**TUGAS AKHIR**

**ALGORITMA *LINEAR CONGRUENTIAL GENERATORS* (LCG) PADA *COMPUTER BASED TEST***

**PENERIMAAN MAHASISWA BARU**

**SEKOLAH TINGGI TEKNOLOGI ADISUTJIPTO**

Diajukan untuk memenuhi sebagian prasyarat

mencapai derajat sarjana S1

****

# ­

###### Diajukan oleh :

###### Saiful Bahri

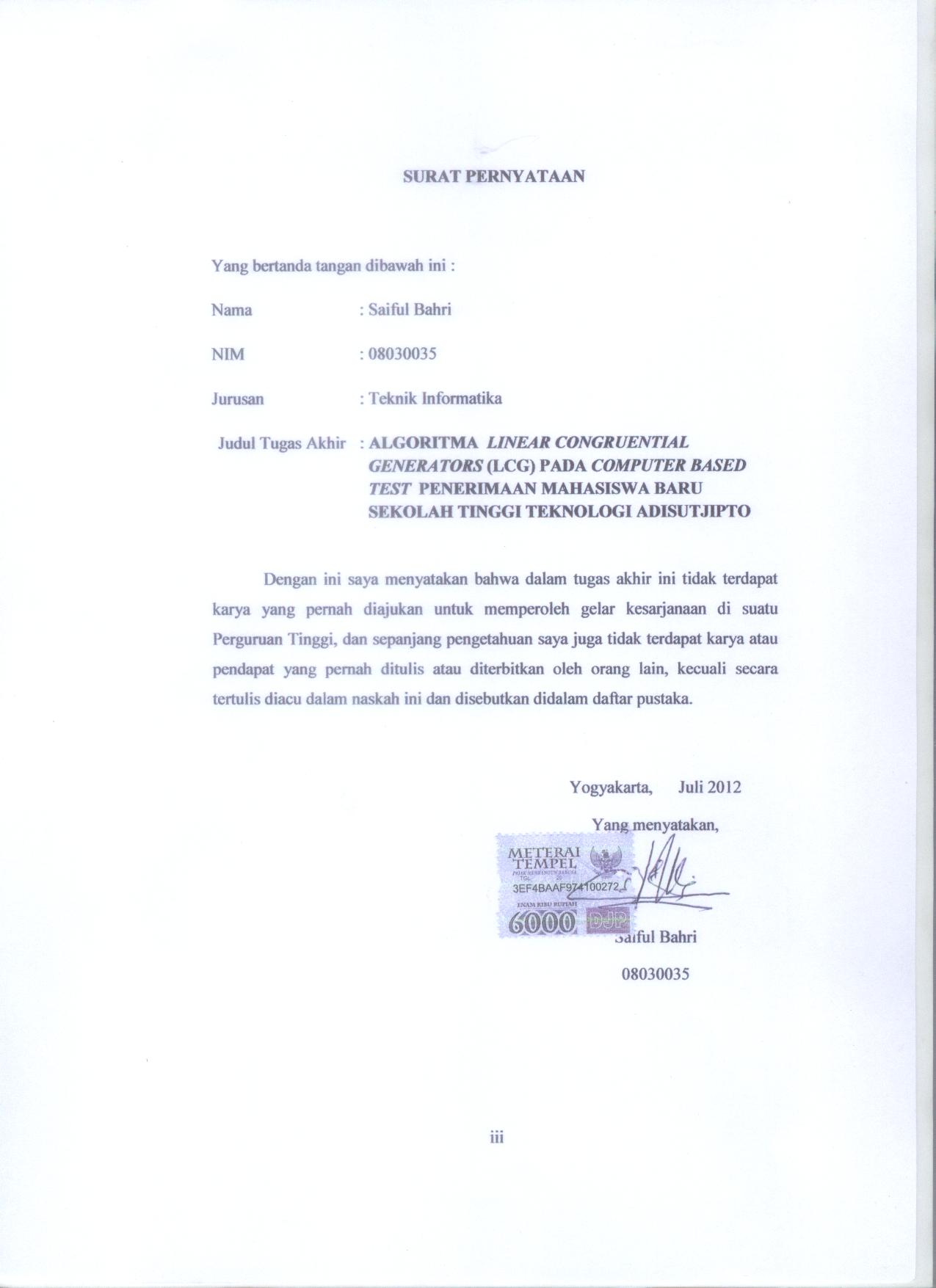
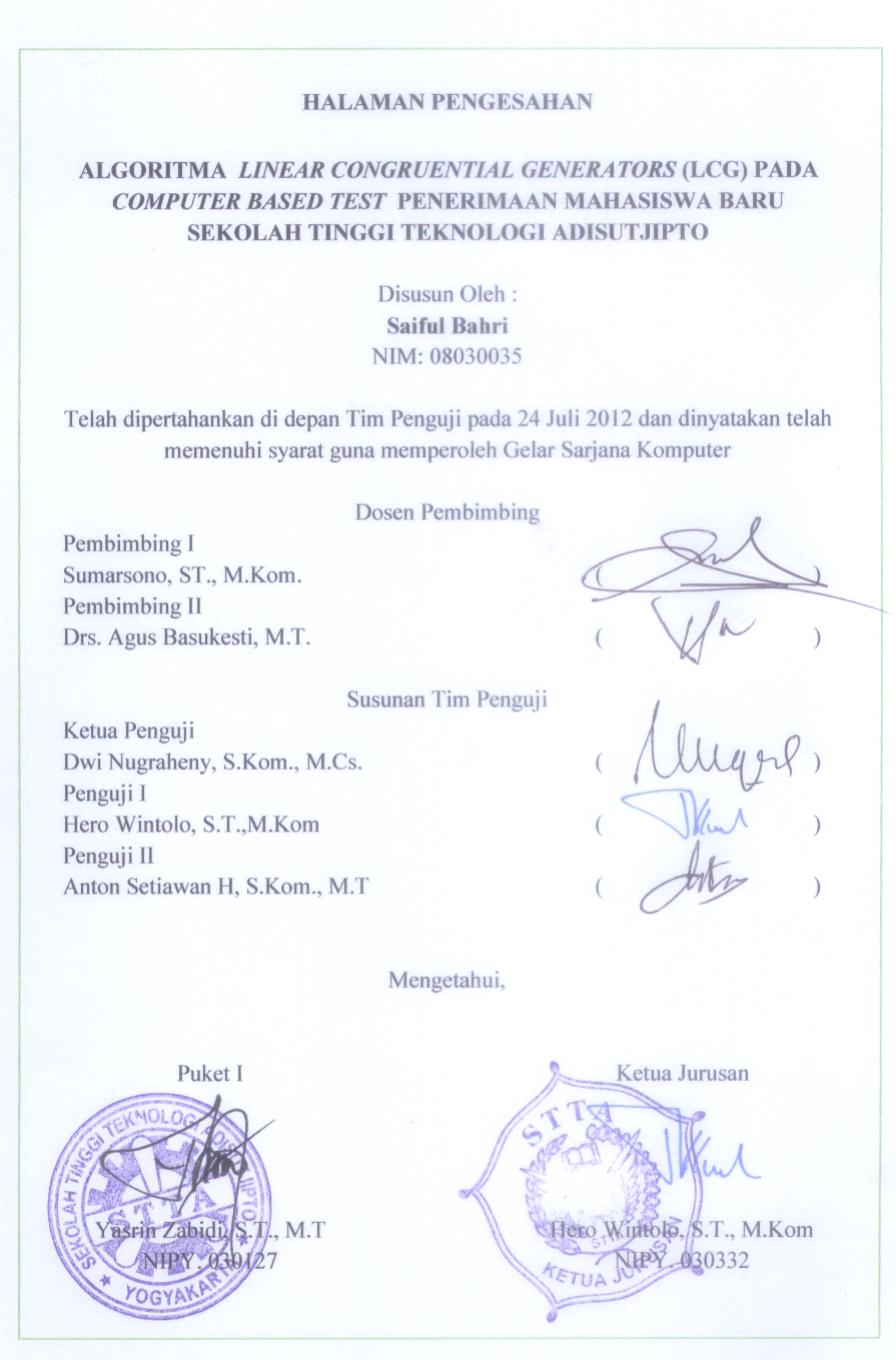
**08030035**

**TEKNIK INFORMATIKA**

### SEKOLAH TINGGI TEKNOLOGI ADISUTJIPTO (STTA)

### YOGYAKARTA

### 2012



**HALAMAN PERSEMBAHAN**

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ

(Segala puji hanya bagi Allah, Tuhan Semesta Alam)

Al-Fatihah : 2

**Kupersembahankan Karya ini kepada :**

Allah SWT yang banyak memberikan keimanan,ketaqwaan, kenikmatan dalam hidup dunia akhirat. Rosulullah yang selalu menjadi teladan.

Ibu dan Bapak dengan segala perjuangannya, kakak-kakakku yang menjadikanku adik, dan saudara-saudaraku yang selama ini telah banyak menemani dalam susah dan duka, memberikan doa, semangat, optimisme dan dukungan.

**KATA PENGANTAR**

Puji syukur kepada Allah SWT atas segala berkat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini ditujukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Sekolah Tinggi Teknologi adisutjipto STTA Yogyakarta.

Pada kesempatan ini pula penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Marsma TNI(Purn) Ir. Sutjianto S., M.T., selaku Ketua Sekolah Tinggi Teknologi Adisutjipto STTA Yogyakarta.
2. Hero Wintolo, S.T., M.Kom., selaku Ketua Jurusan Teknik Informatika Sekolah Tinggi Teknologi Adisutjipto STTA Yogyakarta.
3. Sumarsono, ST., M.Kom. dan Drs. Agus Basukesti, M.T., selaku dosen pembimbing yang telah membimbing penulis selama pelaksanaan skripsi ini.
4. Seluruh dosen dan staf karyawan Sekolah Tinggi Teknologi Adisutjipto STTA Yogyakarta.
5. Bapak dan Ibu, serta kakak-kakakku tercinta.
6. Pakde Win dan keluarga yang telah memberi bantuan dan dukungan kepada penulis selama kuliah di Jogja.
7. Pethukers yang selalu memberikan keceriaan dihari-hari selama ini dan seterusnya.
8. Semua pihak yang telah membantu sampai terselesaikannya penyusunan skripsi.

Akhirnya, penulis menyadari penulisan skripsi ini masih jauh dari sempurna. Untuk itu penulis mengharapkan dengan sangat saran dan masukan yang membangun dari pembaca sekalian. Semoga penulisan skripsi ini dapat membawa manfaat bagi kita semua.

