Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)
Edisi. .. Volume. ... Bulan 20.. ISSN : 2089-9033

PENGEMBANGAN BACKEND PADA PERANGKAT LUNAK SOLIDARE

Adam Mukharil Bachtiar[1], Boby Indra Pratama[2]

[1] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
[2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : adammb@outlook.com[1], boby.indr@gmail.com[2]

ABSTRAK

Solidare merupakan aplikasi yang bertujuan untuk membantu anak-anak putus sekolah yang sebelumnya dikhususkan untuk komunitas sosial yang bernama Gerakan Bantu Seribu Anak Asuh (GBSA). Laporan tahunan dari Unesco Institute of Statistics (UIS) pada Juni 2013 menunjukkan bahwa terdapat 126 juta anak diseluruh dunia mengalami putus sekolah dan 44,4 juta diantaranya hidup dalam kemiskinan. Berdasarkan data dari laporan tersebut, solidare yang sebelumnya hanya dikhususkan pada komunitas GBSA saja, berubah konsep menjadi portal donasi secara global yang dapat bekerja sama dengan lembaga amal di dunia. Namun seiringan dengan perubahan konsep tersebut, sistem backend yang ada pada aplikasi solidare saat ini tidak dapat mendukung sistem dengan konsep baru tersebut. Selain itu tidak terdapatnya keamaan pada web service solidare yang sedang berjalan saat ini.

Oleh karena itu, solusi yang diusulkan adalah dengan mengembangkan backend pada perangkat lunak solidare agar dapat menunjang sistem dengan perubahan konsep tersebut serta membangun web service yang lebih aman pada perangkat lunak solidare. Metodologi penelitian yang digunakan adalah metode penelitian terapan. Pengembangan yang dilakukan pada backend meliputi database, web service dan website admin. Pada web service dilakukan penerapan algoritma kriptografi AES (Advance Encryption Standard)-128 dan pada website admin dilakukan pengembangan pada menu konfirmasi donasi dan rapot anak asuh, serta penambahan menu baru berupa menu lembaga amal, donartur aktif dan rekomendasi anak asuh.

Setelah dilakukan pengembangan dan pengujian dengan metode black box dapat disimpulkan bahwa sistem backend yang dikembangkan dari sisi database, web service dan website admin telah dapat mendukung sistem yang ada pada perangkat lunak solidare dengan konsep barunya yaitu sebagai portal donasi, serta web service yang telah di bangun sudah lebih aman dengan diterapkannya algoritma kriptografi AES-128 ke dalam web service yang di bangun.

Kata kunci : Solidare, donasi, web service, kriptografi, Advance Encryption Standard (AES)-128.

1. PENDAHULUAN

Pada bagian ini akan dipaparkan mengenai latar belakang masalah, maksud dan tujuan, serta landasan teori.

1.1 Latar Belakang Masalah

Solidare merupakan sebuah perangkat lunak yang bertujuan untuk membantu anak-anak yang putus sekolah maupun terancam putus sekolah. Perangkat lunak solidare sebelumnya bernama bantu anak asuh yang dikhususkan untuk sebuah komunitas yang bernama Gerakan Bantu Seribu Anak Asuh (GBSA).

Laporan tahunan dari *Unesco Institute of Statistics* (UIS) pada Juni 2013 menunjukkan bahwa terdapat 126 juta anak yang tersebar di seluruh dunia mengalami putus sekolah dan 44,4 juta diantaranya hidup dalam kemiskinan [1]. Berdasarkan data dari laporan tersebut, aplikasi solidare yang sebelumnya dikhususkan untuk komunitas GBSA berubah konsep menjadi aplikasi portal donasi yang bertujuan untuk membantu anak-anak putus sekolah di dunia. Dengan demikian aplikasi solidare ini tidak lagi hanya dikhususkan untuk komunitas GBSA saja, melainkan dapat bekerja sama dengan lembaga-lembaga amal yang ada di dunia.

Seiringan dengan perubahan konsep tersebut, *backend* pada aplikasi solidare yang sedang berjalan saat ini baik dari sisi *database*, *web service* dan website admin tidak dapat mendukung sistem yang ada pada perangkat lunak solidare. Selain itu permasalahan lain yang ada saat ini adalah *web service* yang sedang berjalan saat ini tidak aman, hal itu dikarenakan *web service* yang sedang berjalan saat ini dapat di akses dengan mudah oleh siapa saja. Maka dari itu perlu adanya pengembangan pada

backend aplikasi solidare agar dapat menunjang aplikasi solidare dengan perubahan konsep tersebut.

