BAB 2
LANDASAN TEORI

3.1 Ionic Framework

Pada tahun 2006 merupakan awal dari smartphone era yang ditandai dengan luncurnya Iphone oleh Apple yang kemudian diikuti oleh *Google* yang meluncurkan Sebuah operating system yang menyamai *Iphone IOS* bernama *Android* dua tahun setelahnya. Hal ini juga merupakan awal lahirnya dari konsep *Mobile Application* [3]. Pada penelitian kali ini digunakan *Ionic Framework* dalam membangun *Mobile Application*.

Ionic framework itu sendiri merupakan sebuah user interface framework yang dibangun dengan *HTML*,*CSS*,dan *JavaScript* yang dapat digunakan oleh *Hybrid mobile Application Development*[4]. Ionic framework itu terdiri dari gabungan dari beberapa teknologi yang bekerja sama untuk membangun *Mobile App* yang lebih cepat dan lebih mudah.

Salah satu layer paling atas dari *Ionic framework* itu adalah ionic itu sendiri yang menyediakan *User Interface Application* dan yang berada dibawahnya adalah Angular (Dikenal dengan AngularJS), sebuah aplikasi yang sangat kuat dalam membangun suatu website.

Framework ini dikombinasikan dengan *Capacitor*, dimana memperbolehkan aplikasi web untuk memanfaatkan kapabilitas perangkat native dan menjadi sebuah aplikasi native. Kombinasi inilah yang membuat *ionic* dapat mengirimkan *platform*

yang kuat untuk membuat Aplikasi *Hybrid*. Seperti yang dijelaskan yang di atas pula terdapat beberapa bahasa pemrograman yang dipakai pula sebagai berikut [5]:

1. *TypeScript*
2. *Angular*
3. *RxJS*
4. *Sass*

2.2 AngularJS

AngularJS atau bisa disebut dengan *Google AngularJS* merupakan sebuah *all-inclusive Javascript Model View Controller* (MVC) *Framework* yang mana sangat mudah dan cepat untuk membangun aplikasi yang berjalan baik pada desktop dan *mobile application* [6]. Angular dapat menggunakan bahasa pemrograman berupa Javascript, namun dalam pengembangan aplikasi mobile disarankan menggunakan TypeScript.

TypeScript adalah *Open-Source Language* yang dibangun di atas *JavaScript*, salah satu alat yang paling banyak digunakan di dunia, dengan menambahkan definisi jenis statis [7]. *TypeScript* menyediakan cara untuk mendeskripsikan bentuk objek, memberikan dokumentasi yang lebih baik, dan memungkinkan TypeScript untuk memvalidasi bahwa kode Anda berfungsi dengan benar.

Alasan lainnya menggunakan *TypeScript* dikarenakan *Ionic CLI* tidak dapat memungkinkan untuk membuat suatu project menggunakan *Javascript* [6].

2.3 Perpustakaan Universitas Multimedia Nusantara

Perpustakaan merupakan kumpulan informasi-informasi yang dikelola oleh seorang professional di bidang informasi (Pustakawan) yang dapat menyediakan akses *intellectual* baik secara fisik maupun digital serta pelayanan dan program yang memiliki misi untuk mengedukasi, menginformasikan atau menghibur berbagai kalangan masyarakat dengan tujuan mestimulasi seorang individu untuk belajar dan memajukan masyarakat secara keseluruhan [8].

Perpustakaan Universitas Multimedia Nusantara itu sendiri memiliki beberapa fitur atau layanan antara lain [9]:

a. Menyediakan buku, jurnal, laporan skripsi, majalah, *textbooks* dan masih banyak lainnya.
b. Menawarkan akses secara gratis ke Netflix.
c. Selama pandemi berlangsung, perpustakaan menyediakan layanan *Book Delivery* atau pengiriman buku ke rumah mahasiswa bagi yang ingin meminjam buku.
d. Menawarkan *Information Literacy course* yang ada pada *platform* UMN *e-learning*.

2.4 NFC (Near Field Communication)

NFC atau *Near Field Communication* adalah sebuah teknologi komunikasi wireless berfrekuensi tinggi dengan jarak yang pendek yang ada pada *convergence of contactless identification* seperti RFID (Radio Frequency Identification) dan alat *networking* lainnya seperti *Bluetooth* dan *Wi-Fi* [10].