Yogyakarta, Juli 2010

Penulis,

**DAFTAR ISI**

HALAMAN JUDUL i

HALAMAN PENGESAHAN ii

SURAT PERNYATAAN iii

HALAMAN PERSEMBAHAN iv

KATA PENGANTAR v

DAFTAR ISI vii

DAFTAR GAMBAR x

DAFTAR TABEL xiii

DAFTAR LAMPIRAN xiv

INTISARI xv

*ABSTRACT* xvi

BAB I PENDAHULUAN 1

* 1. Latar Belakang Masalah 1
  2. Rumusan Masalah 2
  3. Batasan Masalah 2
  4. Tujuan 3

BAB II LANDASAN TEORI 5

* 1. Tinjauan Pustaka 5
  2. *Computer Based Test* (CBT) 6
  3. Tes Potensi Akademik (TPA) 7

2.3.1 Materi Tes Potensi Akademik 7

* 1. Algoritma 8

2.4.1 *Linear Congruential Generators* (LCG) 8

BAB III ANALISIS DAN PERANCANGAN SISTEM 10

* 1. Analisis Sistem 10
     1. Analisis Soal 10
     2. Analisis *Input*, Proses dan *Output* 10
     3. Perangkat Lunak 12
     4. Perangkat Keras 12
  2. Analisis Algoritma 12
     1. Algoritma Pengambilan Soal 13
     2. Alur Algoritma *Linear Cngruential Generators* (LCG) 15
     3. Algoritma Acak pada CBT PMB STTA.. 17
     4. Algoritma Acak Soal pada Aplikasi.. 18
     5. Algoritma Perhitungan Hasil Ujian.. 19
  3. Perancangan Sistem 21
     1. Diagram *Use Case* 21
     2. Diagram *Class* 22
     3. Diagram *Sequence* 27
     4. Perancangan *Database* 29

3.3.4.1 Normalisasi (Tahapan Normalisasi) 30

* + 1. Rancangan Tampilan 44

BAB IV IMPLEMENTASI DAN PEMBAHASAN 51

* 1. Implementasi Sistem 51
     1. ImplementasiAlgoritma *Linear Cngruential Generators*  51
     2. Implementasi Algoritma Acak Soal dan Jawaban 53
     3. ImplementasiAlgoritma Pengambilan Soal 54
     4. Impementasi *Database* 56
     5. Impementasi Rancangan Tampilan 57

4.1.5.1 Impementasi Rancangan Tampilan *Client* 57

4.1.5.2 Impementasi Rancangan Tampilan *Server* 59

4.1.5.3 Impementasi Rancangan Tampilan Pewawancara 71

* 1. Uji Fungi 72
  2. Uji Fungsi Algoritma 72
  3. Uji Jaringan *Internet* 74
  4. Uji Pengguna 76

BAB V KESIMPULAN DAN SARAN 79

* 1. Kesimpulan 79
  2. Saran 79

DAFTAR PUSTAKA 80

**DAFTAR GAMBAR**

Gambar 3.1 *Flowchart* Ambil Soal Per Bidang 13

Gambar 3.2 *Flowchart* Pengambilan Soal 14

Gambar 3.3 *Flowchart* Acak Soal 18

Gambar 3.4 *Flowchat* Hasil Soal 20

Gambar 3.5 Diagram *Use Case* 21

Gambar 3.6 Diagram *Class Package Domain* 23

Gambar 3.7 Turunan *Class* Soal 24

Gambar 3.8 *Implement Package Service* 24

Gambar 3.9 Diagram *Class Package Impl* 25

Gambar 3.10 Diagram *Class Package Controller* 26

Gambar 3.11 Diagram *Sequence* 28

Gambar 3.12 Relasi Table CBT 29

Gambar 3.13 *Login Admin* 44

Gambar 3.14 Menu Utama *Admin* 45

Gambar 3.15 *Input Soal* 45

Gambar 3.16 *Upload* Soal 46

Gambar 3.17 *Edit Setting* 46

Gambar 3.18 *Input Operator* 47

Gambar 3.19 *Print Out* Hasil 48

Gambar 3.20 *Edit Grade* 49

Gambar 3.21 *Login* Peserta 49

Gambar 3.22 *Panel* Soal 50

Gambar 3.13 Hasil Pengerjaan Soal 50

Gambar 4.1 Implementasi Tabel 56

Gambar 4.2 Koneksi *Database* 56

Gambar 4.3 Contoh HQL (*Hibernate Query Language*) 56

Gambar 4.4 *Login* Peserta 57

Gambar 4.5 Halaman Pengerjaan Soal 58

Gambar 4.6 Halaman Hasil Pengerjaan 59

Gambar 4.7 *Login Admin* 59

Gambar 4.8 Halaman *Dashboard* 60

Gambar 4.9 Halaman Soal 61

Gambar 4.10 Upload Soal 61

Gambar 4.11 *Input* Soal 62

Gambar 4.12 Halaman Hasil Ujian 63

Gambar 4.13 *Print Out* Hasil Ujian 63

Gambar 4.14 Halaman *Setting* Soal 64

Gambar 4.15 Halaman *Edit Setting* Soal 65

Gambar 4.16 Halaman *Operator* 65

Gambar 4.17 *Input Operator* 66

Gambar 4.18 *Setting Grade* 67

Gambar 4.19 *Edit Setting Grade*  67

Gambar 4.20 Halaman Pewawancara 68

Gambar 4.21 *Input* Pewawancara 69

Gambar 4.22 Statistik 70

Gambar 4.23 *Detail* Statistik 70

Gambar 4.24 *Panel Login* Pewawancara 71

Gambar 4.25 *Dashboard* Pewawancara 72

Gambar 4.26 Uji Algoritma 1 73

Gambar 4.27 Uji Algoritma 2 73

Gambar 4.28 Uji Jaringan 1 74

Gambar 4.29 Uji Jaringan 2 75

Gambar 4.30 Grafik Hasil Kuisioner CBT 77

**DAFTAR TABEL**

Tabel 3.1 Contoh Deret 16

Tabel 3.2 *Admin* 31

Tabel 3.3 Calon\_mahasiswa 33

Tabel 3.4 Jawaban 34

Tabel 3.5 Kategori 35

Tabel 3.6 Pengerjaan\_soal 36

Tabel 3.7 *Role* 37

Tabel 3.8 *Setting* 38

Tabel 3.9 Soal 39

Tabel 3.10 Pengerjaan\_soal\_list 40

Tabel 3.11 Ujian 41

Tabel 3.12 Grade\_lulus 42

Tabel 3.13 Jenis\_soal 43

Tabel 4.1 Format *Upload* 62

Tabel 4.2 Kuisioner 76

Tabel 4.3 Hasil Kuisioner 77

**ALGORITMA *LINEAR CONGRUENTIAL GENERATORS* (LCG) PADA *COMPUTER BASED TEST* PENERIMAAN MAHASISWA BARU**

**SEKOLAH TINGGI TEKNOLOGI ADISUTJIPTO**

Oleh

**Saiful Bahri**

08030035

**INTISARI**

Dalam Penerimaan Mahasiswa Baru (PMB) di Sekolah Tinggi Teknologi Adisutjipto (STTA) Yogyakarta, panitia PMB telah memiliki *Computer Based Test* (CBT) untuk ujian calon mahasiswa baru sejak 2010. Namun CBT yang dimiliki oleh STTA belum mendukung pengacakan soal maupun jawaban soal sehingga setiap ujian yang dikeluarkan, urutan soal serta jawaban akan tetap sama.

Dari sekian banyak Algoritma random, Algoritma *Linear Congruential Generators* (LCG) adalah algoritma yang cocok digunakan dalam pengacakan soal pada CBT karena proses yang cepat dalam menghasilkan bilangan acak serta algoritmanya yang mudah untuk dipelajari dan diimplementasikan. Dengan memberi umpan LCG sebuah nilai panjang waktu sekarang maka akan menghasilkan nilai yang berbeda disetiap acaknya sejak pengacakan pertama.

Dengan pengujian dan pengamatan yang telah dilakukan pada CBT yang menggunakan algoritma LCG, urutan soal dan jawaban selalu berbeda disetiap ujian yang diujikan.

Kata kunci: *Computer Based Test* (CBT), *Random*, *Linear Congruential Generators* (LCG)

***LINEAR CONGRUENTIAL GENERATORS (LCG) ALGORITHM ON A COMPUTER BASED TEST HIGH SCHOOL ADMISSION ADISUTJIPTO TECHNOLOGY***

***By***

**Saiful Bahri**

**08030035**

***ABSTRACT***

*The Admissions in the College of Technology Adisutjipto Yogyakarta, the committee has had PMB Computer Based Test (CBT) to test new students since 2010. However, CBT is owned by STTA not support randomization and answer questions about each exam so issued, the order of questions and answers will remain the same.*

*Of the many random algorithm, algorithm Congruential Linear Generators (LCG) is an algorithm suitable for use in scrambling about in CBT because of the rapid process of generating random numbers and the algorithm is easy to learn and implement. By giving feedback LCG a current value of the length of time it will produce a different value in each random from the first randomization.*

*The tests and observations that have been conducted on the use of CBT LCG algorithm, the order of questions and answers are always different in every exam tested.*

*Keywords: Computer Based Test (CBT), Random, Linear Congruential Generators (LCG)*

**BAB I**

**PENDAHULUAN**

1. **Latar Belakang Masalah**

Ujian masuk Perguruan Tinggi sekerang sudah menjadi hal yang biasa dilakukan oleh para calon mahasiswa baru. Dengan banyaknya calon mahasiswa baru yang mendaftar serta waktu daftarnya yang berbeda-beda, dibutuhkan sistem yang dapat mengatur proses ujian masuk dengan cepat. Dengan keterbatasan panitia yang mengurusi masalah ujian masuk di sebuah perguruan tinggi, studi kasus di Sekolah Tinggi Teknologi Adisutjipto Yogyakarta, dibutuhkanlah sebuah sistem aplikasi yang dapat membantu panitia penerimaan mahasiswa baru dalam ujian masuk. *Computer Based Test* (CBT) merupakan solusi yang dapat membantu panitia penerimaan mahasiswa baru dalam mengurusi tes ujian masuk.

Tujuan utama ujian masuk adalah menyaring calon mahasiswa sesuai kecerdasan dan kemampuan. Untuk itu dibutuhkan CBT yang dapat memaksimalkan kegiatan ujian masuk guna menggali sejauh mana kemampuan calon mahasiswa. Bila secara manual ujian masuk tersebut soal antar calon mahasiswa akan selalu sama, sehinggi kunci jawabanpun akan sama yang berakibat pada calon mahasiswa untuk lebih menggunakan cara tanya pada temannya dalam menjawab soal daripada menggunakan kemampuan diri sendiri. Untuk itu di dalam sistem yang terkomputerisasi, CBT dapat mengacak soal sehingga urutan soal tiap peserta tes ujian masuk akan berbeda, namun cara seperti ini pun masih memungkinkan calon mahasiswa berbuat curang karena urutan jawabanpun akan tetap sama. Oleh karena itu untuk lebih memaksimalkan calon mahasiswa dalam menggunakan kemapuannya sendiri jawaban yang ditampilkan didalam CBT pun akan diacak. Dengan cara ini walaupun soal antar peserta ujian sama, namun dalam urutan jawaban bisa berbeda sehingga peserta ujian dapat teruji secara maksimal kemampuan yang dimilikinya tanpa mengandalkan teman atau orang lain.

1. **Rumusan Masalah**

Berdasarkan latar belakang di atas, dapat dirumuskan permasalahan yang dihadapi adalah bagaimana mengimplementasi algoritma *Linear Congruential Generators* (LCG) dalam pengacakan soal serta jawaban soal pada CBT di Sekolah Tinggi Teknologi Adisutjipto Yogyakarta.

