in Sparse Matric with Unique ID Duplication Handling

Steps	Procedures
1:	***Input***: *Quranic Text in Arabic*
2:	***Output 1***: *Hexadecimal Code of Arabic Words*
3:	***Output 2***: *Sparse Matrix Holding Unique ID*
4:	***Start***
5:	*Quranic Verses, Words* $\leftarrow V$
6:	*Tokenize Verse in to Words W,*
7:	*Tokenize Word into Letters*
8:	$V = \{w1, w2, \dots, wn\}$
9:	$W = \{h1, h2 \dots, hn\}$
10:	*Check Look Up table for Hex.Cod*
11:	*Verse Value in Hex.Cod*
12:	*Word Value in Hex. Cod*
13:	*Letter Value in Hex. Cod*
14:	$CHhex = Findunicode(ch \,\&\, "-")$
15:	*For* $i = 2$ *To* $Len(txt) + 1$
16:	$prech = Mid(T, i - 1, 1) \leftarrow$
17:	$ch = Mid(T, i, 1) \leftarrow$
18:	$nextch = Mid(T, i + 1, 1) \leftarrow$
19:	*If* $Mid(T, i - 1, 1) = " "$ *And* $Mid(T, i + 1, 1) = " " \leftarrow$
20:	*Then*
21:	*'Print* $Mid(T, i, 1) + "->isolate"$
22:	*While* ($hn != \emptyset$)
23:	$Hex.Cod = \Sigma(h1, h2, \dots, hn)$
24:	***Hex.Cod*** $\leftarrow$ ***Unique.ID***
25:	***Construct Sparse Matrix***
26:	*Input: N // Order of the matrix*
27:	*Allocate memory for this.*$\{tablei\}$ $N\ i = 1$
28:	*for* $i = 1$ *to N do*
29:	*this.tablei* $\Rightarrow$ *NULL*
30:	*end*
31:	*End Procedure Construct*
32:	*Procedure AddDataForKeys*
33:	*Input: inData, i, j*
34:	*if this.tablei* $\Rightarrow$ *NULL then this.tablei.Construct()*
35:	*if this.ContainsKeys*$(i, j) = FALSE$ *then*
36:	$d.key \leftarrow j$; $d.data \leftarrow inData$ *// d is of type TableData*
37:	*this.tablei.Add*(d)
38:	*end*
39:	*End Procedure AddDataForKeys*
40:	*End Loop*
41:	*End Loop*
42:	*End*
43:	*for* $j = 1$ *to Ne do*
44:	*for* $k = 1$ *to* 4 *do*
45:	$edge.startNode \leftarrow elArrayj.nodesp1, k$
46:	$edge.endNode \leftarrow elArrayj.nodesp2, k$
47:	*edgeList.Add*$(edge)$
48:	*End*
49:	*End*
50:	***End***

Figure 4.6: Pseudo-Code for Digital Quran Verses in Sparse Matric with Lookup Table

4.4 Sparse Matrix Compression with Double off Set Indexing

The compression technique called double-offset indexing as explained in Section 3.2.3 is to reduce the unused space with zero elements in the matrix thus minimizing the storage requirements of a sparse table by eliminating default entries. However, the indices of a non-default entry must be stored in the compact table to facilitate non-default entry lookup as shown in Figure 3.5. A given row or column index can be repeated multiple. A compression method that tries to eliminate such redundancy and take advantage of default entries, the algorithm is presented in Figure 4.7.

The matrixes data structure or two-dimension array is one of the key components of structural computing with the other component which is double offset indexing to efficiently represent the matrix with non-zero elements to save empty spaces.

Steps	Procedures
1:	**COMPRESS** ( )
2:	**for** $i = 1$ **to** $w \times v$ **do**
3:	$V\_entry\ [i] \leftarrow default$
4:	**for each** $row \in \{1,2, \dots, w\}$ **do**
5:	$R[row] \leftarrow FINDSHIFT(row)$
6:	**for** $j = 1$ **to** v **do**
7:	**if** $T[row, j] \neq default$
8:	**then**
9:	$place \leftarrow R[row] + j$
10:	$V.entry[place] \leftarrow T[row, j]$
11:	$V.from\ row\ [place] \leftarrow row$
12:	**call** $TRUNC(V)$
13:	**end**
14:	**function** $FINDSHIFT(row)$: $Integer$
15:	$return \begin{pmatrix} w \times v - v \\ min\ FITS(row, shift) \\ shift = -v + 1 \end{pmatrix}$
16:	**end**
17:	**function** $FITS(row, shift)$: $Boolean$
18:	**for** $j = 1$ **to** v **do**
19:	**if** $T[row, j] \neq default$ **and not** $ROOMIN\ V(shift + j)$
20:	**then return** (**false**)
21:	**return** (**true**)
22:	**end**
23:	**function** $ROOMIN\ V(where)$: $Boolean$
24:	**if** $where \geq 1$
25:	**then**
26:	**if** $V\_entry\ [where] = default$
27:	**then return** (**true**)
28:	**return** (**false**)
29:	**end**
30:	**procedure** $TRUNK(V)$
31:	**for** $i = w \times v$ $down\ to 1\ do$
32:	**if** $V\_entry\ [i] \neq default$
33:	**then**
34:	**return** ( )
35:	**end**

Figure 4.7: Double Off-Set Indexing Algorithm to Reduce Storage Requirements

4.5 New Digital Quran Model with Lightweight Storage

The complete algorithm put together for the new Digital Quran model with Quranic word representation in Hexadecimal and compressed sparse matrix for the content structure that handle duplications through look up table of unique words is as described in Figure 4.8.

