IMPLEMENTASI QRCODE SEBAGAI TIKET MASUK EVENT DENGAN MEMPERHITUNGKAN TINGKAT KOREKSI KESALAHAN

Edwin Fajar Nurdiansyah [1], Irawan Afrianto [2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-116 Bandung
E-mail : [1] awingawingan@gmail.com, [2] irawan.afrianto@email.unikom.ac.id

ABSTRAK

Tiket merupakan suatu dokumen yang sangat penting dalam suatu acara (*event*) karena tiket menentukan apakah seseorang memiliki akses untuk memasuki acara tersebut atau tidak. Untuk melakukan verifikasi tiket, petugas yang berjaga di pintu masuk harus memastikan bahwa setiap orang yang masuk dapat menunjukan tiket. Kesederhanaan sistem verifikasi tiket ini, banyak disalahgunakan untuk mendapatkan akses masuk ke dalam acara secara cuma-cuma. Tidak adanya proses identifikasi dan validasi tiket dengan data yang dimiliki oleh penyelengara acara membuat kecurangan tetap saja terjadi.

Dari permasalahan yang ada, maka dilakukan penelitian untuk memanfaatkan QR *code* sebagai tiket masuk event dengan memperhitungkan tingkat koreksi kesalahan. Penelitian ini mencoba memperbaharui sistem verifikasi tiket dengan sistem komputerisasi dengan mengganti bentuk fisik tiket menjadi QR *code*. Telah dilakukan pengujian alpha dan pengujian kemampuan pembacaan QR *code* untuk mengetahui apakah tingkat koreksi kesalahan mempengaruhi kemampuan pembacaan QR *code*.

Hasil dari pengujian *alpha* menunjukan bahwa sistem verifikasi tiket yang dibangun bebas dari kesalahan sintaks dan berjalan sesuai dengan fungsinya. Hasil dari pengujian kemampuan pembacaan QR *code* menyatakan bahwa tingkat koreksi kesalahan mempengaruhi kemampuan pembacaan QR *code* untuk QR *code* yang mengalami kerusakan atau bentuknya tidak utuh. Semakin tinggi tingkat koreksi kesalahan, semakin baik kemampuan pembacaan QR *code*.

Kata Kunci : Tiket, *e-Ticketing*, Sistem Verifikasi Tiket, QR *code*, QR *scanner*, Koreksi Kesalahan, *Reed-Solomon.*

1. PENDAHULUAN

Tiket merupakan suatu dokumen yang sangat penting dalam suatu acara (event) karena tiket menentukan apakah seseorang memiliki akses untuk memasuki acara tersebut atau tidak. Pada umumnya tiket berupa kertas yang memiliki ukuran sebesar uang kertas dengan cetakan gambar atau grafik khusus sesuai dengan keinginan penyelenggara acara. Untuk melakukan verifikasi tiket, petugas yang berjaga di pintu masuk harus memastikan bahwa setiap orang yang masuk dapat menunjukan tiket. Apabila tiket yang ditunjukan sesuai, petugas menyobekan sebagian dari tiket tersebut sebagai tanda bahwa tiket tersebut telah digunakan.

Kesederhanaan sistem tiket ini, kini banyak disalahgunakan untuk mendapatkan akses masuk ke dalam acara secara cuma-cuma. Ada yang masuk dengan menggunakan potongan tiket dan berpura-pura telah masuk sebelumnya, bahkan ada yang sampai melakukan pemalsuan tiket. Seperti dilansir dalam tribunnews.com, pada pertandingan Persib melawan Persita Tanggerang ditemukan 4 buah tiket palsu dengan nomor seri yang sama. Tidak adanya proses identifikasi dan validasi tiket dengan data yang dimiliki oleh penyelengara acara membuat kecurangan tetap saja terjadi. Padahal saat ini banyak teknologi informasi yang dapat dimanfaatkan, salah satunya dengan memanfaatkan QR *code* sebagai tiket masuk untuk memperbaharui proses verifikasi.

Berdasarkan masalah yang telah di uraikan, Maksud dari penelitian ini adalah untuk memanfaatkan QR *code* sebagai pengganti tiket masuk event dengan memperhitungkan tingkat koreksi kesalahan.

Adapun tujuan yang ingin dicapai dari Pemanfaatan QR *Code* sebagai Tiket Masuk Event dengan memperhitungkan Tingkat Koreksi Kesalahan diantaranya adalah:

1. Memperbaharui proses verifikasi tiket dengan sistem komputerisasi untuk mengurangi terjadinya berbagai kecurangan pada proses verifikasi tiket.
2. Mengganti fisik tiket yang berupa kertas menjadi QR *code* yang dapat disematkan pada perangkat mobile.
3. Menguji kemampuan pembacaan dari QR *code* pada proses verifikasi tiket dengan tingkat koreksi kesalahan dan kondisi QR *code* yang berbeda-beda.

2. ISI PENELITIAN

2.1Landasan Teori

2.1.1 QR *Code*

Quick Respons *Code* atau biasa disebut QR *code* merupakan salah satu jenis kode matriks yang pertama kali dirancang oleh industri otomotif di Jepang. Sebuah QR *code* menggunakan empat mode standar *encoding* (yaitu numerik, alfanumerik, *byte* / biner dan kanji) untuk menyimpan data secara efisien. Sistem QR *code* pun telah manjadi populer di luar industri otomotif karena pembacaan yang cepat dan kapasitas penyimpanan yang lebih besar dibandingkan dengan standar *barcode* UPC.

Gambar 1. Contoh QR *Code*

QR *code* terdiri dari modul hitam (titik persegi) diatur dalam kotak persegi pada latar belakang putih yang dapat dibaca oleh perangkat pencitraan seperti kamera dan diolah menggunakan koreksi kesalahan *Reed-Solomon* hingga gambar dapat dengan tepat di interpretasikan. Data tersebut kemudian diekstraksi dari pola yang ada dari kedua komponen horizontal dan komponen vertical pada gambar.

2.1.2 *Storage* QR *Code*

Jumlah data yang dapat disimpan pada simbol QR *code* tergantung dari tipe data (mode, atau kumpulan karakter masukan), versi (1, …, 40, mengindikasikan keseluruhan dimensi simbol), dan tingkat kode koreksi kesalahan. Kapasitas penyimpanan maksimum saat ini adalah untuk simbol 40-L (versi 40 dengan koreksi kesalahan tingkat *low*).