1. **Batasan Masalah**

Dengan banyaknya aspek dalam penulisan laporan serta pengembangan aplikasi perangkat lunak maka diperlukan batasan masalah yang jelas untuk menghindari kerancuan dan ketidakjelasan dalam pembahasan. Adapun batasan masalahnya adalah sebagai berikut:

* 1. Algoritma acak yang digunakan didasarkan pada algoritma *Linear Congruential Generators (LCG)* Donald E. Knuth.
  2. Soal yang dikeluarkan adalah Tes Potensi Akademik (TPA) dengan empat bidang yaitu tes kemampuan verbal (bahasa), tes kemampuan numerik (angka), tes kemampuan penalaran (logika), tes kemampuan spasial (gambar).
  3. Aplikasi ini hanya berfungsi sebagai alat bantu dalam ujian masuk penerimaan mahasiswa baru.
  4. *Client* hanya dapat mengakses soal-soal yang sudah dimasukkan ke dalam bank soal oleh panitia.
  5. Diasumsikan pelaksanaan ujian telah berjalan sesuai jadwal yang ditentukan oleh panitia.
  6. *Server* menerima permintaan dari *client* dan bertugas mengirim soal beserta kunci, serta menampilkan hasil tes kepeserta ujian.
  7. Aplikasi ini menampilkan soal-soal dengan tipe pilihan ganda.
  8. Aplikasi ini mempunyai waktu untuk membatasi peserta ujian dalam mengerjakan soal.
  9. Aplikasi *client* hanya menangani dari calon mahasiswa memasukan nomor pendaftaran sampai selesai mengerjakan soal dan melihat nilai yang didapatkan.
  10. Ujicoba aplikasi dalam jaringan *internet* menggunakan *provider* SmartFren.

1. **Tujuan**

Adapun tujuan dari skripsi ini adalah :

* 1. Penerapan algoritma acak LCG dalam sebuah aplikasi CBT.
  2. Untuk membuat Aplikasi CBT berbasis *client-server* untuk membantu ujian penerimaan mahasiswa baru di Sekolah Tinggi Teknologi Adisutjipto Yogyakarta.
  3. Membantu panitia PMB dalam menyelenggarakan ujian masuk dan mengkoreksi serta menampilkan hasilnya secara lebih cepat dan *realtime*.

**BAB II  
LANDASAN TEORI**

1. **Tinjauan Pustaka**

Penelitian seperti ini pernah disusun oleh Randi Mesyah (2012) dengan judul Pre-tes Praktikum Secara *Online* Berbasis *client-server* dengan *Remote Method Invocation* (RMI). Tujuan penelitian tersebut adalah mengimplementasikan RMI *Java* dalam aplikasi pre-tes praktikum guna membantu dosen dalam penyelenggaraan pre-tes mahasiswa. *Interface* aplikasi dibangun menggunakan *Java Desktop*.

Penelitian tentang algoritma acak juga pernah dilakukan oleh Andresta Ramadhan (2006) dengan judul Perbandingan Algoritma *Linear Congruential Generators*, *Blum Blum Shub*, dan *Mersenne Twister* untuk Membangkitkan Bilangan Acak Semu. Penelitian ini bertujuan membandingkan macam macam algoritma yang dapat menghasilkan bilangan acak dengan kelemahan dan keunggulan masing masing.

Penelitian yang akan dilakukan di sini adalah pemanfaatan algoritma acak yang berdasar *Linear Congruential Generators* (LCG) pada aplikasi CBT yang digunakan untuk PMB di sekolah tinggi teknologi adisutjipto (STTA). Algoritma LCG akan digunakan untuk pengacakan soal pada CBT. Aplikasi menggunakan sistem *client-server* berbasis *web* dan dibangun menggunakan bahasa pemrograman *Java* dengan *database* menggunakan *PostgreSQL*.

1. **Computer Based Test (CBT)**

Suatu Penilaian Berbasis Komputer, juga dikenal sebagai *Computer Based Test* (CBT), meliputi penilaian, pengujian dengan menggunakan komputer adalah suatu metode pengelolaan dan pelaksanaan tes yang dilakukan secara elektronik kemudian dicatat, dinilai, atau keduanya. Seperti namanya, Penilaian Berbasis Komputer (PBK) memanfaatkan komputer atau perangkat elektronik setara seperti ponsel atau PDA, sistem PBK memungkinkan pendidik dan pelatih untuk menulis jadwal, menyampaikan laporan tentang *survey*, kuis, tes dan ujian.

Keuntungan sistem PBK dibanding dengan sistem konvensional dengan kertas dan pensil tradisional adalah menyederhanakan administrasi dan efisiensi, mengurangi biaya untuk berbagai elemen siklus pengujian, uji keamanan juga membaik akibat elektronik transmisi dan enkripsi, konsistensi dan kehandalan, lebih cepat dan lebih baik proses uji revisi serta dikendalikan dengan waktu respon lebih singkat, lebih cepat pengambilan keputusan sebagai hasil dari penilaian langsung dan pelaporan hasil ujian. Hal ini juga dipertimbangkan sebagai penilaian formatif khususnya berbasis komputer, BPK akan memainkan peran yang semakin penting dalam proses belajar dengan meningkatnya penggunaan bank soal.

1. **Tes Potensi Akademik (TPA)**

Tes potensi akademik adalah suatu tes yang bertujuan untuk mengukur bakat dan kemampuan seseorang dibidang akademik. Saat ini, selain tes potensi akademik dijadikan salah satu tes standar penerimaan SNMPTN dan perguruan tinggi untuk jenjang S2 dan S3, TPA juga telah menjadi tes standar penyaringan Calon Pegawai Negeri Sipil (CPNS).

1. **Materi Tes Potensi Akademik**

Materi yang diujikan dalam tes potensi akademik identik dengan tes GRE(*Graduate Record Examination*). GRE adalah sebuah tes yang juga mengukur kemampuan dan bakat seseorang dibidang akademik yang menjadi standar internasional penerimaan mahasiswa perguruan tinggi di Amerika Serikat. Untuk mendapatkan informasi lebih jauh tentang tes GRE ini bisa diakses di situs [www.ets.org/gre](http://www.ets.org/gre).

Dalam tes potensi akademik terdapat empat bidang kemampuan yang diujikan, yaitu :

1. Tes kemampuan verbal (bahasa)

Tes ini bertujian menguji kemampuan seseorang dibidang kata dan bahasa.

1. Tes kemampuan numerik (angka)

Merupakan tes yang bertujuan untuk menguji kemampuan seseorang di bidang aritmetik, kemampuan berpikir terstruktur untuk melihat seri angka, seri huruf.

1. Tes kemampuan penalaran (logika)

Tes ini menguji kemampuan seseorang dalam hal penalaran dan pemecahan persoalan secara logis (masuk akal).

1. Tes kemampuan spasial (gambar)

Tes ini bertujian menguji kecerdasan keruangan (spasial) yang dimiliki seseorang.

1. **Algoritma**

Menurut *Donald E. Knuth* (1973), algoritma dalam pengertian modern mempunyai kemiripan dengan istilah resep, proses, metode, teknik, prosedur, rutin. Algoritma adalah sekumpulan aturan-aturan berhingga yang memberikan sederetan operasi-operasi untuk menyelesaikan suatu jenis masalah yang khusus.

1. **Algoritma *Linear Congruential Generators* (LCG)**

*Linear Congruential Generators* (LCG) yang diperkenalkan oleh Donald E. Knut merupakan pembangkit bilangan acak yang sederhana, mudah dimengerti teorinya, dan juga mudah untuk diimplementasikan.

LCG didifinisikan dalam relasi berulang berikut :

Xn=(AXn-1 + B) *mod* M

dimana :

Xn = bilangan acak ke-n dari deretnya

Xn-1 = bilangan acak sebelumnya

A = faktor pengali

B = *increment*

M = *Modulus*

X0 adalah kunci pembangkit atau disebut juga umpan (*seed*)

Periode LCG paling besar adalah M bahkan pada kebanyakan kasus periodenya kurang dari M. Maksudnya adalah deret bilangan acak yang dihasilan tidak lebih banyak dari modulusnya.

LCG akan memiliki periode penuh jika memenuhi syarat sebagai berikut:

1. B relatif prima terhadap M
2. A-1 dapat dibagi dengan semua faktor prima dari M
3. A-1 kelipatan 4 jika M kelipatan 4
4. Nilai M lebih besar dari max(A,B,X0)
5. A > 0 dan B > 0

Algoritma ini dipilih sebagai dasar karena algoritma ini lebih cepat dalam pemrosesan dibanding dengan algoritma acak lainnya seperti *Blum Blum Shub* dan *Mersenne Twister* sesuai dengan hasil penelitian oleh Andresta Ramadhan (2006) dengan judul Perbandingan Algoritma *Linear Congruential Generators, Blum Blum Shub,* dan *Mersenne Twister* untuk Membangkitkan Bilangan Acak Semu.

**BAB III**

**ANALISIS DAN PERANCANGAN SISTEM**

1. **Analisis Sistem**

Analisis adalah kajian yang dilaksanakan untuk mengetahui lebih jauh tentang obyek penelitian secara lebih mendalam. Harapannya akan diperoleh gambaran umum mengenai analisis kebutuhan yang berupa informasi serta pemodelan yang digunakan untuk membangun sistem.

1. **Analisis Soal**

Soal yang akan diujikan didalam CBT adalah soal tes potensi akademik yang mencakup empat bidang yaitu pertama tes kemapuan verbal yang mencakup sinonim, anonim, dan padanan hubungan kata. Kedua tes kemampuan numerik yang mencakup deret angka, deret huruf dan kemampuan teknikal. Ketiga tes kemampuan logika yang mencakup tes logika analisis. Dan keempat adalah tes kemampuan spasial atau gambar yang mencakup tes seri gambar. Tiap bidang mempunyai pengaturan waktu yang berbeda-beda dalam pengerjaan dan mempunyai jumlah soal yang berbeda pula. Contoh soal verbal bisa berjumlah 15 dengan waktu pengerjaan 10 menit, sedangkan soal logika berjumlah 8 dengan waktu 8 menit.

1. **Analisis *Input*, Proses dan *Output***

Aplikasi ini secara khusus bertujuan untuk mengkomputerisasi pelaksanaan ujian PMB dengan metode *client-server* dimana *server* sebagai penyedia layanan dalam memproses permintaan *client*, *server* melakukan pengacakan soal dan mengirim ke *client* serta melakukan pengecekan jawaban sampai mengirimkan hasil ujian ke *client*.

Berikut ini adalah analisis kebutuhan yang meliputi *input*, proses, dan *output*:

1. Kebutuhan *input*

*Input* akan dikategorikan berdasarkan *client* dan *server* sebagai berikut:

1. *Server*

Masukan awal pada *server* adalah berupa menerima nomor pendaftaran dan pengecekan sehingga mendapatkan semua informasi tentang data pendaftar.

1. *Client*

Pendaftar sebagai client memasukan nomor pendaftaran pada aplikasi.

1. Proses akan berupa poin-poin seperti berikut :
2. Pengecekan nomor pendaftaran
3. Proses pengambilan soal dari *database*
4. Proses pengacakan soal
5. Proses pengaturan waktu
6. Proses menampilkan soal
7. Proses menerima jawaban soal
8. Proses kalkulasi hasil ujian
9. Simpan hasil ujian
10. *Output*

*Output* berupa hasil ujian muncul setelah peserta menyelesaikan ujian atau setelah waktu yang ditentukan sudah selesai.