Sebuah Komunikasi dapat terjadi pada NFC dengan cara melakukan "*Touching*" ataupun "*Tapping*" ke satu device NFC dan device lain. Komunikasi antar perangkat dilakukan dalam frekuensi tinggi 13.56Mhz yang biasa digunakan pada RFID. Walaupun RFID dapat menangkap dan mentransmisikan dari kejauhan, namun NFC memiliki jarak terbatas.

integrasi teknologi NFC ke dalam handphone dianggap sebagai solusi yang pratical pada kasus di atas dikarenakan hampir semua orang membawa handphone.

Pengembangan RFID untuk menunjukkan teknologi NFC dapat dijelaskan sebagai berikut [11]:

1. *Short Range Communication*, dimana RFID dapat menggunakan aktif tags jarak jauh yang berisi energy yang tersisipkan di dalamnnya
2. *Passive Tag Usage only* (dapat terjadi pada mode *Reader* dan *Author*) dimana aktif dan pasif *tags* dapat terjadi di RFID
3. Mewarisi *Secure DataChange* dikarenakan komunikasi jarak pendek (*Short Range Communication*)

2.5 Lempel Ziv Welch (LZW) Algorithm

Lempel Ziv Welch atau bisa disebut dengan *LZW Algorithm* merupakan sebuah algoritma hasil upgrade dari LZ78 yang di publish pada tahun 1978 oleh A. Lempel, J. Ziv dan Terry A. Welch [12]. Algoritma berbasis *dictionary-based*, dimana menggunakan *index* dan *dictionary* untuk melakukan operasi encoding-decoding terhadap data yang akan digunakan.

Proses kompresi atau dekompresi menggunakan LZW dimulai dengan membuat sebuah *dictionary* dengan nilai asal yang berisikan 256 karakter ASCII dengan indeks 0-255 [13]. Sehingga pada saat melakukan proses kompresi maupun dekompresi, karakter pertama akan selalu ditemukan pada *dictionary*.

```
Step 1:    Dictionary terlebih dahulu diinisialisasi dengan karakter
           ASCII
Step 2:    CurrentChars ← Karakter pertama dari input stream
Step 3:    DictIndex ← 255
Step 4:    while not EOF character do begin
Step 4a:       NextChar ← Karakter selanjutnya dari CurrentChars
Step 4b:       ConcatStr ← CurrentChars + NextChar
Step 4c:       if ConcatStr ada pada Dictionary then begin
Step 4d:             CurrentChars ← ConcatStr
Step 4e:       end else begin
Step 4f:             Output ← Index dari CurrentChars pada
Step 4g:             dictionary
Step 4h:             Isi ke dalam Dictionary (DictIndex, ConcatStr)
Step 4i:             DictIndex ← DictIndex + 1
Step 4j:             CurrentChars ← NextChar
Step 4k:       end
Step 5:    end
Step 6:    EncodedStream ← Output
```

Gambar 2.1 Pseudocode Compress Algoritma LZW

Contoh Kompresi dari Algoritma LZW dengan inputan: BBABBABAB dapat dilihat pada Table 2.1 berikut:

Tabel 2.1 Tabel Kompresi Algoritma LZW

Dictionary Awal	Prefix P	Char C	*Concat* P+C *Dictionary* ?	Output	*Dictionary* Baru
[65]A	"B"	"B"	Tidak	66	[256] BB
[66]B	B	A	Tidak	66	[257]BA
-	A	B	Tidak	65	[258]AB
-	B	B	Ya	-	-
-	BB	A	Tidak	256	[259]BBA
-	A	B	Ya	-	-
-	AB	A	Tidak	258	[260]ABA
-	A	B	Ya	-	-
-	AB	EOF	-	258	-

Maka Hasil Kompresi dari Inputan BBABBABAB merupakan Kumpulan angka dari kolom Output pada Tabel 2.1 yaitu, 66,66,65,256,258,258. Dari hasil output yang didapat pun dapat dilakukan dekompresi kembali dengan menggunakan *Dictionary* pada Table 2.1 di atas. Proses yang dilakukan dengan cara membaca