Berdasarkan pemamaparan permasalahan tersebut, didapatkan suatu solusi yaitu dengan mengembangkan *backend* pada aplikasi solidare dengan melakukan pengembangan pada *database* dan website serta membangun *web service* yang lebih aman dan dapat menunjang perangkat lunak solidare dengan perubahan konsep tersebut.

1.2 Maksud dan Tujuan

Maksud dari penelitian ini adalah mengembangkan *backend* pada perangkat lunak solidare. Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan *database* pada perangkat lunak solidare agar dapat mendukung sistem yang ada pada aplikasi solidare.
2. Membangun *web service* baru yang lebih aman dan dapat menunjang sistem yang ada pada aplikasi solidare.
3. Mengembangkan menu konfirmasi donasi dan rapot anak asuh serta menambahkan menu lembaga amal, donatur aktif dan rekomendasi anak asuh pada website admin solidare agar dapat mendukung sistem pada aplikasi solidare.

1.3 Web Service

Web Service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya yang terletak di satu server yang terhubung ke internet [2]. Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain XML, terdapat pula format pertukaran data lain yaitu JSON.

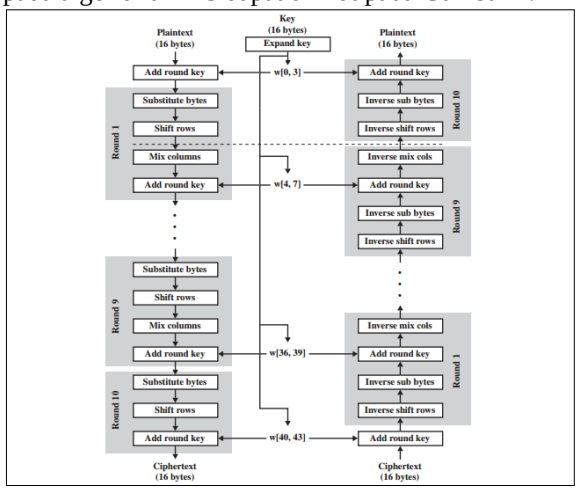
1.4 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [3].

1.5 Kriptografi

Kriptografi didefinisikan sebagai ilmu dan pelajaran untuk tulisan rahasia dengan pertimbangan bahwa komunikasi dan data dapat dikodekan untuk mencegah dari mata-mata atau orang lain yang ingin mengetahui isinya, dengan menggunakan kode-kode dan aturan-aturan tertentu dan metode lainnya sehingga hanya orang yang berhak yang dapat mengetahui isi pesan sebenarnya. Kriptografi memiliki dua jenis cara dalam melakukan enkripsi dan dekripsi yaitu *symmetric* dan *asymmetric*. Algoritma *symmetric* disebut dengan algoritma klasik, karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsinya. Sedangkan algoritma *asymmetric* sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsinya berbeda [4].

Pada penelitian ini jenis algoritma kriptografi yang digunakan adalah algoritma kriptografi simetrik dengan algoritma yang digunakan adalah AES (*Advance Encryption Standard*). Algoritma AES sendiri memiliki tiga jenis dengan panjang kunci yang berbeda, diantaranya AES-128, AES-192, dan AES-256 [5]. Proses enkripsi dan dekripsi pada algoritma AES dapat dilihat pada Gambar 1.



Gambar 1 Kriptografi pada Algoritma AES

1.6 Pengujian Blackbox

Metode pengujian *blackbox* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan *software*. Pengujian dengan menggunakan metode *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya:

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

Blackbox memiliki banyak teknik pengujian diantaranya *Equivalence Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing*. Salah satu teknik yang akan di bahas adalah *Equivalence Partitioning*.

Equivalence Partitioning merupakan metode black box testing yang membagi domain masukan dari suatu program menjadi beberapa kelas data dari kasus uji coba yang dihasilkan [6].

2. ISI PENELITIAN

2.1 Analisis dan Perancangan Sistem

Berikut akan dipaparkan mengenai analisis dan perancangan dari sistem yang dikembangkan.