1. **Perangkat Lunak**

Perangkat lunak yang digunakan dalam pembuatan aplikasi ini antara lain adalah:

1. JDK (*Java Development Kit*) versi 6
2. *Intellij IDEA* 11.0.2
3. *PostgreSQL* 8.2
4. *Apache Maven* 3.0
5. *Apache Tomcat* 7.0
6. **Perangkat Keras**

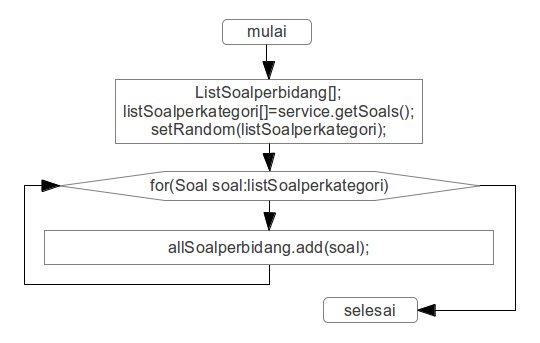
Perangkat keras yang dibutuhkan untuk membangun sistem aplikasi ini adalah :

1. Layar LCD *Compaq* 14”
2. Prosesor *intel core* 2 *Duo*
3. Harddisk 160 GB
4. RAM DDR 2 2048 MB
5. **Analisis Algoritma**

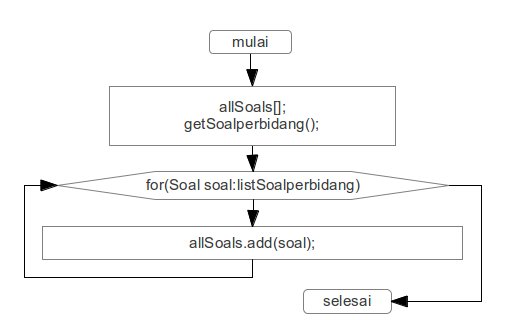
Dibawah akan dibahas berbagai alur algoritma dalam mendukung aplikasi CBT yang dipakai untuk PMB di Sekolah Tinggi Teknologi Adisutjipto mulai dari bagaimana pengambilan soal, pengacakan soal, serta algoritma mendapatkan nilai akhir.

1. **Algoritma Pengambilan Soal**

Dalam aplikasi CBT PMB STTA soal diambil dari *database* dengan jumlah soal per bidang tergantung dari *setting* yang dibuat oleh *admin*. Pertama soal diambil per bidang dan per kategori. Setelah itu diacak dan dimasukan ke dalam *list* bidang masing masing dan kemudian setelah *list* bidang semua terisi lalu dimasukan ke dalam *list* yang menampung semua soal.



Gambar 3.1 *Flowchart* Ambil Soal Per Bidang



Gambar 3.2 *Flowchart* Pengambilan Soal

Keterangan :

Gambar 3.1 menerangkan bagaimana soal diambil per bidang dengan cara mengambil jumlah soal per kategori dari tabel *setting* dan mengambil soal sebanyak jumlah soal per kategori kemudian diacak dan di dimasukan ke dalam *list* per bidang. Gambar 3.2 menjelaskan setelah *list* per bidang semuanya terisi kemudah fungsi untuk mengambil soal per bidang dipanggil dan semua *list* per bidang terisi data kemudian data per bidang dimasukan ke dalam listAllSoals untuk kemudian disimpan dan tampilkan di dalam tampilan soal CBT.

1. **Alur Algoritma Acak *Linear Congruential Generators* (LCG)**

*Linear congruential generators* dapat didefinisikan sebagai berikut:

Xn = ( AXn-1 + B ) *mod* M

Dimana :

Xn = bilangan acak ke-n dari deret

Xn-1 = bilangan acak sebelumnya

A = faktor pengali

B = *increment*

M = *modulus*

Dan ada satu lagi yaitu X0 sebagai kunci pembangkit atau disebut juga umpan(*seed*).

Rumus diatas dapat diterapkan pada contoh dibawah ini yang nantinya akan menghasilkan bilangan acak :

Misal untuk X1n A=5, B=13, M=23 dan X0=0

Tabel 3.1 Contoh Deret

|  |  |  |
| --- | --- | --- |
| **n** | **X1n = ( 5\*X1n-1 + 13 ) *mod* 23** | **X1n** |
| 0 | X1 0 | 0 |
| 1 | X1 1 = ( 5\*0 + 13 ) *mod* 23 | 13 |
| 2 | X1 2 = ( 5\*13 + 13 ) *mod* 23 | 9 |
| 3 | X1 3 = ( 5\*9 + 13 ) *mod* 23 | 12 |
| 4 | X1 4 = ( 5\*12 + 13 ) *mod* 23 | 4 |
| 5 | X1 5 = ( 5\*4 + 13 ) *mod* 23 | 10 |
| 6 | X1 6 = ( 5\*10 + 13 ) *mod* 23 | 17 |
| 7 | X1 7 = ( 5\*17 + 13 ) *mod* 23 | 6 |
| 8 | X1 8 = ( 5\*6 + 13 ) *mod* 23 | 20 |
| 9 | X1 9 = ( 5\*20 + 13 ) *mod* 23 | 21 |
| 10 | X1 10 = ( 5\*21 + 13 ) *mod* 23 | 3 |
| 11 | X1 11 = ( 5\*3 + 13 ) *mod* 23 | 5 |
| 12 | X1 12 = ( 5\*5 + 13 ) *mod* 23 | 15 |
| 13 | X1 13 = ( 5\*15 + 13 ) *mod* 23 | 19 |
| 14 | X1 14 = ( 5\*19 + 13 ) *mod* 23 | 16 |
| 15 | X1 15 = ( 5\*16 + 13 ) *mod* 23 | 1 |
| 16 | X1 16 = ( 5\*1 + 13 ) *mod* 23 | 18 |
| 17 | X1 17 = ( 5\*18 + 13 ) *mod* 23 | 11 |
| 18 | X1 18 = ( 5\*11 + 13 ) *mod* 23 | 22 |
| 19 | X1 19 = ( 5\*22 + 13 ) *mod* 23 | 8 |
| 20 | X1 20 = ( 5\*8 + 13 ) *mod* 23 | 7 |
| 21 | X1 21 = ( 5\*7 + 13 ) *mod* 23 | 2 |
| 22 | X1 22 = ( 5\*2 + 13 ) *mod* 23 | 0 |

Dari contoh diatas didapatkan deret bilangan acak {0,13,9,12,4,10,17,6,20,21,3,5,15,19,16,1,18,11,22,8,7,2,0}.

1. **Algoritma Acak pada CBT PMB STTA**

Algoritma acak yang dipakai pada CBT PMB STTA adalah pengembangan dari algoritma LCG dengan menetapkan banyaknya soal sebagai *modulus* (M). Dan *time-of-day* (waktu saat ini) dalam *long* sebagai umpan (*seed*). Untuk memenuhi syarat penuh satu *periode* maka juga akan digunakan nilai A(faktor pengali) dan B(*increment*) yang konstan. Nilai-nilai tersebut sudah diuji dan dipakai dalam bahasa pemrograman *C* serta *Java*.

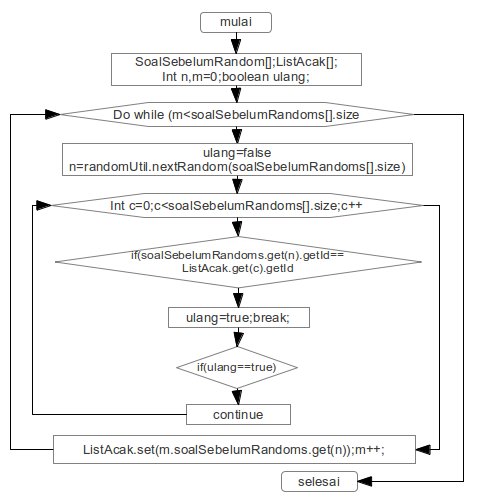
Rumus acak yang dipakai dalam CBT PMB STTA adalah :

Val n=(((*multiplier*\**oldseed* + *addend*) & *mask*) >>> 17) *mod* n

Dimana :

1. *Mulitplier* adalah A(faktor pengali) yang bernilai konstan 0x5DEECE66DL atau 25214903917 dalam desimal.
2. *Oldseed* adalah Val n-1 atau nilai umpan yang dihasilkan sebelumnya.
3. *Addend* adalah B(*increment*) yang bernilai konstan 0xBL atau 11 dalam desimal.
4. *Mask* bernilai konstan (1L <<48)-1
5. 17 didapat dari (48 *bit* – 31 *bit*) berguna untuk menaikan *bit* dan masih dalam jangkauan *integer*. *Bit* tinggi akan lebih acak dari pada *bit* rendah.
6. Rumus diatas akan mengembalikan nilai antara 0 (*inclusive*) sampai n (*exclusive*).
7. **Algoritma Acak Soal pada Aplikasi**

Soal yang sudah didapat dari *database* yang berjumlah sudah ditentukan dari tabel *setting* akan akan diacak didalam sebuah *method* yang bernama setRandomSoal. Disini nantinya soal yang sudah terkumpul secara urut akan di acak menggunakan algoritma acak dengan cara mencari nilai acak lalu nilai acak tersebut digunakan untuk mencari indek *list* soal dan dipindahkan ke *list* yang lain yang sudah disediakan untuk menampung soal acak. *Flowchart* acak soal dapat dilihat pada gambar 3.3.



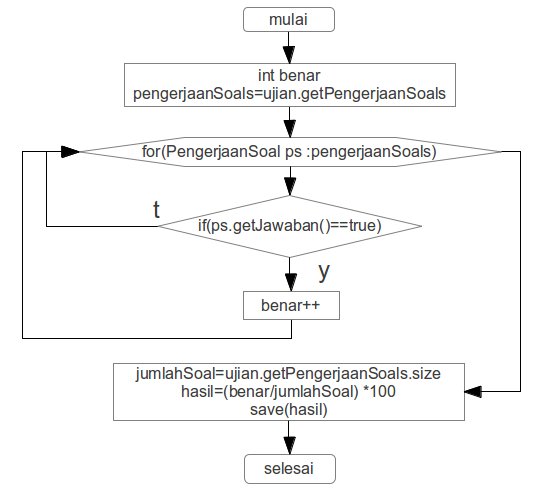
Gambar 3.3 *Flowchart* Acak Soal

Keterangan :

Pertama kali soalSebelumRandoms adalah *variable* *array* yang menampung soal dari *database*. Setelah itu disediakan *array* kosong dengan nama listAcak, *int* m, n dan *boolean* ulang adalah *variable* pendukung. Setelah itu dilakukan perlangan menggunakan *do while* yang berisi pertama ulang di isi *false*, setelah itu n diisi dengan bilangan hasil acak. Setelah itu dilakukan perulangan apakah id indek n pada soalSebelumRandoms sama dengan id indek pada listAcak, bila ya maka ulang diset *true* dan perulangan dilakukan kembali, bila tidak maka id indek n pada soalSebelumRandom akan dipindahkan ke indek m pada listAcak. Setelah itu m *increment* satu. Dan berulang terus menerus selama nilai m kurang dari *size* soalSebelumRandoms. Setelah perulangan selesai maka acak soal selesai dan diteruskan oleh fungsi yang lain.