kumpulan angka output hasil dari kompresi dimana *dictionary* yang ada dijadikan acuan sebagai pembentukkan *string* asli.

```
Step 1:    Dictionary terlebih dahulu diinisialisasi dengan karakter
           ASCII
Step 2:    PreviousCodeWord ← Codeword pertama dari encoded stream
Step 3:    String ← toString(PreviousCodeWord);
Step 4:    Char ← toChar(first input codeword);
Step 5:    DictIndex ← 256;

Step 6:    while NOT EOF encoded stream
Step 6a:       CurrentCodeWord ← codeword selanjutnya pada encoded
               stream
Step 6b:       if CurrentCodeWord ada pada Dictionary then begin
Step 6c:          String ← toString(CurrentCodeWord)
Step 6d:       end else begin
Step 6e:          String ← toString(PreviousCodeWord) + Char
Step 6f:       end;

Step 6g:       Output ← Output + String;
Step 6h:       Char ← Karakter pertama dari string sekarang
Step 6i:       insertToDictionary(DictIndex,
                                  toString(PreviousCodeWord) + Char;
Step 6j:       PreviousCodeWord ← CurrentCodeWord;
Step 6k:       DictIndex++;
Step 7:    end
Step 8:    DecodedStream ← Output
```

Gambar 2.2 Pseudocode LZW Decompressing

Berikut contoh decompress dari output Tabel 2.1 yang berisikan angka 66,66,65,256,258,258 didekompresi dengan algoritma LZW yang dapat dilihat pada Tabel 2.2 berikut:

Tabel 2.2 Tabel Dekompresi Algoritma LZW

Dictionary	*Previous Codeword*	*Current Codeword*	*Output (String)*	*First Char of String*	*Current CodeWord* di dalam *Dictionary*
[65]A	66	-	B	-	-
[65]B	66	66	B	B	Ya
[256]BB	66	65	A	A	Ya
[257]BA	65	256	BB	B	Ya
[258]AB	256	258	AB	A	Ya
[259]BBA	258	258	AB	A	Ya
[260]ABA	-	-	-	-	-

Hasil dekompresi dari Tabel 2.2 adalah kumpulan karakter atau *string* pada kolom output pada Tabel 2.2 yaitu BBABBABAB. Hal ini di dapat dengan menggunakan *Dictionary* yang telah dibuat pada saat kompresi yang mana dapat menerjemahkan atau melakukan dekompresi terhadap kode-kode yang ada untuk membentuk *string* yang ada pada input Tabel 2.1.

2.6 Black Box Testing

Black Box Testing atau bisa disebut dengan *Functional Testing* merupakan sebuah metode uji fungsionalitas dari *software* dimana penguji dari software yang menggunakan *Black Box Testing* tidak diperbolehkan atau tidak memiliki akses untuk melihat kode program yang ada di dalamnnya. *Black Box Testing* tidak

mempedulikan bagaimana mekanisme yang ada di dalam ataupun *code* (*Typescript*) yang dipakai untuk menjalankan aplikasi yang terjadi dari sebuah halaman aplikasi melainkan hasil keluaran dari aplikasi setelah penguji atau pengguna menggunakan input atau kondisi tertentu pada tampilan antarmuka dari aplikasi yang telah dibuat [14].

Dalam testing ini, pengguna diibaratkan seperti melihat "*Big Black Box*" dimana pengguna tidak dapat melihat isi kotak di dalamnnya. Penguji hanya sebatas mengetahui bahwa data dapat dimasukkan ke dalam *Black Box* dan *Black Box* akan merespon berdasarkan data yang dimasukkan tadi. Hal ini dapat terjadi murni dari *requirement specification knowledge;* penguji mengetahui apa yang diharapkan untuk *black box* kirim atau respon dan pengujian untuk memastikan *black box* mengirim apa yang seharusnya dikirim [15].

2.7 Likert Scale

Skala Likert atau bisa disebut dengan *Likert Scale* merupakan teknik pengukuran yang dikemukakan oleh Rensis Likert dimana pengukuran yang dilakukan menggunakan skala berbasis kuesioner. Kuesioner ini menggunakan range skala bernilai 1 hingga skala bernilai 5 yang merepresentasikan jawaban sangat tidak sejutu hingga jawaban sangat setuju [16]. Nilai beserta representasi jawaban pada skala 5 poin likert dapat dilihat pada Tabel 2.3 berikut:

Tabel 2.3 Tabel Skala Likert

Nilai	Representasi Jawaban Berdasarkan Skala
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Netral
4	Setuju
5	Sangat Setuju