2.1.1 Analisis Masalah

Solidare merupakan perangkat lunak yang bertujuan untuk membantu anak-anak yang putus sekolah maupun terancam putus sekolah. Solidare sebelumnya hanya dikhususkan untuk sebuah komunitas gerakan seribu anak asuh (GBSA). Saat ini solidare berubah konsep menjadi lebih global, yaitu sebagai portal donasi yang dapat bekerja sama dengan lembaga-lembaga amal yang ada di dunia. Seiringan dengan perubahan konsep tersebut, terdapat masalah-masalah yang ada pada perangkat lunak solidare. Masalah-masalah tersebut diantaranya:

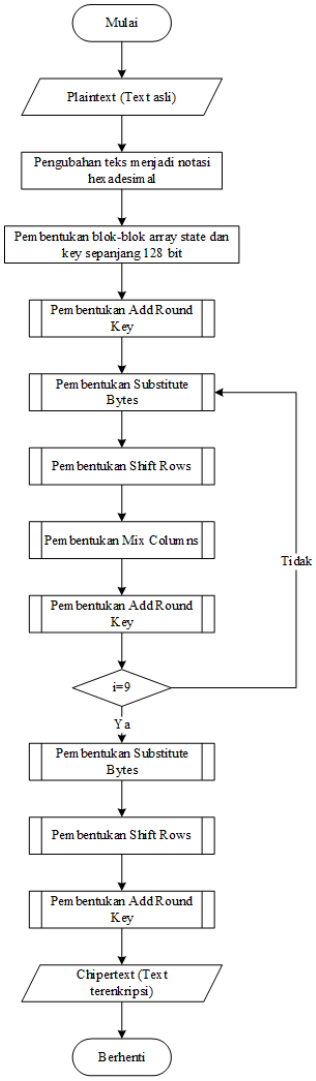
1. *Database* pada perangkat lunak solidare yang sedang berjalan saat ini belum optimal untuk mendukung sistem pada aplikasi solidare.
2. *Web service* yang sedang berjalan pada perangkat lunak solidare saat ini tidak dapat menunjang sistem yang ada pada aplikasi solidare, serta tidak terdapatnya keamanan pada *web service* solidare saat ini.
3. Menu konfirmasi dan laporan yang ada pada website solidare yang sedang berjalan saat ini belum optimal untuk mendukung sistem solidare, serta perlunya penambahan menu pengelolaan lembaga amal, donatur aktif dan rekomendasi anak asuh pada website solidare agar dapat mendukung sistem pada aplikasi solidare sebagai aplikasi portal donasi yang lebih global.

2.1.2 Analisis Algoritma Kriptografi

Analisis algoritma kriptografi menjelaskan bagaimana algoritma kriptografi diimplementasikan kedalam sistem yang ada pada perangkat lunak solidare. Algoritma kriptografi yang digunakan dalam perangkat lunak ini adalah algoritma kriptografi AES (Advance Ecryption Standart) – 128. Di dalam algoritma kriptografi yang digunakan terdapat dua proses yaitu proses enkripsi dan proses dekripsi.

1. Enkripsi

Proses enkripsi teks yang diimplementasikan ke dalam sistem dapat dilihat pada Gambar 2.



Gambar 2 Flowchart Enkripsi Teks

Penjelasan mengenai proses enkripsi yang ada pada Gambar 2 adalah sebagai berikut:

1. Pengubahan teks menjadi notasi hexadecimal

Proses ini merupakan standarisasi sebagai tahap awal sebelum teks dienkripsi oleh sistem dimana teks akan di konversi terlebih dahulu ke dalam bentuk notasi *hexadecimal*.

2. Pembentukan *block-block* array state dan key sepanjang 128 bit

Teks yang telah mengalami transformasi ke dalam bentuk *hexadecimal* selanjutnya akan dibentuk menjadi *block-block* sepanjang 128 bit dan dimasukkan ke dalam suatu *array* yang bernama *array state*.

Pada proses enkripsi teks, diperlukan sebuah *key* yang digunakan sebagai pengamanan teks. Panjang *key* akan disesuaikan dengan jenis AES yang digunakan. Pada penelitian ini algoritma AES yang digunakan adalah AES-128, yang berarti memiliki

panjang kunci 16 byte dengan perhitungan 128/8 = 16. Kunci tersebut akan ditransformasikan ke dalam notasi *hexadecimal* seperti teks pada tahap sebelumnya.

3. Pembentukan *Add Round Key*

Tahap ini merupakan *state* awal atau *initial state* proses XOR antara *array state* dengan *key*. Maksudnya adalah dengan meng-*XOR* kolom pertama pada *array state* dengan kolom pertama *key* hingga N-kali sepanjang teks tersebut habis dibentuk ke dalam *array state*.

4. Pembentukan *Substitute Bytes*

Proses ini bertujuan untuk memetakan atau menukar setiap *byte* dari *array state* dengan menggunakan tabel substitusi *S-Box*.

5. Pembentukan *Shift Rows*

Proses ini bertujuan untuk menggeser pada tiga baris terakhir pada *array state*. Baris yang bergeser adalah baris ke r=1 sebanyak 1 *byte*, baris ke r=2 sebanyak 2 *byte*¸ dan baris ke r=3 sebanyak 3 *byte*.