1. **Algoritma Perhitungan Hasil Ujian**

Hasil ujian keluar ketika peserta menekan tombol selesai atau karena waktu sudah habis dan setelah itu maka sistem akan melakukan pengecekan berapa jawaban yang benar lalu dibagi dengan jumlah soal dan dikalikan 100. Setelah itu disimpan ditabel ujian. *Flowchart* pengecekan hasil ujian dapat dilihat pada gambar 3.4.



Gambar 3.4 *Flowchat* Hasil Ujian

Keterangan:

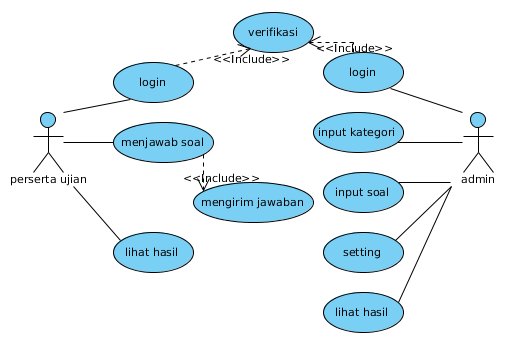
Pertama disediakan *variable* benar untuk menampung jumlah jawaban yang benar. pengerjaanSoals menampung hasil pengerjaan soal. Setelah itu di lakukan perulangan sebanyak pengerjaanSoals. Dicek bila pengerjaan soal benar maka *variable* benar akan bertambah nilai satu, dan bila tidak maka akan dilewati. Setelah perulangan selesai sistem akan menghitung hasil ujian dengan rumus benar dibagi jumlah soal dikali 100. Setelah itu disimpan ke tabel ujian. Proses pengecekan soal selesai.

1. **Perancangan Sistem**

Perancangan sistem ini menjelaskan pemodelan sistem algoritma acak dan perancangan *interface* yang digunakan pada aplikasi.

1. **Diagram *Use Case***

Diagram *Use Case*, menggambarkan interaksi antara sistem yang sedang diteliti dengan sistem diluarnya dan dengan *user*. Dengan kata lain, *use case* melukiskan siapa yang akan menggunakan sistem dan dengan cara bagaimana *user* berharap dapat berinteraksi dengan sistem itu. *Use case* selalu dilengkapi dengan narasi yang menjelaskan urutan langkah dari setiap interaksi. Dalam aplikasi ini diagram *use case* dapat dilihat pada gambar 3.5.

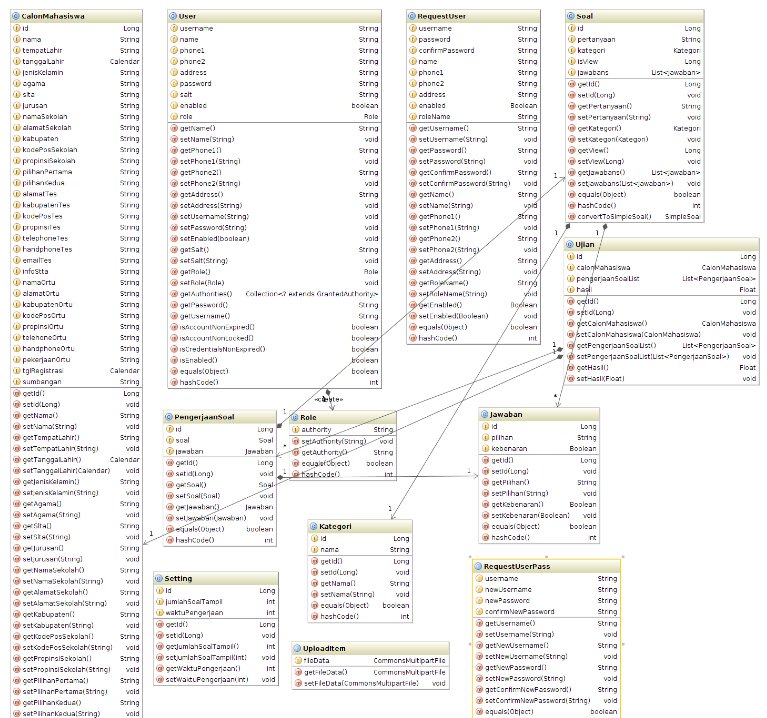


Gambar 3.5 Diagram *Use Case*

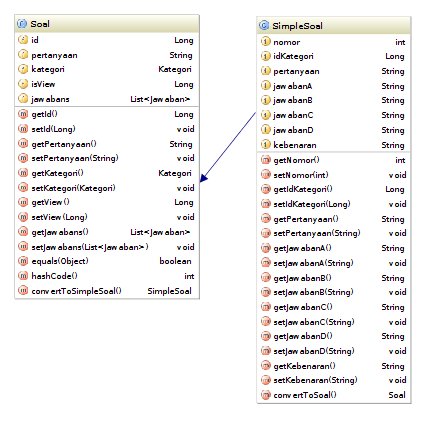
Pada gambar 3.5 diagram *use case* diatas dijelaskan bahwa *actor* terdiri dari *client* dan *server*, dimana *server* berinteraksi dengan aplikasi pertama kali dengan *login* untuk menghindari penyusupan pada bagian *server*, kemudian setelah *login* baru *server* dapat memasukkan soal. *Actor* *client* (pengguna) berinteraksi dengan aplikasi pertama kali login memasukkan nomor pendaftaran dan bila verifikasi berhasil maka soal akan muncul dan *client* dapat memulai menjawab soal, disetiap jawaban aplikasi akan mengirimkan data ke *server* untuk disimpan dan pada akhir proses *client* akan dapat melihat nilai yang didapatkan.

1. **Diagram *Class***

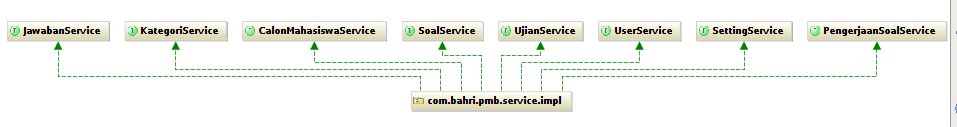
Diagram *class* adalah diagram yang digunakan untuk menampilkan beberapa *class* serta *package* yang ada dalam sistem yang akan dibangun. Diagram ini menunjukkan hubungan antara *class* dalam sistem tentang bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Pada aplikasi ini, struktur *class* membantu dalam visualisasi dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Diagram *class* juga memperlihatkan hubungan antar *class* dan penjelasan detail tiap-tiap *class* yang didalamnya terdapat objek dan atribut. Diagram *class* dapat dilihat pada gambar 3.6 , gambar 3.7 , gambar 3.8, gambar 3.9, gambar 3.10.



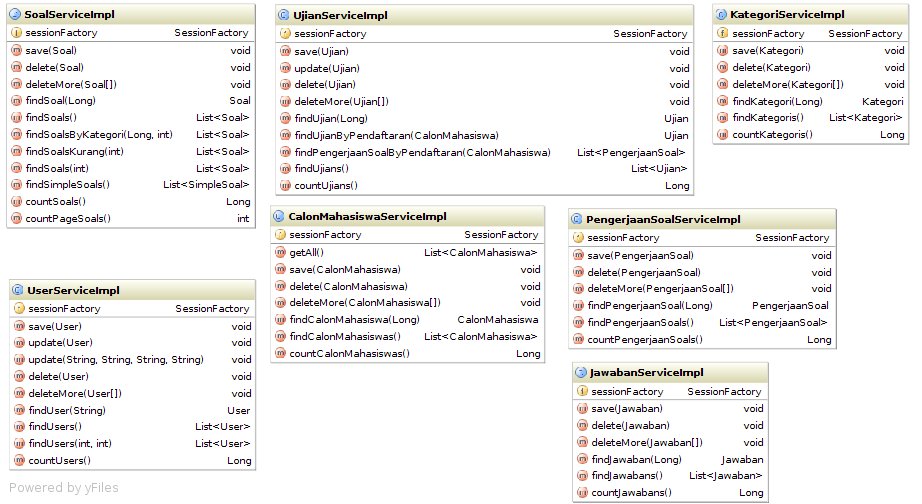
Gambar 3.6 Diagram *Class Package Domain*



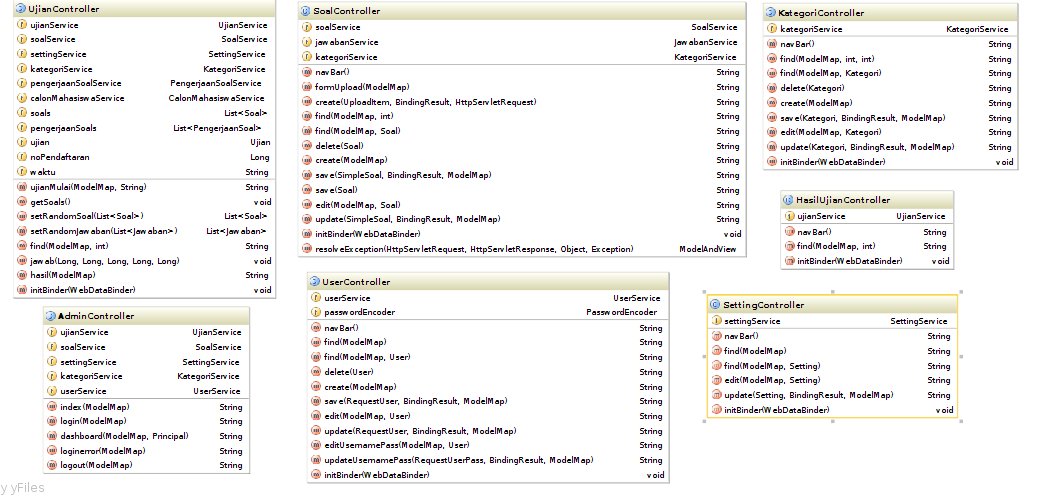
Gambar 3.7 Turunan *Class* Soal



Gambar 3.8 *Implement* *Package Service*



Gambar 3.9 Diagram *Class Package* *Impl*



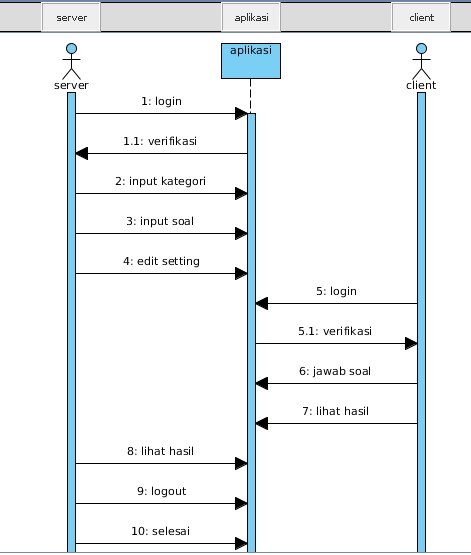
Gambar 3.10 Diagram *Class Package Controller*

*Class* adalah sesuatu yang membungkus (*encapsulate*) informasi dan perilaku. Dalam *Object Oriented Programing* (OOP) akan digambarkan dalam tiga area pokok yaitu nama *class*, atribut dan *method*. Gambar menjelaskan perancangan diagram *class* untuk aplikasi CBT dimana *package* *domain* berisi *class* yang merepresentasikan tabel yang berada pada *database* serta *class* pendukung aplikasi. Ada satu *class* yang *extend* kepada *class* Soal yaitu *class* SimpleSoal. *Package* impl berisi *class* yang berfungsi sebagai kumpulan *transaction* untuk melakukan fungsi *insert* *update* *delete* serta *get* dan mengimplemen pada *class* yang berada di *package* *service*. *Package* *controller* adalah berisi *class* yang berfungsi sebagai pengatur tampilan atau *view* menerima data dari *view* dan memprosesnya untuk di simpan pada *database* melalui *class* yang berada di *package* *impl*.