Setelah pengumpulan data kuesioner, berikut cara menghitung persentase nilai rata-rata dari jawaban yang dipilih oleh responden pada kuesioner yang dapat dilihat pada persamaan 2.1.

$$\text{Persentase} = \frac{\text{Total Score}}{\text{Nilai dari Skala tertinggi} * \text{Jumlah Responden}} * 100\% \quad (2.1)$$

Untuk menghitung persentase dibutuhkan beberapa data seperti *Total Score* yang dapat dihitung pada persamaan 2.2 berikut:

$$\text{Total Score} = (S1 * 1) + (S2 * 2) + (S3 * 3) + (S4 * 4) + (S5 * 5) \quad (2.2)$$

Berdasarkan persamaan 2.2, S1 yang ditulis merepresentasikan jumlah responden yang menjawab skala dengan nilai 1 atau sangat tidak setuju, S2 yang ditulis merepresentasikan jumlah responden yang menjawab skala dengan nilai 2 atau tidak setuju, S3 yang ditulis merepresentasikan jumlah responden yang menjawab skala dengan nilai 3 atau netral, S4 yang ditulis merepresentasikan jumlah responden yang menjawab skala dengan nilai 4 atau setuju dan terakhir S5 yang ditulis merepresentasikan jumlah responden yang menjawab skala dengan nilai 5 atau sangat setuju.

Setelah menghitung persentase nilai rata-rata dari jawaban yang dipilih responden, dibutuhkan pula interval yang dapat menentukan apakah kuesioner yang diajukan disetujui oleh mayoritas responden atau tidak. Berikut cara menentukan persentase interval dari skor(persentase nilai rata-rata) yang dapat dilihat pada persamaan 2.3:

$$\text{Interval} = \frac{100}{\text{Nilai Skala Tertinggi}} \quad (2.3)$$

Berdasarkan persamaan 2.3, dapat disimpulkan skala 5 poin likert memiliki nilai interval sebesar 20 (Nilai Skala Tertinggi 5poin likert adalah 5) sehingga persentase interval dari skala 5 poin likert dapat ditulis berdasarkan Tabel 2.3 berikut:

Tabel 2.4 Tabel Interval Skala Likert

Nilai	Representasi Jawaban Berdasarkan Skala	Interval
1	Sangat Tidak Setuju	0% - 20,00%
2	Tidak Setuju	20,01% - 40.00%
3	Netral	40.01% - 60,00%
4	Setuju	60,01% - 80,00%
5	Sangat Setuju	80,01% - 100%

2.8 Cronbach Alpha Coefficient

Cronbach Alpha Coefficient atau bisa disebut dengan *Alpha Coefficient Method* merupakan sebuah *method* atau cara menghitung realibilitas yang bisa digunakan dalam *Likert Scale.* Oleh karena itu, tidak terbatas pada nilai berformat *true-false* atau nilai berformat *correct-incorrect* [17].

Cronbach Alpha Coefficient itu sendiri merupakan hasil *weighted standard variations mean* yang di dapat dari membagi total barang yang ada pada skala menjadi *general variance* [18].

$$\alpha = \frac{n}{(n-1)}\left[1 - \frac{\sum_{i=1}^{n}\sigma_{yi}^{2}}{\sigma_{x}^{2}}\right] \quad (2.4)$$

Apabila *item* pada skala telah di standarisasi, *coefficient* dapat dihitung menggunakan *correlation mean* atau *variance-covariances mean*. Berikut contoh rumus dari *correlation mean* dan *variance-covariances mean*:

a. *Correlation Mean*

$$\alpha = \frac{n\,\overline{\rho}}{1 = (n-1)\overline{\rho}} \tag{2.5}$$

b. Variance-Covariances Mean

$$\alpha = \frac{n\,\overline{\sigma xx}/\overline{\sigma x}}{1 + (n-1)\overline{\sigma xx}/\overline{\sigma x}} \tag{2.6}$$

Setelah salah satu formula untuk menghitung Cronbach Alpha Coefficient dengan menggunakan *correlations mean* antara *item* pada skala telah dilakukan, dapat dilihat bahwa jumlah *item* pada skala dan korelasi *mean* antara *item* skala secara proposional memiliki hubungan antara satu dengan yang lain [19]. Apabila korelasi antara *item* pada skala menghasilkan nilai negatif, maka nilai Cronbach Alpha Coefficient pun akan ikut negatif juga.