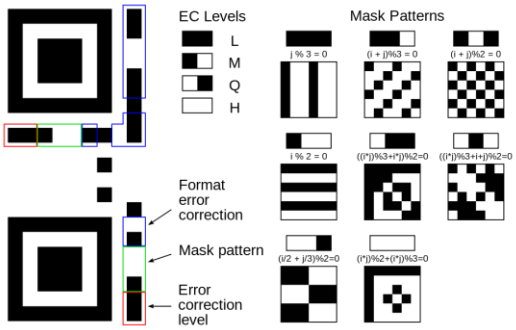
Tabel 1. Kapasitas Maksimum Penyimpanan

Input Mode	**Maks. Karakter**	**Bits/ Karakter**	**Default Encoding**
Numeric only	7.089	31/3	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Alpha-numeric	4.296	51/2	0–9, A–Z (upper-case only), space, $, %, *, +, -, ., /, :
Binary/ byte	2.953	8	ISO 8859-1
Kanji/ kana	1.817	13	Shift JIS X 0208

Setiap versi akan mempunyai ukuran yang berbeda dengan versi lainnya.

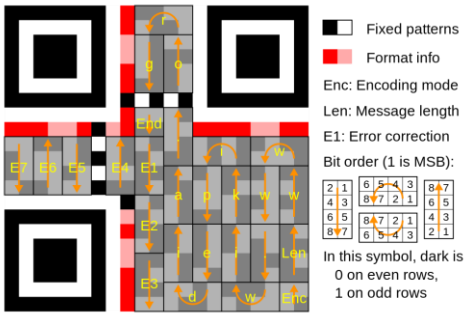
2.1.3 *Encoding* QR *Code*

Informasi format mencatat dua hal: tingkat koreksi kesalahan dan mask pattern yang digunakan untuk simbol. Masking digunakan untuk memecah pola di area data yang mungkin membingungkan scanner, seperti daerah kosong besar atau fitur menyesatkan yang terlihat seperti tanda locator. Mask pattern didefinisikan pada grid yang diulang sediperlukannya untuk menutupi seluruh simbol. Modul yang sesuai dengan daerah gelap mask terbalik. Informasi format dilindungi dari kesalahan dengan kode BCH, dan dua salinan lengkap termasuk dalam setiap simbol QR.



Gambar 2. Maksud Informasi Format

Data pesan ditempatkan dari kanan ke kiri dalam pola zigzag. Dalam simbol yang lebih besar, cukup rumit karena adanya pola keselarasan dan penggunaan beberapa blok interleaved koreksi kesalahan.



Gambar 3. Penempatan Pesan QR *code*

Indikator 4 bit digunakan untuk memilih mode *encoding* dan menyampaikan informasi lainnya. Mode pengkodean dapat dicampur sesuai kebutuhan dalam simbol QR.

Tabel 2. Menu *Encoding*

Indikator	**Keterangan**
0001	*Numeric encoding (10 bits per 3 digits)*
0010	*Alphanumeric encoding (11 bits per 2 characters)*
0100	*Byte encoding (8 bits per character)*
1000	*Kanji encoding (13 bits per character)*
0011	*Structured append (used to split a message across multiple QR symbols)*
0111	*Extended Channel Interpretation (select alternate character set or encoding)*
0101	*FNC1 in first position (see Code 128 for more information)*
1001	*FNC1 in second position*
0000	*End of message*

Setelah setiap indikator memilih mode *encoding*, panjang field lah yang memberitahu berapa banyak karakter dikodekan dalam mode tersebut. Jumlah bit pada panjang field tergantung pada *encoding* dan versi simbol.

Tabel 3. Jumlah Bit Per Panjang Field

Encoding	**Ver.1–9**	**10–26**	**27–40**
Numerik	10	12	14
Alfanumerik	9	11	13
Byte	8	16	16
Kanji	8	10	12

Mode *encoding* alfanumerik menyimpan pesan lebih kompak daripada yang mode byte, tetetapi mode alfanumerik tidak bisa menyimpan huruf-huruf non-kapital dan memiliki pilihan tanda baca terbatas yang cukup untuk alamat *web* yang belum sempurna. Dua karakter dikodekan dalam nilai 11-bit dengan rumus:

$$V = 45 \times C_1 + C_2 \quad \ldots(1)$$

Tabel 4. Kode Karakter Alfanumerik

Kode	**Karakter**	**Kode**	**Karakter**
00	0	23	N
01	1	24	O
02	2	25	P
03	3	26	Q
04	4	27	R
05	5	28	S
06	6	29	T
07	7	30	U
08	8	31	V
09	9	32	W
10	A	33	X
11	B	34	Y
12	C	35	Z
13	D	36	SP
14	E	37	$
15	F	38	%
16	G	39	*
17	H	40	+
18	I	41	–
19	J	42	.
20	K	43	/
21	L	44	:
22	M		

2.1.4 Koreksi Kesalahan

Codeword dengan panjang 8 bit dan menggunakan algoritma koreksi kesalahan *Reed-Solomon* dengan empat tingkat koreksi kesalahan. Semakin tinggi tingkat koreksi kesalahan, kapasitas penyimpanan akan semakin kurang.

Tabel 5. Tabel Tingkat Koreksi Kesalahan

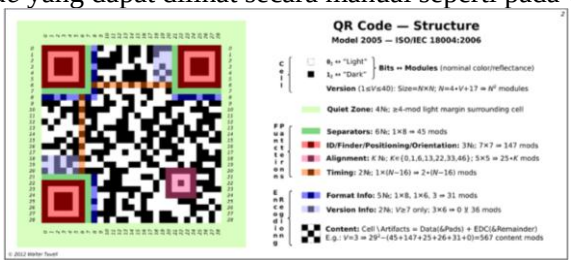
Tingkat EC	Kemampuan Koreksi Kesalahan	Indikator
Level L (Low)	7% dari codewords dapat dikembalikan	01
Level M (Medium)	15% dari codewords dapat dikembalikan	00
Level Q	25% dari	11

(Quartile)	codewords dapat dikembalikan	
Level H (High)	30% dari codewords dapat dikembalikan	10

Dalam simbol QR *code* yang lebih besar, pesan akan dipecah menjadi beberapa blok kode *Reed-Solomon*. Ukuran dipilih sehingga paling banyak 15 kesalahan dapat diperbaiki di setiap blok, ini membatasi kompleksitas dari algoritma decoding. Blok kode tersebut kemudian disisipkan bersama-sama, sehingga lebih kecil kemungkinannya bahwa kerusakan lokal untuk simbol QR akan membanjiri kapasitas setiap blok tunggal. Tingkat Koreksi kesalahan (EC *Level*) dan versi dari QR *code* ini dapat menentukan jumlah data yang dapat ditampung dalam setiap QR *code*.

2.1.5 *Decoding* QR *Code*

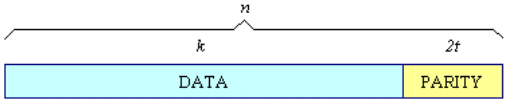
Decoding adalah proses pembacaaan QR *code* untuk menghasilkan informasi dari data yang ada pada QR *code*. *Decoding* merupakan kebalikan dari proses *encoding* yang merupakan proses untuk mengubah data ke dalam bentuk QR *code*. Ada beberapa bagian pada QR *code* yang dapat dilihat secara manual seperti pada Gambar 4.