1. **Diagram *Sequence***

Diagram ini menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Dalam diagram ini secara khusus berasosiasi dengan *use case*. Diagram ini juga memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case.*

Tipe diagram ini sebaiknya digunakan diawal tahap desain dan analisis karena kesederhanaannya dan mudah untuk dimengerti. Diagram *sequence* dapat dilihat pada gambar 3.11.

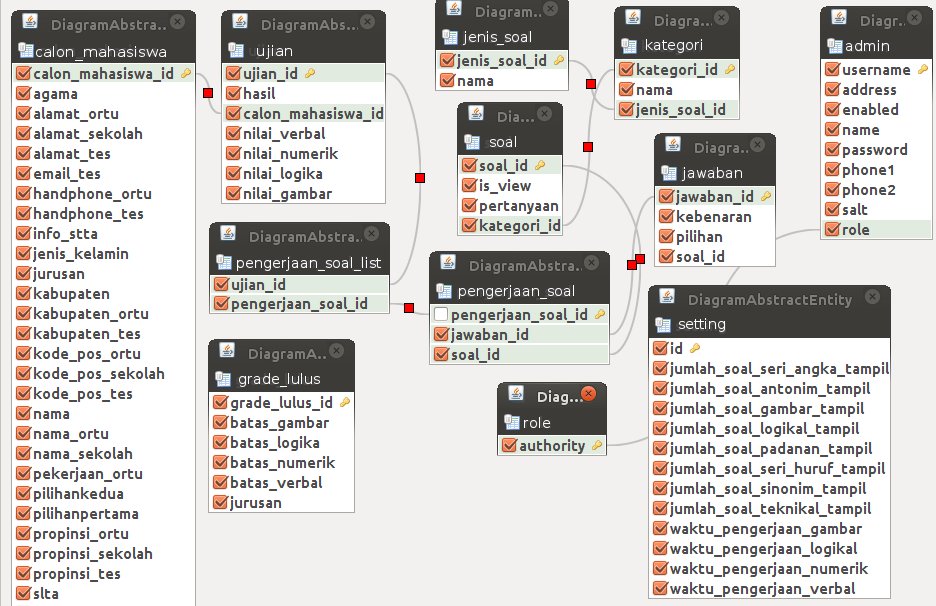


Gambar 3.11 Diagram *Sequence*

Urutan proses yang terjadi pada pengaksesan yang dilakukan oleh pengguna baik *server* maupun *client* yaitu, *server* harus *login* terlebih dahulu, hanya bisa menginputkan kategori, soal, serta melakukan *setting* banyak soal dan lama waktu pengerjaan soal. Setelah soal dimasukan baru *client* dapat menjawab soal-soal namun diharuskan untuk *login* menggunakan nomor pendaftaran terlebih dahulu. Setelah *login* *client* dapat menjawab soal sampai selesai dan dikirim pada *server*. Setelah selesai mengerjakan soal maka *client* dapat langsung melihat hasil pengerjaan. Disisi *server* dapat melihat nilai hasil pengerjaan peserta. Kemudian terakhir *server* bisa *logout*.

1. **Perancangan *Database***

*Database* yang digunakan untuk aplikasi ini adalah *Postgresql 8.2*. Perancangan *database* menggunakan pemodelan ER(*entity relationship*) dalam bentuk tabel, untuk menggambarkan dalam hubungan antar tabel dapat dilihat pada gambar 3.12.



Gambar 3.12 Relasi Tabel CBT

Keterangan :

1. Relasi tabel calon\_mahasiswa dengan tabel ujian adalah *one to one* (1 calon mahasiswa hanya bisa melakukan satu kali ujian)
2. Relasi tabel soal dengan tabel kategori adalah *many to one* ( satu kategori bisa mempunyai banyak soal)
3. Relasi tabel soal dengan tabel jawaban adalah *one to many* (satu soal bisa mempunyai banyak jawaban)
4. Relasi tabel ujian dengan tabel pengerjaan\_soal adalah *one to* *many* dengan tabel pengerjaan\_soal\_list sebagai tabel penghubung ( satu kali ujian bisa mempunyai banyak pengerjaan soal)
5. Relasi tabel *admin* dengan tabel role adalah *many to one* ( satu *role* bisa mempunyai banyak *admin*)
6. Tabel *setting* tidak berelasi dengan tabel manapun karena fungsinya hanya menyimpan setting CBT berupa jumlah soal yang ditampilkan dan lama waktu pengerjaan.
7. **Normalisasi (Tahapan Normalisasi)**
8. Bentuk Normal Pertama (*1NF*)

Sebuah model data dikatakan memenuhi bentuk normal pertama apabila setiap atribut yang dimilikinya memiliki satu dan hanya satu nilai. Apabila ada atribut yang memiliki nilai lebih dari satu, atribut tersebut adalah kandidat untuk menjadi entitas tersendiri.

1. Bentuk Normal Kedua (2NF)

Sebuah model data dikatakan memenuhi bentuk normal kedua apabila ia memenuhi bentuk normal pertama dan setiap atribut non-*identifier* sebuah entitas bergantung sepenuhnya hanya pada semua *identifier* entitas tersebut.

1. Bentuk Normal Ketiga (3NF)

Sebuah model data dikatakan memenuhi bentuk normal ketiga apabila ia memenuhi bentuk normal kedua dan tidak ada satupun atribut *nonidentifying* (bukan pengidentifikasi unik) yang bergantung pada atribut *non-identifying* lain. Apabila ada, pisahkan salah satu atribut tersebut menjadi entitas baru, dan atribut yang bergantung padanya menjadi atribut entitas baru tersebut.

1. Tabel *Admin*

Tabel 3.2 *Admin*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| *Username* | *Character varying(50)* | *Not null* | *Primery* |
| *Address* | *Character varying(255)* | *Not null* | *-* |
| *Enabled* | *Boolean* | *Not null* | *-* |
| *Name* | *Character varying(100)* | *Not null* | *-* |
| *Password* | *Character varying(255)* | *No null* | *-* |
| *Phone1* | *Character varying(20)* | *-* | *-* |
| *Phone2* | *Character varying(20)* | *-* | *-* |
| *Salt* | *Character varying(255)* | *Not null* | *-* |
| *Role* | *Character varying(50)* | *Not null* | *Foreign key* |

Analisis normalisasi tabel *admin* :

1. Tabel *admin* sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel *admin* yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel *admin* juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci primer.
3. Tabel *admin* sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci primer.
4. Tabel Calon\_mahasiswa

Tabel 3.3 Calon\_mahasiswa

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| calon\_mahasiswa\_id | *Bigint* | *Not null* | *primery* |
| agama | *character varying(25)* | *Not null* | *-* |
| alamat\_ortu | *character varying(100)* | *Not null* | *-* |
| alamat\_sekolah | *character varying(100* | *Not null* | *-* |
| alamat\_tes | *character varying(100* | *-* | *-* |
| email\_tes | *character varying(50)* | *-* | *-* |
| handphone\_ortu | *character varying(20)* | *-* | *-* |
| handphone\_tes | *character varying(20* | *-* | *-* |
| info\_stta | *character varying(100)* | *-* | *-* |
| jenis\_kelamin | *character varying(20)* | *Not null* | *-* |
| jurusan | *character varying(15)* | *Not null* | *-* |
| kabupaten | *character varying(100)* | *Not null* | *-* |
| kabupaten\_ortu | *character varying(100)* | *Not null* | *-* |
| kabupaten\_tes | *character varying(100)* | *-* | *-* |
| kode\_pos\_ortu | *character varying(10)* | *Not null* | *-* |
| kode\_pos\_sekolah | *character varying(10)* | *Not null* | *-* |
| kode\_pos\_tes | *character varying(10* | *-* | *-* |
| nama | *character varying(30)* | *Not null* | *-* |
| nama\_ortu | *character varying(100)* | *Not null* | *-* |
| nama\_sekolah | *character varying(100)* | *Not null* | *-* |
| pekerjaan\_ortu | *character varying(100)* | *Not null* | *-* |
| pilihankedua | *character varying(50)* | *Not null* | *-* |
| pilihanpertama | *character varying(50)* | *Not null* | *-* |
| propinsi\_ortu | *character varying(100)* | *Not null* | *-* |
| propinsi\_sekolah | *character varying(30)* | *Not null* | *-* |
| propinsi\_tes | *character varying(100)* | *-* | *-* |

Analisis normalisasi tabel calon\_mahasiswa :

1. Tabel calon\_mahasiswa sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel calon\_mahasiswa yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel calon\_mahasiswa juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel calon\_mahasiswa sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Jawaban

Tabel 3.4 Jawaban

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***key*** |
| Jawaban\_id | *Bigint* | *Not null* | *Primary* |
| Kebenaran | *Boolean* | *Not null* | *-* |
| Pilihan | *Text* | *Not null* | *-* |
| Soal\_id | *Bigint* | *Not null* | *Foreign* |

Analisis normalisasi tabel jawaban :

1. Tabel jawaban sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel jawaban yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel jawaban juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel jawaban sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Kategori

Tabel 3.5 Kategori

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***key*** |
| Kategori\_id | *Bigint* | *Not null* | *Primary* |
| Nama | *character varying(50)* | *Not null* | *-* |
| Jenis\_soal\_id | *Bigint* | *Not null* | *foreign* |

Analisis normalisasi tabel kategori :

1. Tabel kategori sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel kategori yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel kategori juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel kategori sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Pengerjaan\_soal

Tabel 3.6 Pengerjaan\_soal

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Pengerjaan\_soal\_id | *Bigint* | *Not null* | *Primary* |
| Jawaban\_id | *Bigint* | *-* | *Foreign* |
| Soal\_id | *Bigint* | *-* | *foreign* |

Analisis normalisasi tabel pengerjaan\_soal :

1. Tabel pengerjaan\_soal sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel pengerjaan\_soal yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel pengerjaan\_soal juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel pengerjaan\_soal sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel *Role*

Tabel 3.7 *Role*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| *Authority* | *character varying(50)* | *Not null* | *primary* |

Analisis normalisasi tabel *role* :

1. Tabel *role* sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel *role* yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel *role* juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel *role* sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel *Setting*

Tabel 3.8 *Setting*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Id | *Bigint* | *Not null* | *Primary* |
| Jumlah\_soal\_seri\_angka\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_antonim\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_gambar\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_logikal\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_padanan\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_seri\_huruf\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_sinonim\_tampil | *Integer* | *Not null* | *-* |
| Jumlah\_soal\_teknikal\_tampil | *Integer* | *Not null* | *-* |
| Waktu\_pengerjaan\_gambar | *Integer* | *Not null* | *-* |
| Waktu\_pengerjaan\_logikal | *Integer* | *Not null* | *-* |
| Waktu\_pengerjaan\_numerik | *Integer* | *Not null* | *-* |
| Waktu\_pengerjaan\_verbal | *Integer* | *Not null* | *-* |