6. Pembentukan *Mix Columns*

Proses ini bertujuan untuk mengalikan setiap kolom dari *array state* dengan polinom *a(x)* mod (x^4+1). Transformasi ini dinyatakan sebagai perkalian matriks:

$$\begin{bmatrix} S'0c \\ S'1c \\ S'2c \\ S'3c \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S0,c \\ S1,c \\ S2,c \\ S3,c \end{bmatrix}$$

7. Pembentukan *Add Round key*

Sebelum melakukan proses *Add Round Key* ini, maka selanjutnya adalah melakukan ekspansi kunci atau membentuk kunci baru untuk proses *Add Round Key*. Setelah mendapatkan kunci ekspansi, maka langkah selanjutnya adalah melakukan perhitungan *Add Round Key*.

8. Pembentukan *Substitute Bytes*

Proses ini akan memetakan kembali hasil dari *Add Round Key* dengan tabel *S-Box.*

9. Pembentukan *Shift Rows*

Proses *Shift Rows* ini kembali akan menggeser baris terakhir pada *array state*. r=1 akan bergeser 1 *byte*, r=2 akan bergeser 2 *byte*, dan r=3 akan bergeser 3 *byte*.

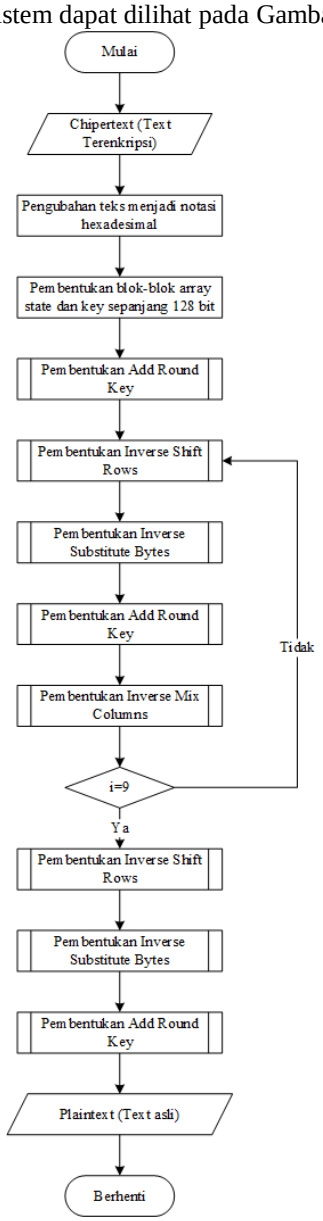
10. Pembentukan *Add Round key*

Proses *Add Round Key* ini adalah tahap terakhir atau putaran ke-10 dari proses enkripsi satu *array state* sepanjang 128 bit, maka selanjutnya adalah melakukan ekspansi kunci atau membentuk kunci baru untuk proses *Add Round Key*. Setelah kunci ekspansi didapatkan, maka selanjutnya adalah perhitungan *Add Round Key*.

Setelah proses enkripsi dilakukan, teks masih ada pada *block array state*. Maka oleh sistem akan ditransformasikan kembali untuk menjadi teks yang terenkripsi.

2. Dekripsi

Proses dekripsi teks yang diimplementasikan ke dalam sistem dapat dilihat pada Gambar 3.



Gambar 3 Flowchart Dekripsi Teks

Penjelasan mengenai proses dekripsi yang ada pada Gambar 3 adalah sebagai berikut:

1. Pengubahan teks menjadi notasi hexadecimal

Teks yang akan didekripsi akan di konversi terlebih dahulu ke dalam bentuk notasi *hexadecimal* oleh sistem.

2. Pembentukan *block-block* array state dan key sepanjang 128 bit

Dari proses yang telah dilakukan tahap sebelumnya, maka selanjutnya akan dibentuk kembali menjadi blok-blok sepanjang 128 bit dan dimasukkan ke dalam *array state*. *Cipherkey* yang digunakanpun sama dengan kunci ketika proses enkripsi teks pada putaran akhir proses enkripsi.

3. Pembentukan *Add Round Key*

Setelah proses Add Round Key selesai, maka akan menghasilkan *array state* yang nilainya sama dengan proses *Shift Rows*. Artinya, proses ini bertujuan untuk mengembalikan nilai dengan cara *XOR* atau *rollback*.

4. Pembentukan *Inverse Shift Rows*

Proses ini bertujuan untuk meggeser kembali baris akhir pada *array state* ke posisi awal.