Gambar 4. Lokasi Area QR *Code*

2.1.6 *Reed-Solomon Error Correction Code*

Reed-Solomon code adalah kode siklik nonbiner yang terbuat dari 2n bit biner dimana m lebih besar daripada 2. Diciptakan oleh Irving S. Reed dan Gustave Solomon. Mereka menjelaskan secara sistematis kode bangunan yang dapat mendeteksi dan memperbaiki beberapa kesalahan simbol acak. Untuk sebuah *codeword*, panjang kode adalah 8 bit, maka *Reed-Solomon code*: $28 - 1 = 255$. Untuk mengkoreksi kesalahan pada *codeword*, maka ditambahkan *Reed-Solomon code* agar dapat terlindung dari kerusakan tanpa harus kehilangan data. Kemampuan koreksi *error*-nya bergantung pada jumlah data yang dikodekan. *Reed-Solomon code* terdiri atas 2 bagian, yaitu bagian data dan bagian paritas. *Reed-Solomon code* dinyatakan dengan kode (n,k) atau RS(n,k) dimana n adalah maksimum *codeword*, yaitu 255, sedangkan k adalah jumlah dari *codeword* data. Berikut ini merupakan penjelasan dari gambar 5 mengenai struktur *Reed-Solomon code*.



Gambar 5. Struktur *Reed-Solomon Code*

2t atau simbol paritas adalah *codeword* yang digunakan untuk koreksi kesalahan dengan nilai maksimun adalah t. Sebagai contoh, RS (255,223) artinya terdapat total 255 *codeword* yang terdiri atas 223 *codeword* data dan 32 *codeword* paritas.

n = 255, k = 223

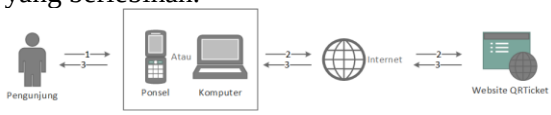
2t = (255-223) → t = 16

sehingga kesalahan yang dapat diperbaiki adalah sebanyak 16 *codeword*. Sebagai informasi 1 *codeword* adalah 8 bit, sehingga total koreksi yang dapat diperbaiki adalah 16 x 8 bit = 128 bit.

Ketika panjang *Reed-Solomon code* kurang dari 28 – 1, maka *padding* 0 digunakan untuk membuat *Reed-Solomon code* tepat 28 – 1. Sewaktu proses pembacaan kode, *padding* 0 akan dibuang. Hal ini disebut dengan penyingkatan *Reed-Solomon code*. Sebagai contoh, misalkan terdapat 100 *codeword* data untuk mengkoreksi 8 buah kesalahan, sehingga akan dibutuhkan tambahan 16 *codeword* paritas. Jadi total keseluruhan dari *codeword* menjadi 116, yang mana masih kurang dari 28 – 1, sehingga harus ditambahkan 139 *codeword padding* 0.

2.2 Analisis Sistem

Untuk mengganti fisik sebuah tiket menjadi QR *code*, diperlukan aplikasi yang terdiri dari *server* yang juga berfungsi sebagai *encoder* dan *scanner* yang berfungsi juga sebagai *decoder*. Pada penelitian ini, *encoder* dibangun pada flatform website karena tiket yang dikelola merupakan tiket yang dijual secara *presale* (dijual pada hari-hari sebelum dilaksanakannya *event*) sehingga membutuhkan sebuah *website* untuk melakukan penjualan secara *online*. Penjualan tiket dilakukan secara *online* untuk menghindari antrian yang berlebihan.



Gambar 6. Gambaran Umum Proses Pemesanan Tiket

Keterangan:

1) Pengunjung mengakses *website*/*server* menggunakan ponsel atau pun komputer melalui *internet* dan melakukan pemesanan tiket dengan mengisi form pemesanan dengan data yang valid.
2) Data pemesanan tiket dikirim ke *server* melalui *internet*.
3) *Server* akan memberikan kode registrasi untuk digunakan ketika mengirimkan nomor resi pembayaran.

Karena penelitian ini tidak membahas pembayaran tiket, maka diasumsikan pembayaran dilakukan dengan cara sebagai berikut.

1) Pengunjung melakukan transfer ke rekening penyelenggara acara.
2) Pengunjung mengakses *website* dan menginputkan nomor registrasi pada Form Pembayaran sebagai autentifikasi pengunjung.
3) Jika nomor registrasi valid, pengunjung mengirimkan bukti transfer atau nomor resi pembayaran melalui Form Pembayaran.
4) Setelah pengunjung mengirimkan nomor resi pembayaran, sistem akan melakukan encoding QR *code* dengan status "Non-aktif" dan pengunjung pun dapat mengunduh tiket dalam bentuk QR *code*.
5) Administrator melakukan verifikasi secara manual untuk menentukan validitas dari bukti transfer yang dikirimkan pengunjung. Apabila valid, maka administrator akan

melakukan update pada status pembayaran menjadi lunas, sehingga status tiket pun akan ikut terupdate menjadi "Aktif".

Perlu diketahui, tiket QR *code* yang ada pada sistem ini memiliki tiga jenis status yang berbeda. Hanya tiket yang berstatus "Aktif" yang dianggap valid oleh sistem ketika proses verifikasi tiket.

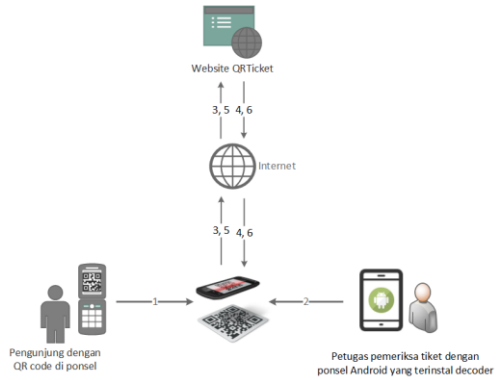
Tabel 6. Keterangan Status Tiket

No	Status Tiket	Keterangan
1	Non-Aktif	Ketika pertama kali dibangkitkan (dimana user telah mengirimkan nomor resi pembayaran)
2	Aktif	Ketika administrator mengubah status pembayaran menjadi lunas (nomor resi pembayaran terbukti valid)
3	Terverifikasi	Ketika petugas pemeriksa tiket melakukan verifikasi terhadap tiket dengan status "Aktif". Tiket dengan status "Terverifikasi" menandakan bahwa tiket tersebut telah digunakan sehingga dalam sistem tiket tersebut tidaklah valid

Selain proses *encoding* QR *code* dan pemesanan tiket, *website* ini juga berfungsi sebagai *server* yang menyimpan dan mengelola data acara, data pemesanan dan data tiket. Fungsional pada pengelolaan data ini berupa tambah, edit dan hapus data. Data pemesanan yang dikelola adalah status pembayaran dimana status pembayaran ini juga mempengaruhi status dari tiket yang dibeli. Seluruh pengelolaan ini dikelola oleh administrator.