Analisis normalisasi tabel *setting* :

1. Tabel *setting* sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel *setting* yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel *setting* juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel *setting* sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Soal

Tabel 3.9 Soal

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Soal\_id | *Bigint* | *Not null* | *Primary* |
| *Is\_view* | *Bigint* | *-* | *-* |
| Pertanyaan | *Text* | *Not null* | *-* |
| Kategori\_id | *Bigint* | *Not null* | *foreign* |

Analisis normalisasi tabel soal :

1. Tabel soal sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel soal yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel soal juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel soal sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Pengerjaan\_soal\_list

Tabel 3.10 Pengerjaan\_soal\_list

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Ujian\_id | Bigint | Not null | - |
| Pengerjaan\_soal\_id | Bigint | Not null | - |

Analisis normalisasi tabel pengerjaan\_soal\_list :

1. Tabel pengerjaan\_soal\_list sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel pengerjaan\_soal\_list yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel pengerjaan\_soal\_list juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel pengerjaan\_soal\_list sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Ujian

Tabel 3.11 Ujian

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Ujian\_id | *Bigint* | *Not null* | *Primary* |
| Hasil | *Real* | *-* | *-* |
| Calon\_mahasiswa\_id | *Bigint* | *Not null* | *foreign* |
| Nilai\_verbal | *Real* | *-* | *-* |
| Nilai\_numerik | *Real* | *-* | *-* |
| Nilai\_logika | *Real* | *-* | *-* |
| Nilai\_gambar | *Real* | *-* | *-* |

Analisis normalisasi tabel ujian :

1. Tabel ujian sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel ujian yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel ujian juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel ujian sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Grade\_lulus

Tabel 3.12 Grade\_lulus

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Grade\_lulus\_id | *Bigint* | *Not null* | *Primary* |
| Batas\_gambar | *Real* | *Not null* | *-* |
| Batas\_logika | *Real* | *Not null* | *-* |
| Batas\_numerik | *Real* | *Not null* | *-* |
| Batas | *Real* | *Not null* | *-* |
| Jurusan | *Character varying(255)* | *Not null* | *-* |

Analisis normalisasi tabel grade\_lulus :

1. Tabel grade\_lulus sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel grade\_lulus yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel grade\_lulus juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel grade\_lulus sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. Tabel Jenis\_soal

Tabel 3.13 Jenis\_soal

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field*** | ***Type*** | ***Null*** | ***Key*** |
| Jenis\_soal\_id | Bigint | Not null | Primary |
| Nama | Character varying(255) | Not null | - |

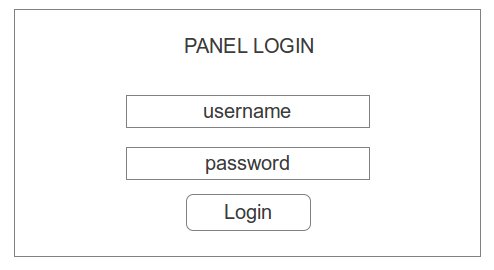
Analisis normalisasi tabel jenis\_soal :

1. Tabel jenis\_soal sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel jenis\_soal yang setiap atributnya bernilai tunggal untuk setiap barisnya.
2. Tabel jenis\_soal juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.
3. Tabel jenis\_soal sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.
4. **Rancangan Tampilan**

Rancangan tampilan pada aplikasi ini dibagi menjadi dua yaitu pada sisi *client* dan sisi *server* dan berikut sebagai rancangannya :

1. Rancangan *server*
2. Rancangan *panel login server admin*

*Panel Login admin* akan berfungsi sebagai tempat admin memasukan *username* dan *password* guna masuk pada *dashboard* aplikasi *server*. Rancangan dapat dilihat pada gambar 3.13.



Gambar 3.13 Login *Admin*

1. Rancangan menu utama *admin*

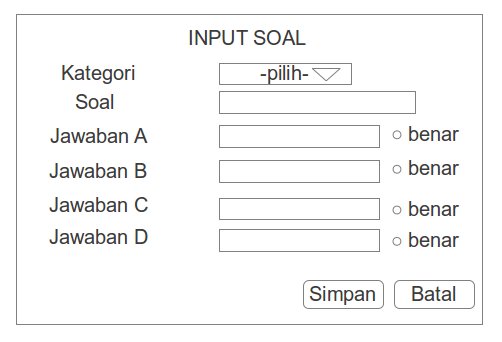
Menu utama akan berisi sebelah kiri menu dan sebelah kanan adalah konten dari tiap-tiap menu. Rancangan dapat dilihat pada gambar 3.14.



Gambar 3.14 Menu Utama *Admin*

1. Rancangan *input* soal

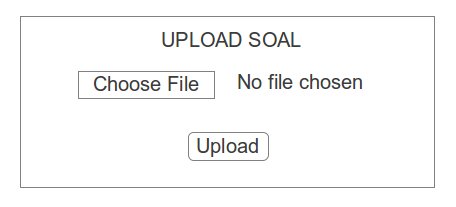
*Input* soal akan berupa tampilan *form* yang berfungsi sebagai tempat menulis soal. Rancangan dapat dilihat pada gambar 3.15.



Gambar 3.15 *Input* Soal

1. Rancangan *upload* soal

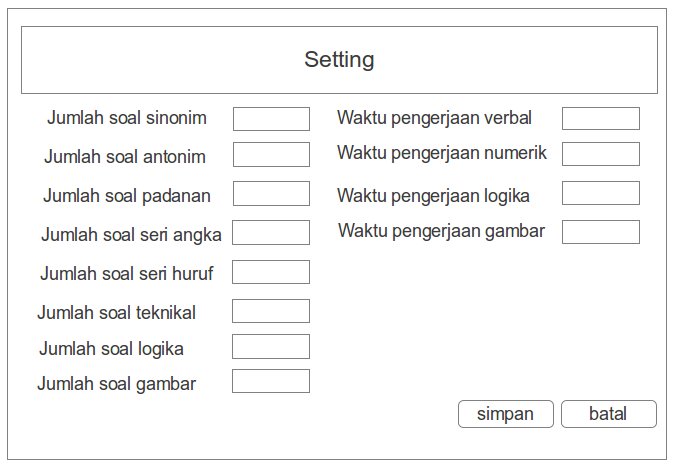
*Upload* soal berupa *form* untuk mencari *file* yang akan di *upload* ke aplikasi. Rancangan dapat dilihat pada gambar 3.16.



Gambar 3.16 *Upload* Soal

1. Rancangan *edit setting*

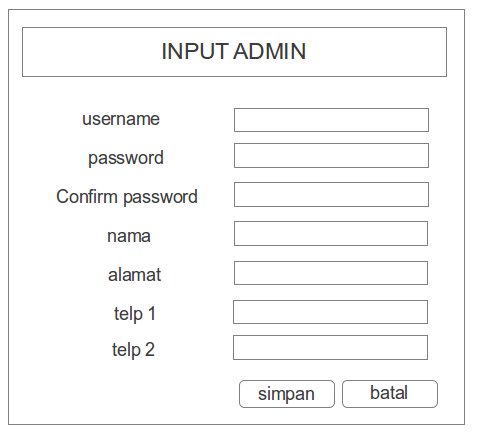
*Edit setting* akan berupa *form* untuk melakukan edit pada pengaturan soal seperti jumlah soal dan waktu pengerjaan. Rancangan dapat dilihat pada gambar 3.17.



Gambar 3.17 *Edit Setting*

1. Rancangan *input* *operator*

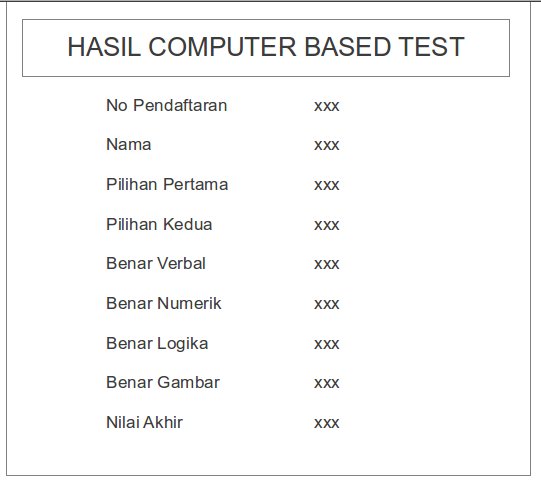
*Input operator* akan berisi *form* yang berfungsi sebagai *input* *operator* baru. Rancangan dapat dilihat pada gambar 3.18.



Gambar 3.18 *Input Operator*

1. Rancangan *print out* hasil

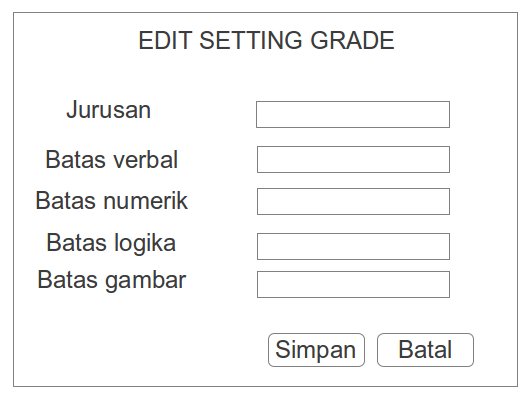
*Print out* hasil akan berupa data hasil ujian yang akan dicetak untuk dijadikan hasil dari CBT. Rancangan dapat dilihat pada gambar 3.19.



Gambar 3.19 *Print Out* Hasil

1. Rancangan *edit setting grade*

*Edit setting grade* akan berisi form yang berfungsi sebagai *editing grade* kelulusan tiap jurusan. Rancangan dapat dilihat pada gambar 3.20.



Gambar 3.20 *Edit Grade*

1. Rancangan client
2. Rancangan *panel login* peserta

*Panel login* peserta akan berisi *form input* nomor pendaftaran yang akan digunakan untuk login ke lembar soal. Rancangan dapat dilihat pada gambar 3.21.



Gambar 3.21 *Login* Peserta

1. Rancangan *panel* soal peserta

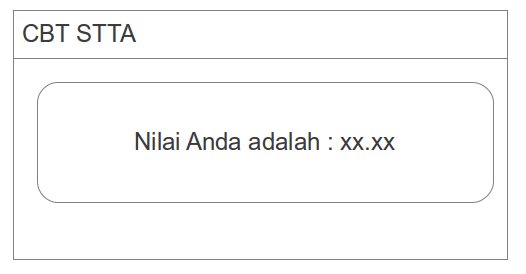
*Panel* soal peserta berisi pertanyaan dan jawaban serta waktu yang diberikan dan tombol untuk melanjutkan ke soal yang lain. Rancangan dapat dilihat pada gambar 3.22.



Gambar 3.22 *Panel* Soal

1. Rancangan hasil pengerjaan soal

Hasil pengerjaan soal akan berisi tampilan data hasil dari pengerjaan yang peserta lakukan setelah selesai melakukan ujian. Rancangan dapat dilihat pada gambar 3.23.