Steps	***DQM Procedure***
1:	***Input***: *A Set of Quranic Text in Arabic.*
2:	***Output***: *Hexadecimal Code of Arabic Words.*
3:	$V = \{w1, w2, \dots, wn\}$
4:	$W = \{h1, h2, \dots, hn\}$
5:	*Find Each Letter or Harf Representation in Hexadecimal Based on UTF* $-$ 8
	Check Harf Position:
6:	*Beginning, Middle, Last or Isolated.*
7:	***Repeat***
8:	***Verse Conversion***
9:	***Until*** *Last Verse*
10:	***Initialization***
11:	*Define Unique ID for Words Code.*
12:	*Initialize Sparse Matrix to hold Word and verse ID's.*
13:	***Placement***
14:	***Input***: *Unique ID for Words Code.*
15:	***Output***: *Sparse Matrix Holding Unique ID.*
16:	*Generate Sparse Matrix.*
17:	*Place Unique ID in Matrix Cells.*
18:	*Generate A Sequence of ID's for Verses.*
	Apply Double Offset Indexing Compression Techniques.
	End

Figure 4.8: The Algorithm Description of the Complete DQM Approach

The new DQM consists of 4 key algorithms as illustrated in Figure 4.9.

a. Word Conversion – to convert Quran words into Hexadecimal representation

b. Verse Conversion – to convert Quran verses into a DQM content structure

c. DQM in sparse matrix with Look Up table

d. DQM in compressed sparse matrix with double offset indexing

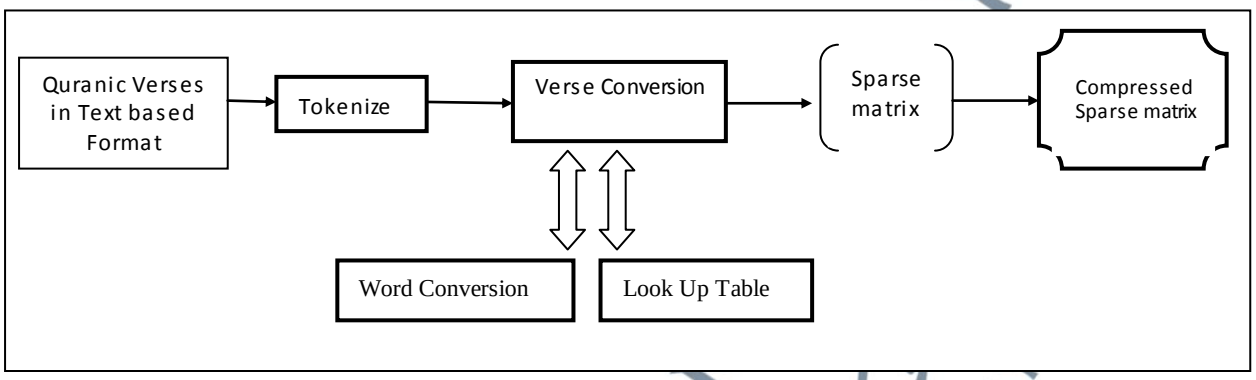

Figure 4.9: The key components of the new DQM with lightweight storage

4.6 Discussion and Consultation

The researcher has conducted consultation with an expert as mentioned in Chapter 3 with a qualitative approach was undertaken, with unstructured open-ended interview to gain in-depth knowledge and understanding that can be beneficial and add to the merits of this study. In accord with the consultation with the expert in software development specifically Quran, it was noted that the authenticity aspect is a matter that almost all applications and platforms follow a strict structure to establish. In this regard, the expert emphasized the fact that while most platforms and applications are reliable and use very accurate systems for data integration and verification, some sources are biased or deviated from the original meaning.

When the expert was questioned with a follow-up in the same regard, it was noted that a number of factors can be influential from the perspective of consultants that are namely, anti-Islamic groups or movements, novice developers that can have errors in their development processes, data integration flaws in terms of coding, and data transmission or data flow malfunctions (both in terms of coding and or structure), which imply a need and requirement for further optimization.

Moreover, the expert was asked whether there is a solid optimization system for digital content in general and specific to Quran. In response, the expert noted that an integrated holistic system should be implemented with clear standards and measures regarding development, transmission, translation, and other relevant factors. These standards should be agreed upon by authorities and Islamic governments so that documents

The consultant further explained that a commonly agreed-upon system and structure for digital Quran content can reduce or in the best-case scenario eliminate the falsehood of information from the sources that are unreliable and invalid. As users and developers can have access to a strategic system to encode or extract Quranic text, the validity of the source can be checked, which significantly improves safety measures. In addition, this in time can lead to the elimination of biased sources as there will be no usage due to an established system.

As a follow-up question, the consultant further elaborated that to achieve the above-mentioned consensus; it requires a legal and political process that should have economic

justifications for those who would undertake its conduct. Due to its complex nature that can be verified by most nations and/or institutions (governmental or non-governmental), it is a dire challenge for developers to establish software that can entail all aspects. However, "there is always hope".

When asked regarding the specifics of current research and its implications, the expert while acknowledging the merits of the study and its contributions to the literature, pinpointed limitations and restrictions that pose various challenges for the conduct of this study. According to the expert, it is virtually impossible for research to include all the elements that can be influential while maintaining its effectiveness, generalizability, and re-productivity. Furthermore, if a study aims to be conducted massively and on a great scale, it will require funding and a long process to first develop software, conduct experiments and pilot tests, design programs, collect longitudinal data and further optimize the software. This process is beyond the scope of current research. However, it is important to note that the current identified limitations of this research create a pathway for future studies in the same context that are explained in the final chapter.

4.7 Summary

This chapter presented the conceptual model of the new Digital Quran Model and presented 4 key algorithms involved. The overall combination of the algorithm together lead to an optimized space management thus may lead to lightweight version of the

digital Quran practical for standard application and maintain the content integrity of the Quran.