Decoder pada sistem ini berupa aplikasi *scanner* yang dibangun pada platform Android. Nantinya, aplikasi *scanner* ini harus terinstal pada telepon seluler bersistem operasi Android yang memiliki kamera serta dapat terhubung dengan jaringan internet. Hal ini diperlukan untuk melakukan proses verifikasi secara langsung dan mencocokan data tiket dari data hasil decoding QR *code* dengan data tiket pada *server*.

Sebelum melakukan verifiaksi tiket, petugas pemeriksa tiket harus memilih event terlebih dahulu sesuai dengan *event* yang berlangsung dengan memasukan kode *event* yang petugas dapatkan dari administrator atau koordinator acara. Apabila tiket yang dimiliki pengunjung sesuai dengan event tersebut dan memiliki status "Aktif", maka proses verifikasi tiket ini akan menggati status tiket menjadi "Terverifikasi".



Gambar 7. Gambaran Umum Proses Verifikasi Tiket

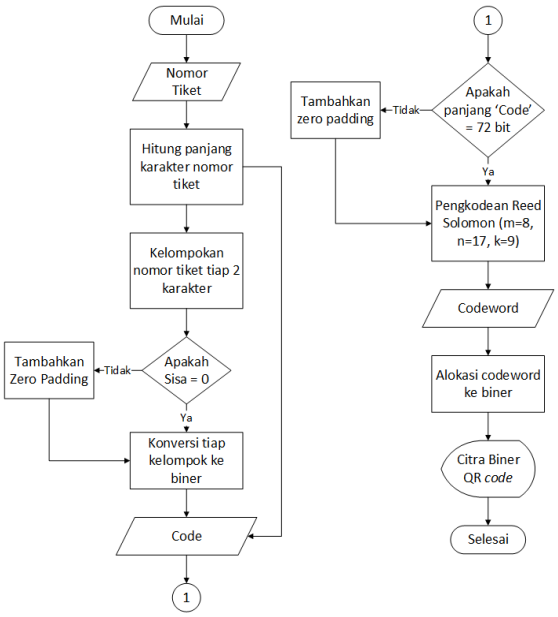
Keterangan:

1) Pengunjung memperlihatkan tiket QR *code* yang ada pada ponselnya kepada petugas penjaga pintu masuk.
2) Petugas memilih *event* sesuai dengan *event* yang berlangsung saat itu. Kemudian petugas melukakan verifikasi tiket menggunakan aplikasi *decoder* yang terinstal pada perangkat Androidnya.
3) Jika data pada QR *code* terbaca, *decoder* akan mengirimkan data dari hasil *decoding* tersebut melalui *internet*. Kemudian *server* melakukan verifikasi terhadap data tersebut.
4) Server mengirimkan hasil validasi dari nomor tiket tersebut ke *decoder*. Apabila tiket valid (nomor tiket terdaftar, nomor *event* sesuai dengan *event* yang berlangsung dan status tiket "Aktif") maka tiket dapat diproses.
5) Petugas pemeriksa tiket menekan tombol verifikasi. Verifiaksi disini, ialah mengganti status tiket menjadi "Terverifikasi". Decoder akan meminta *server* untuk mengganti status tiket yang "Aktif" tersebut menjadi "Terverifikasi" sebagai tanda bahwa tiket telah digunakan.
6) Jika update berhasil, *server* akan mengirimkan pesan ke *decoder* bahwa tiket telah terverifikasi.

2.2.1 Analisis *Encoding* QR *Code*

Encoding QR *code* merupakan proses mengkodekan data pada QR *code*. Pada penelitian ini, data yang dikodekan berupa nomor tiket. Misalkan ambil contoh nomor tiket adalah "BMT00384" yang terdiri dari 8 digit dan QR *code* yang digunakan adalah versi 1 dengan koreksi kesalahan tingkat High (1-H) dimana 30% dari jumlah *codewords* data dapat dikembalikan. Ini semua berkat adanya kode koreksi kesalahan *Reed-Solomon*.

Pada proses *encoding* QR *code*, data yang akan dikodekan akan dibentuk menjadi *codewords* data. Kemudian kode koreksi kesalahan *Reed-Solomon* akan membentuk suatu *codewords* baru. *Codewords* ini akan menyimpan data yang sama dengan data yang ada pada *codewords* data yang berfungsi sebagai cadangan apabila data pada *codewords* data tidak terbaca oleh pemindai.



Gambar 8. Diagram Alir *Encoding* QR *Code*

Langkah-langkah yang dilakukan saat proses *encoding* data menjadi *codewords* dapat juga dituliskan dengan singkat sebagai berikut.

1) Perhitungan jumlah karakter serta pemberian indikator mode.
2) *Encoding* data dalam representasi biner.
3) Pemenuhan jumlah bit dan simbol sesuai dengan jumlah yang ditentukan
4) Menghitung *codeword* kode koreksi kesalahan *Reed-Solomon*.

2.2.1.1 Perhitungan Jumlah Karakter

Indikator mode dibuat dengan panjang 4 bit sebagai representasi biner seperti pada Tabel 2. Data yang dikodekan adalah "BMT00384", itu berarti data yang digunakan adalah alfanumerik. Maka indikator mode yang dipilih sesuai adalah **0010**.

Indikator jumlah karakter adalah jumlah karakter yang dapat disimpan dalam setiap mode. Seperti pada Tabel 3 untuk data alfanumerik, panjang representasi binernya adalah 9 bit. Panjang contoh nomor tiket adalah 8 bit, maka 8 bit direpresentasikan dalam biner adalah **000001000** sehingga bilangan biner yang sudah terbentuk adalah 0010 **000001000**.

2.2.1.2 Encoding Data dalam Representasi Biner

Dalam mode alfanumerik, setiap karakter di konversikan ke dalam nilai sesuai dengan Tabel 4. Data dibagi dalam kelompok-kelompok per 2 karakter dimana nilai dari karakter pertama dikali 45 hasilnya ditambah dengan nilai karakter kedua. Seperti dijelaskan pada rumus (1). Kemudian nilai dari V dikodekan dalam representasi biner sepanjang 11 bit. Tabel 7 merupakan hasil penguraian dari nomor tiket "BMT00384".