5. Pembentukan *Inverse Subtitute Bytes*

Proses *Inverse Substitute Bytes* akan memetakan kembali *array state* hasil dari proses *Inverse Shift Rows* dengan tabel *S-Box*.

6. Pembentukan *Add Round Key*

Proses *Add Round Key* ini akan mengembalikan nilai *array state* dengan cara melakukan proses XOR terhadap hasil *Inverse Substitute Bytes* dengan key yang telah di ekspansi sebelumnya.

7. Pembentukan *Inverse Mix Columns*

Proses ini bertujuan untuk mengembalikan hasil perkalian setiap kolom dari *array state* yang telah dilakukan pada proses *mix column* pada saat enkripsi.

8. Pembentukan *Inverse Shift Rows*

Proses *Inverse Shift Rows* akan kembali menggeser 1 *byte* pada baris akhir di *array state*.

9. Pembentukan *Inverse Subtitute Bytes*

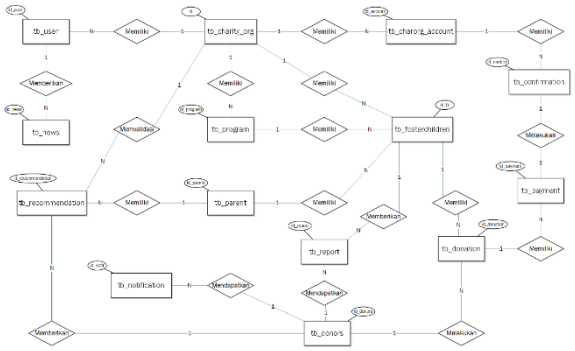
Proses *Inverse Substitute Bytes* akan kembali memetakan setiap nilai *array state* dengan tabel *S-Box* untuk mengembalikan setiap nilai yang ada di dalam *array state* ke dalam kondisi awal sebelum masuk proses *Add Round Key* putaran pertama saat enkripsi.

10. Pembentukan *Add Round Key*

Proses *Add Round Key* ini adalah proses terakhir dari rangkaian langkah proses dekripsi dari teks. Tahap ini akan dilakukan proses *XOR* antara ekspansi key putaran pertama enkripsi ketika proses *Add Round Key* dengan hasil proses *Inverse Substitute Bytes*. Setelah proses *Add Round Key* tersebut selesai, maka data akan kembali pada kondisi awal sesuai dengan teks sebelum melalui proses enkripsi dilakukan. Namun teks tersebut harus di transformasikan terlebih dahulu karena hasil dari dekripsi ini masih dalam bentuk notasi *hexadecimal*. Sebelum ditransformasikan, notasi *hexadecimal* tersebut dibentuk kedalam blok-blok *array state* sebanyak 128bit sampai *array state* terakhir pada posisi ke-N. Setelah proses ini selesai, baru dilakukan transformasi oleh sistem untuk dikembalikan menjadi teks semula atau teks aslinya.

2.1.3 Analisis Data

Analisis data yang menjadi acuan dalam pengembangan perangkat lunak ini adalah dokumen-dokumen yang berhubungan dengan perangkat lunak solidare serta objek-objek lain yang terkait. *Entity relational diagram* (ERD) dan kardinalitasnya dapat dilihat pada Gambar 4.



Gambar 4 ERD dan Kardinalitas

Konsep analisis data lain yang digunakan dalam penelitian ini adalah konsep *Obejct Relational Mapping* (ORM). Object relational mapping melakukan pemetaan terhadap tabel-tabel pada basis data relasional dengan suatu class entitas yang ada pada bahasa pemrograman berorientasi objek.

Pemetaan yang dilakukan ORM akan membutuhkan suatu jembatan sebagai media pertukaran datanya. Konsep pertukaran data yang digunakan pada penelitian ini berupa format data JSON yang dapat menghubungkan ORM dengan database fisik yang ada pada server. Struktur format data JSON yang digunakan pada penelitian ini terbagi menjadi tiga yaitu data objek, data array dan data null.

1. Data Objek

Format data ketika data yang diterima dari server merupakan data tunggal atau berupa satu objek dapat dilihat pada Tabel 1.

Tabel 1 Struktur Data Objek

Struktur Data Objek

{
“key” : “value”,
“key” : “value”,
“key” : “value”
}

2. Data Array

Format data ketika data yang diterima dari server merupakan data array dapat dilihat pada Tabel 2.

Tabel 2 Struktur Data Array

Struktur Data Array

{
“result” : “value”,
“item” : [
{
“key” : “value”,
“key” : “value”,
“key” : “value”
},
{
“key” : “value”,
“key” : “value”,
“key” : “value”
}
]
}

3. Data Null

Format data ketika data yang diterima dari server merupakan null dapat dilihat pada Tabel 3.