Tabel 7. Encoding Data dalam Representasi Biner

"BM"	**"T0"**	**"03"**	**"84"**
45*11+22	45*29+0	45*0+3	45*8+4

517	553	3	93
01000000101	10100011001	00000000011	00101101100

Lalu lakukan terminator atau penambahkan biner 0000 kepada hasil representasi biner. Akan tetapi jika panjang data yang di *encode* penuh sesuai dengan versi dan tingkat koreksi kesalahan, maka tidak perlu dilakukan terminator sehingga didapat deret bit 0010 000001000 01000000101 10100011001 00000000011 00101101100 **0000.**

2.2.1.3 Pemenuhan Jumlah Bit dan Simbol

Hasil representasi data nomor tiket 0010 000001000 01000000101 10100011001 00000000011 00101101100 0000 kemudian dikelompokan setiap 8bit. Sehingga hasil representasi tersebut menjadi:

00100000 01000010 00000101 10100011 00100000 00001100 10110110 00000

Jika terdapat kelompok dengan panjang data kurang dari 8 bit, maka tambahkan 0 sampai menjadi 8 bit. Penambahan "0" inilah yang dinamakan *zero padding*. Penambahan *zero padding* tersebut diperlukan karena jumlah simbol yang menjadi masukan *Reed-Solomon* harus tetap yaitu 8 bit disebut dengan *codeowrds*. Setelah dilakukan *zero padding*, maka kelompok atau deret *codewords* tersebut menjadi:

00100000 01000010 00000101 10100011 00100000 00001100 10110110 00000000

Jika jumlah *codewords* kurang dari kapasitas simbol, maka tambahkan byte "11101100" dan "00010001" secara bergantian hingga panjang bit memenuhi kapasitas maksimum yang telah ditentukan. Byte tersebut masing-masing ekuivalen dengan 236 dan 17. Kedua *byte* tersebut secara khusus diperlukan oleh spesifikasi QR *code* untuk ditambahkan jika bit string terlalu pendek pada tahap ini.

Untuk kapasitas maksimum QR *code* versi 1 dengan tingkat koreksi kesalahan *High*, didapat bahwa QR *code* 1-H memiliki kapasitas maksimum 9 *codewords*. Jika dikalikan dengan 8 bit maka representasi biner tersebut harus berjumlah 72 bit.

1: 00100000 5: 00100000
2: 01000010 6: 00001100
3: 00000101 7: 10110110
4: 10100011 8: 00000000

Setelah dihitung, jumlah *codewords* pada representasi biner berjumlah 8 *codewords* dengan jumlah bit sebesar 64 bit sehingga perlu dilakukan penambahan *byte* "11101100". Maka kapasitas dari representasi biner tersebut telah berjumlah 9 *codewords*.

1: 00100000 6: 00001100
2: 01000010 7: 10110110
3: 00000101 8: 00000000
4: 10100011 9: 11101100
5: 00100000

Kemudian *codewords* tersebut dikonversi kembali kedalam bilangan desimal sehingga diperoleh deret desimal **32 66 5 163 32 12 182 0 236**. Deret desimal tersebut nantinya akan menjadi masukan dalam proses pengkodean *Reed-Solomon*.

2.2.1.4 Menghitung Codewords Kode Koreksi Kesalahan

Pertama hasil *encoding* data yaitu **32 66 5 163 32 12 182 0 236** dipisah sesuai ketentuan dari blok RS pada. Kemudian pilih rumus g(x) berdasarkan jumlah *Error-*

Correction codewords per blok dari sesuai dengan versi QR *code* dan tingkat koreksi kesalahan.

1) Langkah Divisi Polinomial

Langkah pertama untuk divisi ini adalah mempersiapkan polinomial pesan untuk divisi. Polinom pesan lengkap adalah:

32x8 + 66x7 + 5x6 + 163x5 + 32x4 + 12x3 + 182x2 + 0x1 + 23 …(2)

Untuk memastikan bahwa eksponen dari variabel pertama tidak menjadi terlalu kecil selama divisi, kalikan polinomial pesan dengan xn dimana n adalah jumlah *codewords* koreksi kesalahan yang diperlukan. Dalam hal ini n adalah 17, untuk 17 *codewords* koreksi kesalahan, kalikan polinomial pesan oleh x17, maka akan mendapatkan hasil:

32x25 + 66x24 + 5x23 + 163x22 + 32x21 + 12x20 + 182x19 + 0x18 + 236x17 …(3)

Istilah utama dari generator polinomial juga harus memiliki eksponen yang sama, jadi kalikan dengan x8 untuk mendapatkan

α0x25 + α43x24 + α139x23 α206x22 + α78x21 α43x20 + α239x19 + α123x18 + α206x17 + α214x16 + α147x15 α24x14 + α99x13 α150x12 + α39x11 α243x10 + α163x9 + α136x8 …(4)

Sekarang sudah mungkin untuk melakukan langkah-langkah pembagian berulang. Jumlah langkah di divisi harus sama dengan jumlah variabel dalam polinomial pesan. Dalam hal ini, divisi ini akan mengambil 9 langkah untuk menyelesaikan. Hal ini akan menghasilkan sisa yang memiliki 17 variabel. Istilah-istilah ini akan menjadi 17 *codewords* koreksi kesalahan yang diperlukan.

2) Langkah 1a: Kalikan Generator Polinomial dengan Variabel Pertama pada Polinomial Pesan

Langkah pertama adalah untuk memperbanyak generator polinomial dengan variabel pertama dari polinomial pesan. Dalam studi kasus ini, variabel pertama adalah 32x25. Karena notasi α membuatnya lebih mudah untuk melakukan perkalian, dianjurkan untuk mengkonversi 32x25 notasi α. Untuk nilai integer 32, eksponen α adalah 5 . Oleh karena 32 = α5 . Kalikan generator polinomial oleh α5 :

(α5 * α0) x25 + (α 5* α43) x24 + (α5 * α139) x23 + (α5 * α206) x22 + (α5 * α78) x21 + (α5 * α43) x20 + (α5 * α239) x19 + (α5 * α123) x18 + (α5 * α206) x17 + (α5 * α214) x16 + (α5 * α147) x15 + (α5 * α24) x14 + (α5 * α99) x13 + (α5 * α150) x12 + (α5 * α39) x11 + (α5 * α243) x10 + (α5 * α163) x9 + (α5 * α136) x8 …(5)

Eksponen dari α ditambahkan bersama-sama. Hasilnya adalah:

α5x25 + α48x24 + α144x23 + α211x22 + α83x21 + α48x20 + α244x19 + α128x18 + α211x17 + α219x16 + α152x15 + α29x14 + α104x13 + α155x12 + α44x11 α248x10 + α168x9 + α141x8…(6)

Sekarang , konversikan kembali ke notasi integer:

32x25 + 70x24 + 168x23 + 178x22 + 187x21 + 70x20 + 250x19 + 133x18 + 178x17 + 86x16 + 73x15 + 48x14 + 13x13 + 114x12 + 238x11 + 27x10 + 252x9 + 21x8 …(7)

3) Langkah 1b: XOR hasil dengan polinomial pesan

Karena ini adalah langkah divisi pertama, lakukan operasi XOR hasil dari 1a dengan polinomial pesan.

(32 ⊕ 32) x25 + (66 ⊕ 70) x24 + (5 ⊕ 168) x23 + (163 ⊕ 178) x22 + (32 ⊕ 187) x21 + (12 ⊕ 70) x20 + (182 ⊕ 250) x19 + (0 ⊕ 133) x18 + (236 ⊕ 178) x17 + (0 ⊕ 86) x16 + (0 ⊕ 73) x15 + (0 ⊕ 48) x14 + (0 ⊕ 13) x13 + (0 ⊕ 114) x12 + (0 ⊕ 238) x11 + (0 ⊕ 27) x10 + (0 ⊕ 252) x9 + (0 ⊕ 21) x8 …(8)

Hasilnya adalah :

0x25 + 4x24 + 17x22 + 173x23 + 155x21 + 74x20 + 76x19 + 133x18 + 94x17 + 86x16 + 73x15 + 48x14 + 13x13 + 114x12 + 238x11 + 27x10 + 252x9 + 21x8 …(9)

Buang variabel depan 0 untuk mendapatkan:

4x24 + 17x22 + 173x23 + 155x21 + 74x20 + 76x19 + 133x18 + 94x17 + 86x16 + 73x15 + 48x14 + 13x13 + 114x12 + 238x11 + 27x10 + 252x9 + 21x8 …(10)

4) Langkah 2 Lakukan Iterasi Hingga Akhir

Setelah mendapatkan hasil XOR pada langkah 1b, lakukan kembali iterasi seperti pada langkah 1a dan 1b hingga mendapatkan hasil:

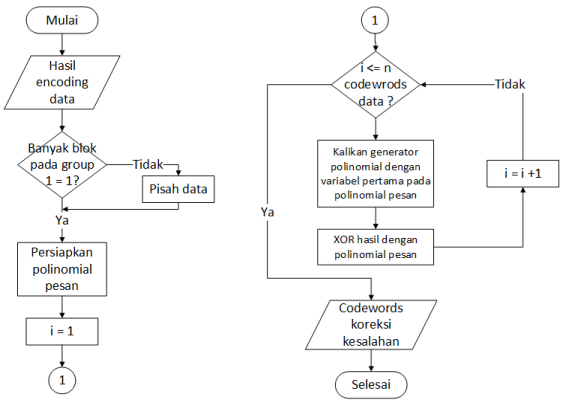
76x16 + 27x15 + 190x14 + 249x13 + 106x12 + 237x11 + 203x10 + 70x9 + 90x8 + 167x7 + 150x6 + 71x5 + 99x4 + 75x3 + 32x2 + 230x1 + 106

5) Menggunakan variabel sisanya sebagai *codewords* koreksi kesalahan

Divisi ini telah dilakukan 9 kali, yang merupakan jumlah variabel dalam polinomial pesan. Ini berarti bahwa iterasi selesai dan ketentuan polinomial di atas adalah *codewords* koreksi kesalahan yang digunakan untuk pesan polinomial asli:

76 27 190 249 106 237 203 70 90 167 150 71 99 75 32 230 106

Seluruh proses menghitung *codewors* koreksi kesalahan dapat juga dilihat dalam bentuk diagram alir.



Gambar 9. Diagram Alir Menghitung *Codewords* Koreksi Kesalahan

2.2.1.5 Alokasi *Codewords* ke Biner

Pada QR code, 1 modul berarti 1 bit. Data hasil di bagian sebelumnya dikodekan ke representasi biner, kemudian data tersebut dialokasikan. Selain itu, ketika nomor blok RS adalah 2 atau lebih tinggi, maka data harus dialokasikan dalam *interleaved*. Dalam contoh data, untuk 1-H (QR code versi 1 dengan tingkat koreksi kesalahan H) memiliki 1 RS blok, data tidak perlu dialokasikan dalam *interleave*. Ada beberapa aturan untuk alokasi data, diantaranya:

1) Pengalokasian memiliki koordinat, i menunjukan baris dan j menunjukan kolom. Pojok kiri atas, memiliki koordinat (0,0). Untuk QR *code* versi 1 ini, memiliki modul dari (0,0) hingga (20,20).
2) Awal modul adalah pojok kanan bawah. Pada QR *code* 1-H ini, awal modul adalah (20,20).
3) Setiap *codewords* terdiri dari 8 bit, sehingga pengalokasian pun dikelompokan menjadi 8 modul dengan ukuran modul 2x4.

4) Pola pengalokasian data dimulai dari bawah keatas kemudian dari atas kebawah, terus berlanjut hingga akhir.
5) Ketika alokasi data sedang mengarah keatas, maka data pun akan dialokasikan secara *zig-zag* ke arah atas. Mulai dari (20,20), kemudian berlanjut ke (19,20), (19,19), (20,19) hingga (20,17) kemudian dilanjutkan ke codeword berikutnya.
6) Ketika alokasi data sedang mengarah kebawah, maka data pun akan dialokasikan ke arah bawah, tetap dengan pola *zig-zag*.

Setelah didapatkan desimal *codewords* data dan *codewords* koreksi kesalahan dari bagian sebelumnya maka tambahkan *codewords* koreksi kesalahan ke codewords data sehingga didapatkan:

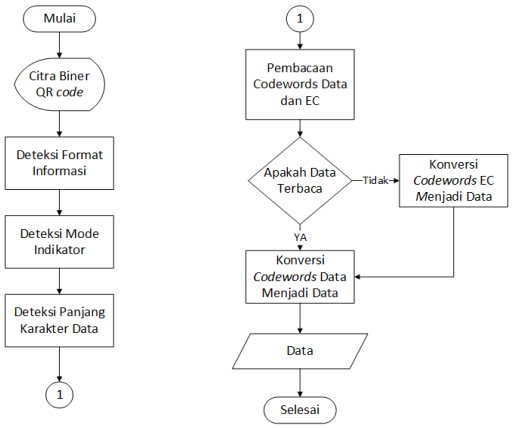
32 66 5 163 32 12 182 0 236 76 27 190 249 106 237 203 70 90 167 150 71 99 75 32 230 106

Kemudian konversikan kembali *codewords* ke biner dan lakukan operasi XOR. Kemudian deretan bit-bit *codewords* data tersebut dialokasikan seperti yang telah dijelaskan sebelumnya untuk mengisi modul-modul yang ada pada QR *code*. Posisi *codewords* koreksi kesalahan akan berada tepat setelah *codewords* data.