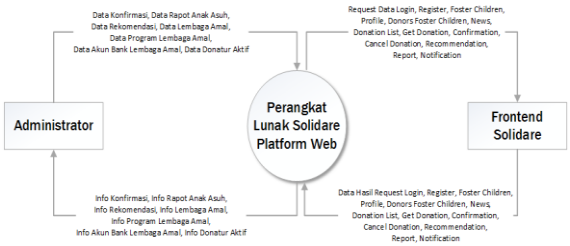
Tabel 3 Struktur Data Null

Struktur Data Null

{
“result” : “value”,
“message” : “value”,
“status_code” : “value”
}

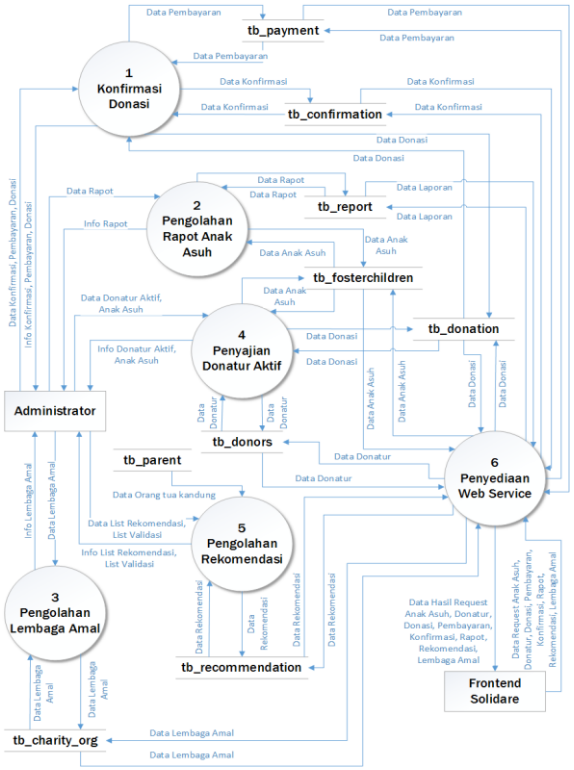
2.1.4 Analisis Kebutuhan Fungsional

Berikut adalah Diagram konteks dari pengembangan perangkat lunak ini yang dapat dilihat pada Gambar 5.



Gambar 5 Diagram Konteks

Berikut adalah DFD level 1 dari pengembangan perangkat lunak ini yang dapat dilihat pada Gambar 6.



Gambar 6 DFD Level 1

2.2 Implementasi dan Pengujian Sistem

Berikut akan dipaparkan mengenai implementasi dan pengujian sistem dari perangkat lunak yang dikembangkan.

2.2.1 Lingkungan Implementasi

Lingkungan implementasi merupakan spesifikasi hardware dan software dimana sistem ini akan dipasang dan diakses. Berikut adalah lingkungan implementasinya:

1. Lingkungan Hardware

Berikut adalah spesifikasi hardware dalam pembangunan perangkat lunak ini.

Tabel 4 Lingkungan Hardware Pengembangan Perangkat Lunak

No	Item	Spesifikasi
1	Processor	Intel(R) Core i7-2640M @2.80Ghz
2	Harddisk	750 GB
3	Memory	4 GB
4	VGA	Intel HD Graphics 3000
5	Monitor	13.3 inch
6	Mouse	Optical Logitech
7	Keyboard	Qwerty Sony S Series
8	USB Port	3 buah
9	Koneksi Internet	Ada

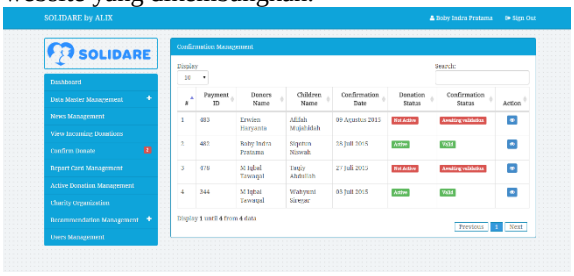
2. Lingkungan Software

Perangkat lunak yang digunakan dalam pembangunan perangkat lunak ini adalah:

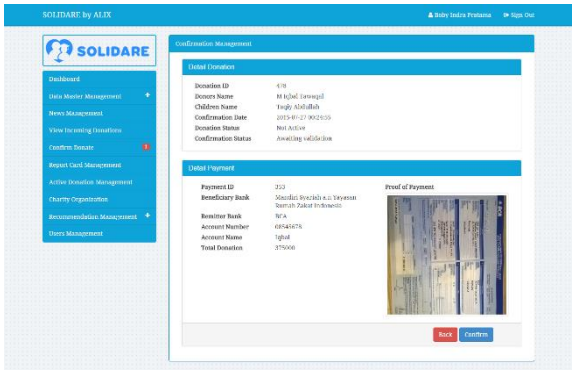
a. Sistem Operasi Windows 8.1 Professional 64-bit.
b. Bahasa Pemrograman PHP.
c. DBMS : MySql 5.6.14.
d. Web Browser : Google Chrome 43.0.2357.130 m dan Mozilla Firefox 38.5.
e. Code Editor menggunakan Sublime text.
f. Web Server : XAMPP.
g. Pertukaran data yang dilakukan menggunakan JSON.

2.2.2 Implementasi Antarmuka

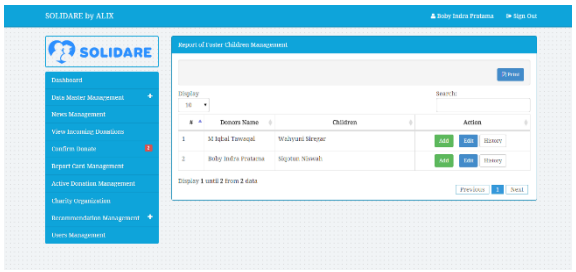
Berikut adalah implementasi antarmuka dari website yang dikembangkan:



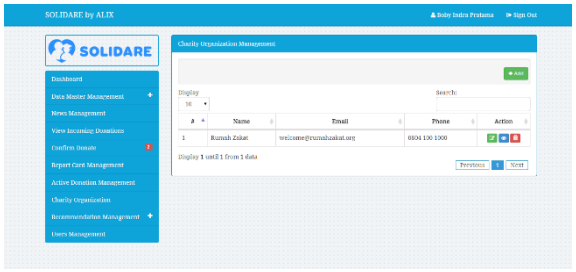
Gambar 7 Management Konfirmasi Donasi



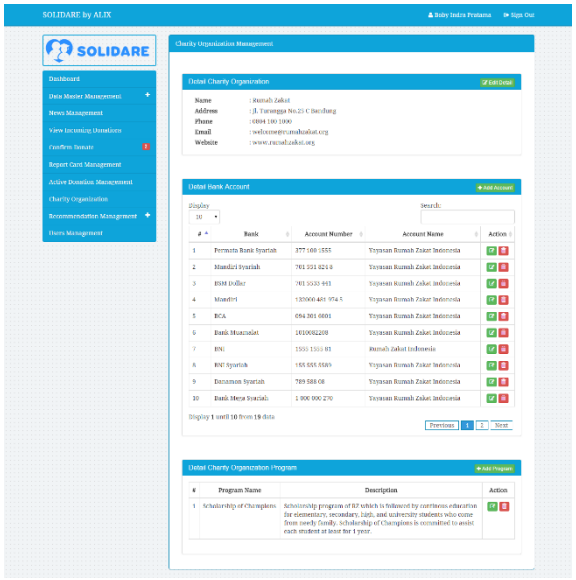
Gambar 8 Detail Konfirmasi Donasi



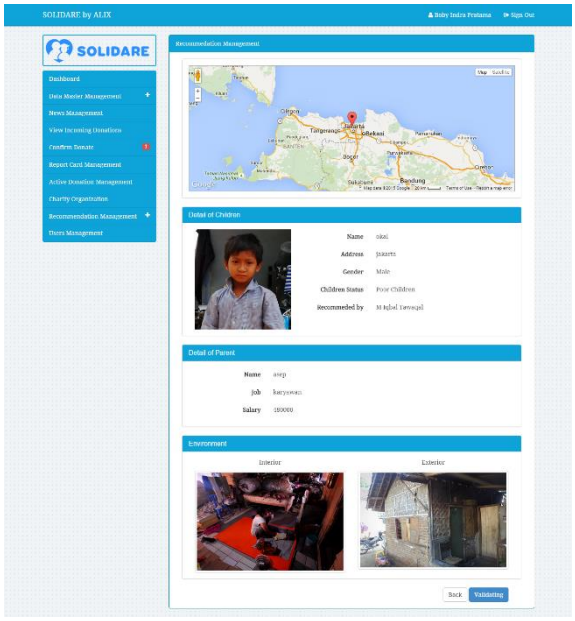
Gambar 9 Management Rapot Anak Asuh



Gambar 10 Management Lembaga Amal



Gambar 11 Detail Lembaga Amal



Gambar 12 Detail Rekomendasi

2.2.3 Pengujian

Pengujian sistem ini terdiri dari dua tahap yakni pengujian alpha dan pengujian beta. Metode yang digunakan dalam pengujian sistem ini adalah dengan menggunakan metode pengujian *black box* dimana pengujian ini berfokus pada persyaratan fungsional perangkat lunak.