Gambar 10. QR Code Hasil Encoding Data “BMT00384”

2.2.2 Analisis *Decoding* QR *Code*

Sistem *decoder* yang dibangun berbasis *mobile* untuk mempermudah mobilitas pemeriksa tiket dalam memeriksa tiketnya. *Decoder* ini akan memanfaatkan kamera yang ada pada perangkat mobile untuk menangkap QR *code*. Kemudian QR *code* tersebut akan di *decode* untuk mengetahui isi data nomor tiket yang ada pada QR *code*.



Gambar 11. Diagram Alir *Decoding* QR *Code*

Proses *decoding* akan memanfaatkan *library* dari ZXing sehingga *decoder* akan melakukan pemanggilan class yang telah disertakan dari ZXing. Apabila QR *code* berhasil di decode maka, *decoder* akan menampilkan data hasil decoding tersebut dan

kemudian mengirimkan data tersebut ke *server* untuk kemudian dilakukan verifikasi tiket.

Langkah-langkah yang dilakukan pada proses decoding QR *code* menjadi data dapat juga dituliskan dengan singkat sebgai berikut.

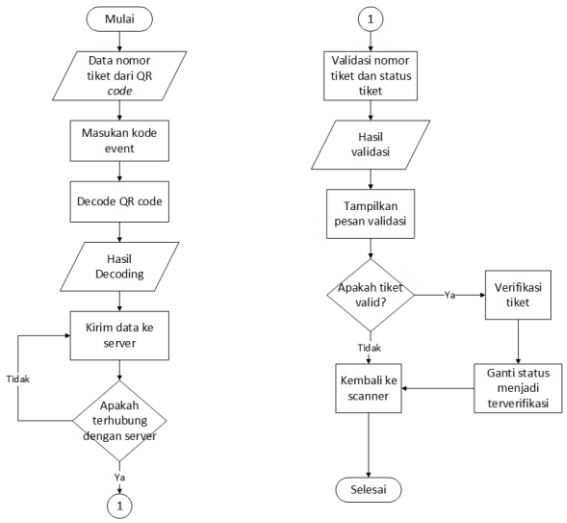
1) Deteksi format informasi, mode indikator dan panjang karakter data.
2) Pembacaan isi pada QR *code* yang berupa *codewords*.
3) Konversi kembali *codewords* data menjadi data.

2.2.3 Pembacaan Tiket

Sebelum melakukan validasi tiket, petugas pemeriksa tiket harus memasukan kode event terlebih dahulu untuk memastikan bahwa tiket yang diverifikasi merupakan tiket event yang bersesuai dengan *event* yang berlangsung.

Setelah *event* yang dipilih valid, maka dilakukan *decoding* pada tiket QR *code* tersebut. Setelah itu, didapatkan hasil *decoding* yang berupa data yang seharusnya adalah nomor tiket. Kemudian data tersebut dikirimkan ke *server* melaui *internet*. Setelah itu *server* akan mencocokan kode event, nomor tiket dan status tiket dengan data yang ada pada *database*. Hanya tiket yang memiliki status Aktif lah yang dinyatakan valid oleh sistem.

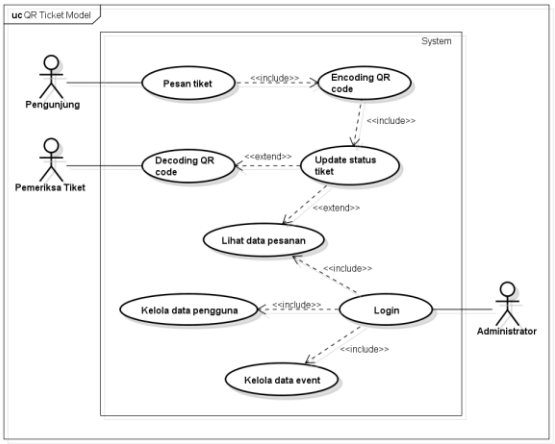
Setelah *server* melakukan validasi dengan database, maka *server* akan mengirimkan hasil validasi tersebut. Jika data tersebut merupakan nomor tiket yang valid, maka petugas pemeriksa tiket dapat melakukan verifikasi tiket. Perintah verifikasi tersebut dikirimkan ke *server* dan kemudian *server* akan mengganti status tiket dari yang asalnya Aktif menjadi Terverifikasi sehingga tiket tersebut kini tidak lagi valid.



Gambar 12. Diagram Alur Proses Pembacaan Tiket

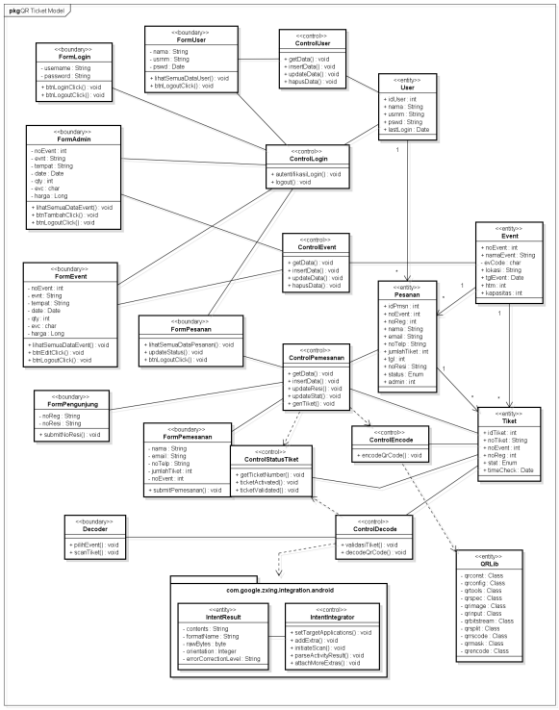
2.3Analisis Kebutuhan Fungsional

2.3.1 *Use Case Diagram*



Gambar 15. *Use Case Diagram*

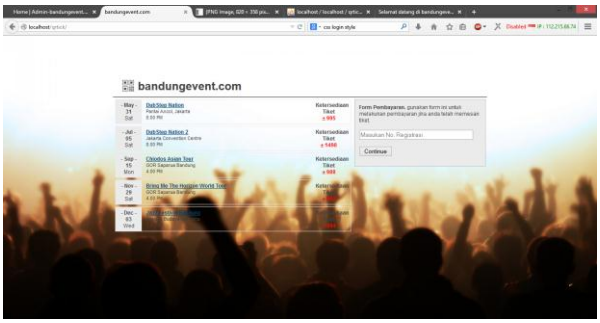
2.3.2 Class Diagram



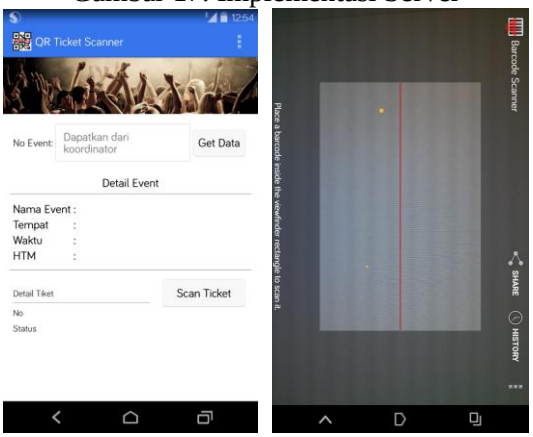
Gambar 16. *Class Diagram*

2.4Implementasi

Implementasi dari sistem dibagi menjadi *server* berbasis *website* dan *decoder* berbasis *mobile*.