1. Rencana Pengujian

Rencana pengujian yang akan dilakukan pada *backend* perangkat lunak solidare adalah pengujian fungsional dengan jenis pengujian *black box*.

2. Skenario Pengujian

Skenario pengujian fungsional yang dilakukan adalah dengan menguji website solidare dengan skenario urutan pengujian yang dapat dilihat pada Tabel 5.

Tabel 5 Tabel Skenario Pengujian Website Admin Solidare

No	Kelas Uji	Poin Pengujian	Jenis Pengujian
1	Pencarian data rapot anak asuh	Mengisi kata kunci pencarian	*Equivalence Partitioning*
2	Menambahkan data rapot anak asuh	Mengisi data rapot	*Equivalence Partitioning*
		Validasi *field* rapot	*Equivalence Partitioning*
		Menekan tombol *save data*	*Equivalence Partitioning*
3	Pencarian data lembaga amal	Mengisi kata kunci pencarian	*Equivalence Partitioning*
4	Menambahkan data lembaga amal	Mengisi data lembaga amal	*Equivalence Partitioning*
		Validasi *field* lembaga amal	*Equivalence Partitioning*
		Menekan tombol *save data*	*Equivalence Partitioning*
5	Pencarian data akun bank lembaga amal	Mengisi kata kunci pencarian	*Equivalence Partitioning*
6	Menambahkan	Mengisi data akun	*Equivalence*

No	Kelas Uji	Poin Pengujian	Jenis Pengujian
	data akun bank lembaga amal	bank lembaga amal	*Partitioning*
		Validasi *field* akun bank lembaga amal	*Equivalence Partitioning*
		Menekan tombol *save data*	*Equivalence Partitioning*
7	Menambahkan data program lembaga amal	Mengisi data program lembaga amal	*Equivalence Partitioning*
		Validasi *field* program lembaga amal	*Equivalence Partitioning*
		Menekan tombol *save data*	*Equivalence Partitioning*
8	Pencarian data donatur aktif	Mengisi kata kunci pencarian	*Equivalence Partitioning*
9	Percarian data rekomendasi anak asuh	Mengisi kata kunci pencarian	*Equivalence Partitioning*
10	Pencarian data validasi rekomendasi anak asuh	Mengisi kata kunci pencarian	*Equivalence Partitioning*

3. Evaluasi Pengujian

Berdasarkan hasil pengujian fungsional yang telah dilakukan maka dapat disimpulkan bahwa website yang telah dikembangkan sudah berjalan sesuai dengan yang diharapkan, baik itu dari segi validasi maupun proses penanganan kesalahan.

3. PENUTUP

Kesimpulan dari pengembangan *backend* pada perangkat lunak solidare ini adalah:

1. *Database* yang telah dikembangkan sudah dapat mendukung sistem yang ada pada perangkat lunak solidare.
2. *Web service* yang telah di bangun sudah lebih aman dengan telah diimplementasikannya algoritma kriptografi AES-128 dan *web service* yang telah di bangun sudah dapat mendukung sistem yang ada pada perangkat lunak solidare.
3. Menu konfirmasi donasi dan rapot anak asuh sudah dikembangkan dengan melakukan perubahan UI (*user interface*) dan penambahan fungsional. Menu lembaga amal, donatur aktif dan rekomendasi anak asuh sudah ditambahkan pada website solidare. Semua menu yang sudah dikembangkan dan ditambahkan sudah dapat mendukung sistem yang ada pada perangkat lunak solidare dan telah berjalan sesuai dengan yang diharapkan.

DAFTAR PUSTAKA

[1] UNESCO INSTITUTE FOR STATISTICS, Fixing The Broken Promise Of Education For All, UNICEF, 2014.

[2] J. Simarmata, Rekayasa Web, Yogyakarta: ANDI, 2010.

[3] j. org, "json org," [Online]. Available: http://json.org/json-id. [Diakses 25 April 2014].

[4] B. Schneier, Applied Cryptography, India: Wiley, 2007.

[5] W. Stalling, Cryptography and Network Security Principles and Practice, Pearson, 2014.

[6] M. E. Khan dan F. Khan, "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques," *International Journal of Advanced Computer Science and Applications*, vol. 3, no. 6, pp. 12-15, 2012.