Gambar 17. Implementasi Server



Gambar 18. Implementasi *Decoder*

2.5Hasil Pengujian

Pengujian dilakukan 2 media QR *code* berbeda dengan beberapa konsdisi, dari kondisi lingkungan dan tingkat kerusakan untuk membandingkan kemampuan pembacaan QR *code* dari 4 tingkat koreksi kesalahan yang berbeda.

Tabel 8. Hasil Pengujian QR *Code* Utuh

QR Code Ditampilkan pada Layar Ponsel			
Kondisi Pengujian	**Tingkat Koreksi Kesalahan**	**Jumlah QR Code Terbaca / Jumlah Sampel**	**Persentase QR Code Terbaca**
Cukup Cahaya	*Low*	20/20	100%
	Medium	20/20	100%
	Quartile	20/20	100%
	High	20/20	100%
Kurang Cahaya	*Low*	20/20	100%
	Medium	20/20	100%
	Quartile	20/20	100%

	High	20/20	100%
Rata-Rata		160/160	**100%**
QR Code Dicetak pada Kertas			
Kondisi Pengujian	**Tingkat Koreksi Kesalahan**	**Jumlah QR Code Terbaca / Jumlah Sampel**	**Persentase QR Code Terbaca**
Cukup Cahaya	*Low*	20/20	100%
	Medium	20/20	100%
	Quartile	20/20	100%
	High	20/20	100%
Kurang Cahaya	*Low*	20/20	100%
	Medium	20/20	100%
	Quartile	20/20	100%
	High	20/20	100%
Rata-Rata		160/160	**100%**

Tabel 9. Hasil Pengujian QR *Code* Tidak Utuh

QR Code Dicetak pada Kertas			
Kondisi Pengujian	**Tingkat Koreksi Kesalahan**	**Tingkat Kerusakan QR Code**	**Jumlah QR Code Terbaca / Jumlah Sampel**
Cukup Cahaya	*Low*	5%	5/5
		10%	0/5
	Medium	5%	5/5
		10%	1/5
	Quartile	5%	5/5
		10%	2/5
	High	5%	5/5
		10%	5/5
Kurang Cahaya	*Low*	5%	5/5
		10%	0/5
	Medium	5%	5/5
		10%	1/5
	Quartile	5%	5/5
		10%	2/5

	High	5%	5/5
		10%	5/5
Rata-Rata (Total QR code terbaca / Jumlah sampel)			**56/80**

3. PENUTUP

3.1Kesimpulan

Setelah melakukan analisis, perancangan, implementasi dan pengujian, maka dapat diperoleh kesimpulan sebagai berikut.

1. Verifikasi tiket yang diperbaharui dengan sistem komputerisasi dapat meningkatkan kualitas dari verifikasi tiket. Berdasarkan hasil pengujian, hanya tiket dengan kode event yang sesuai dengan yang sedang berlangsung dan memiliki status Aktif yang dapat diverifikasi oleh sistem.
2. Berhasil dibangun suatu sistem untuk menggantikan bentuk fisik tiket menjadi QR *code* untuk memudahkan proses distribusi tiket secara *presale* dan membuat tiket tersebut lebih akurat dalam proses verifikasi.
3. Tingkat koreksi kesalahan *Reed-Solomon error correction code* yang ada pada QR *code* ternyata memang mempengaruhi kemapuan baca dari QR *code*. Akan tetepi pengaruh dari tingkat koreksi kesalahan ini sangat terasa pada tingkat koreksi kesalahan *High* dan ketika ada tiket QR code yang mengalami kerusakan dengan toleransi tingkat kerusakan fisik sekitar 10%.

3.2Saran

Agar pemanfaatan QR *code* sebagai tiket masuk *event* ini semakin baik untuk kedepannya, maka ada beberapa saran yang perlu dilakukan dalam penelitian berikutnya, diantaranya:

1. Ketika QR *code* akan digunakan pada lingkungan yang rawan kerusakan akibat lingkungan atau media QR *code* itu sendiri, sebaiknya gunakan tingkat koreksi kesalahan *High* karena dapat mengembalikan *codewords* hingga 30%.
2. Sistem yang dibangun, khususnya *encoder* dibatasi pada tiket yang dijual *presale* dan sistem ini tidak mengelola pembayaran. Pembayaran disini hanya sebagai simulasi untuk menentukan status tiket. Jadi, untuk kedepannya disarankan agar menambahkan fungsionalitas pembayaran sehingga pengunjung akan lebih dimudahkan dalam proses pembelian tiket.
3. Jika melihat batasan akan tiket *presale*, disarankan untuk menambahkan fungsional mencetak tiket bagi pihak *Event Organizer* sehingga tiket yang dijual *on-the-spot* pun sudah dalam bentuk QR *code* walaupun tercetak dalam kertas. Dengan itu, sistem ini pun akan memiliki fungsionalitas yang lebih lengkap sehingga akan lebih handal untuk menggantikan sistem tiket konvensional.

DAFTAR PUSTAKA

[1] Sankara, A. 2012. *QR Codes and Security Solutions*. International Journal of Computer Science and Telecommunications, Vol. 3, Issue 7.

[2] Septirasyahyani dan Usman, Koredianto. 2012. *Desain dan Implementasi Qr Code Berbasiskan Pengolahan Citra Digital untuk Sistem Parkir di IT Telkom*. Jurnal Informatika. *Retrived from*: http://kru.blog.ittelkom.ac.id/blog/files/downloads/2012/12/JURNAL_SEPTI_KRU_LDN_U_SNAKOM2012.pdf

[3] Zhang, Mu dan Yao, Dan. 2012. *The Application and Design of QR Code in Scenic Spot's eTicketing System*. International Journal of Science and Technology, vol. 2, ISSN 2224-3577.

[4] Swetake. *"How to create QRcode"* (online). http://www.swetake.com/qrcode/qr1_en.html (30 Apr 2014)

[5] Thonky. *"Error Correction Coding"* (online). http://www.thonky.com/qr-code-tutorial/error-correction-coding (4 Jun 2014)