Gambar 3.23 Hasil Pengerjaan Soal

**BAB IV**

**IMPLEMENTASI DAN PEMBAHASAN SISTEM**

1. **Implementasi Sistem**

Aplikasi *Computer Based Test* (CBT) Penerimaan Mahasiswa Baru Sekolah Tinggi Teknologi Adisutjipto dilengkapi dengan algoritma acak soal yang dikembangkan dari algoritma *Linear Congruential Generators* (LCG) dikembangakan menggunakan bahasa pemrograman *Java* dengan teknologi *Java Enterprise Edition* (JEE). Berdasarkan analisis dan perancangan sistem yang telah dibuat pada bab tiga, maka implementasi merupakan tahap dimana algoritma dan sistem siap dioperasikan pada keadaan yang sebenarnya.

Pada implementasi sistem ini akan dijelaskan mengenai hasil implementasi dari aplikasi, baik berupa implementasi algoritma, tampilan *interface* aplikasi maupun potongan dari kode program.

1. **Implementasi Algoritma *Linear Congruential Generators***

Besarakan analisis dan perancangan sistem pada bab tiga, algoritma acak yang digunakan dengan rumus :

Val n=(((*multiplier*\**oldseed* + *addend*) & *mask*) >>> 17) *mod* n

Diimplementasikan ke dalam bentuk *sourcecode* sebagai berikut :

public class RandomUtil {

protected static final AtomicLong seed=new AtomicLong();

private static final long multiplier = 0x5DEECE66DL;

private static final long addend = 0xBL;

private static final long mask = (1L << 48) - 1;

public static int nextRandom(int n) {

int bits, val;

do {

bits = next(31);

val = bits % n;

} while(bits - val + (n-1) < 0);

return val;

}

protected static int next(int bits) {

long oldseed, nextseed;

AtomicLong seed = RandomUtil.seed;

do {

oldseed = seed.get();

if(oldseed==0){

seed.set(new GregorianCalendar().getTimeInMillis());

oldseed = seed.get();

}

nextseed = (oldseed \* multiplier + addend) & mask;

} while (!seed.compareAndSet(oldseed, nextseed));

return (int)(nextseed >>> (48 - bits));

}

}

Keterangan :

n disini adalah bilangan *exclusive* dari *random* atau sebagai batas atas. N akan diisi oleh banyaknya soal yang akan diacak. AtomicLog adalah *class* yang akan menyimpan umpan(*seed*). Pada method next(int bists) sudah disediakan *oldseed*, *nextseed* dan juga *seed* yang *reference* dari *class* AtomicLog untuk menyimpan *seed* yang baru atau disini adalah *nextseed*; kembalian dari *method* next(int bits) adalah berupa *integer* yang akan diterima oleh *bits* yang berada pada *method* nextRandom(int n). setelah *bits* terisi lalu di *mod* dengan n dan hasilnya dikembalikan keprogram yang memanggil *method* nextRandom(int n) pada *class* RandomUtil. Hasil kembalian adalah bilangan bulat dari 0 sampai n-1.

1. **Implementasi Algoritma Acak Soal dan Jawaban**

Berdasarkan *flowchart* acak soal pada gambar 3.2 yang terdapat pada bab tiga maka algoritma acak soal ini dapat diimplementasikan dalam *sourcecode* sebagai berikut :

public List<Soal> setRandomSoal(List<Soal> soalSebelumRandoms) {

List<Soal> listAcak = new ArrayList<Soal>();

Soal sb = new Soal();

for (int i = 0; i < soalSebelumRandoms.size(); i++) {

listAcak.add(sb);

}

boolean ulang = false;

int n = 0, m = 0;

do {

ulang = false;

n = RandomUtil.nextRandom(soalSebelumRandoms.size());

for (int c = 0; c < soalSebelumRandoms.size(); c++) {

if (soalSebelumRandoms.get(n).getId() == listAcak.get(c).getId()) {

ulang = true;

break;

}

}

if (ulang) {

continue;

}

listAcak.set(m, soalSebelumRandoms.get(n));

m++;

} while (m < soalSebelumRandoms.size());

return listAcak;

}

Keterangan :

Algoritma acak soal dan acak jawaban sama karena sama-sama mengacak urutan *list* yang soal dan jawaban miliki. Jadi pertama *list* sebelum diacak yang sudah terisi oleh data disiapkan, siapkan juga *list* kosong untuk menampung hasil acak. Setelah itu diulang sebanyak *list* yang sudah terisi. Kemudian cari nilai *random* menggunakan fungsi *random* yang ada pada penjelasan sebelumnya. Nilai ini yang akan digunakan untuk mencari indek *list* yang sudah terisi setelah itu dimasukan kedalam *list* yang baru dengan indek berurutan. Sehingga didapankan list baru berisi nilai dari *list* sebelum acak dengan urutan yang acak. Algoritma ini berlaku untuk acak soal dan acak jawaban. Algoritma ini nantinya yang akan dipakai oleh *method* getSoal() yang mengimplementasi algoritma ambil soal yang akan dijelaskan setelah ini.

1. **Implementasi Algoritma Pengambilan Soal**

Berdasarkan analisis dan perancangan sistem tentang algoritma pengambilan soal yang berada pada bab tiga, dengan gambaran *flowchart* yang berada pada gambar 3.1 dan gambar 3.2 tentang bagaimana soal diambil per bidang dan kemudian dikumpulkan dalam satu *list* allSoals dapat di implementasikan dalam *sourcecode* sebagai berikut :

public void getSoalGambars() {

soalGambars = new ArrayList<Soal>();

int jumlahSoalGambar = settingService.getSetting().getJumlahSoalGambarTampil();

this.soalGambar = jumlahSoalGambar;

List<Soal> soalGambarList = soalService.findSoalsByKategori(8L, jumlahSoalGambar);

soalGambarList = setRandomSoal(soalGambarList);

for (Soal soal : soalGambarList) {

soal.setView(soal.getView() + 1);

soalService.save(soal);

soal.setJawabans(setRandomJawaban(soal.getJawabans()));

soalGambars.add(soal);

}

}

public void getAllSoals() {

getSoalVerbals();

getSoalNumerik();

getSoalLogikals();

getSoalGambars();

allSoals = new ArrayList<Soal>();

for (Soal soal : soalVerbals) {

allSoals.add(soal);

}

for (Soal soal : soalNumeriks) {

allSoals.add(soal);

}

for (Soal soal : soalLogikals) {

allSoals.add(soal);

}

for (Soal soal : soalGambars) {

allSoals.add(soal);

}

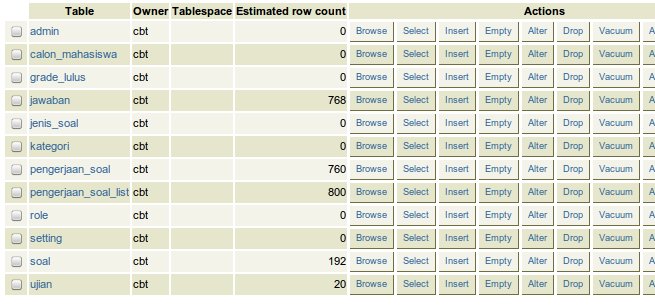
}

Keterangan :

Pada *sourcecode* soal diambil per bidang dengan ketentuan seperti pada *setting* soal, seperti banyaknya soal per bidang yang setelah diambil, kemudian diacak urutan per bidang dengan algoritma acak soal. Setelah itu dilanjutkan acak jawaban persoal. Kemudian soal yang sudah diambil perbidang dimasukan kedalam *list* allSoals untuk kemudian di simpan pada *database* dan didistribusikan kepada peserta ujian dalam tampilan soal.

1. **Implementasi *Database***

*Database* yang dipakai dalam aplikasi ini adalah *PostgreSQL*. Adapun hasil implementasi *database* dapat dilihat pada gambar 4.1.



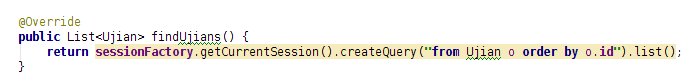
Gambar 4.1 Implementasi Tabel

Koneksi antara aplikasi dan *database* berada dalam satu *file* yang berada pada aplikasi dapat di lihat pada gambar 4.2.



Gambar 4.2 Koneksi *Database*

*Query* yang dipakai dalam aplikasi ini adalah HQL*(Hibernate Query Language)* yang contohnya dapat dilihat pada gambar 4.3.

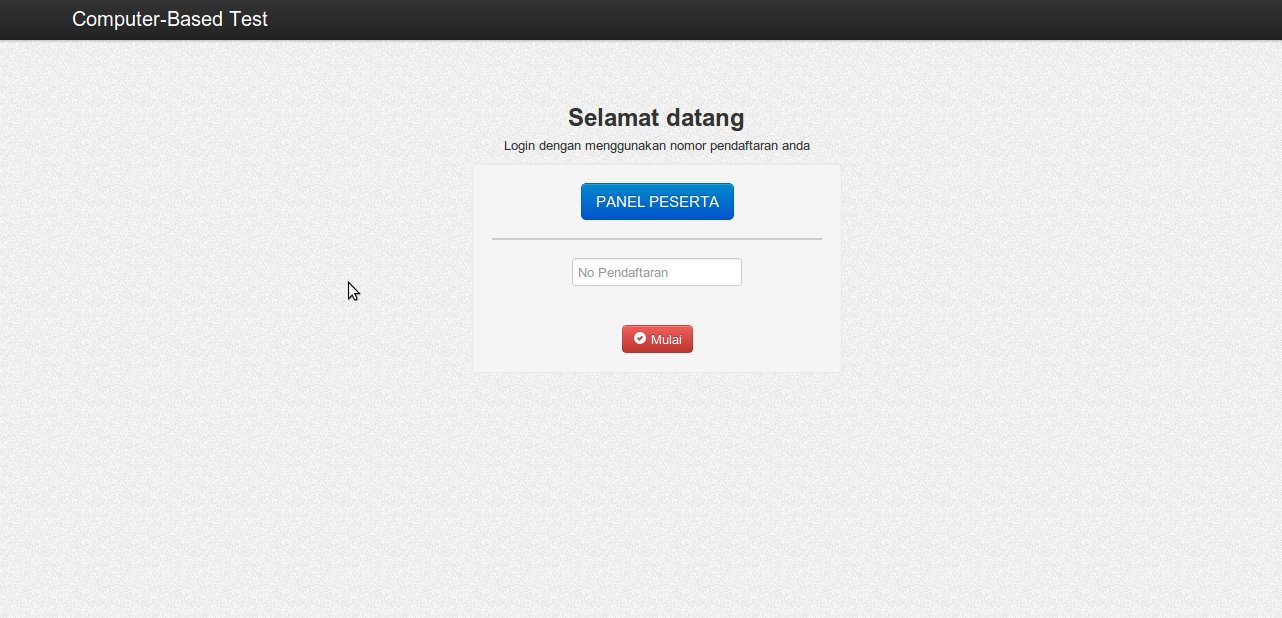


Gambar 4.3 Contoh HQL (*Hibernate Query Language*)

1. **Implementasi Rancangan Tampilan**

Berdasarkan pada rancangan tampilan baik *client* maupun *server* yang ada pada bab tiga, maka disini akan ditampilkan implementasi rancangan yang sudah diterapkan dalam aplikasi *web*.

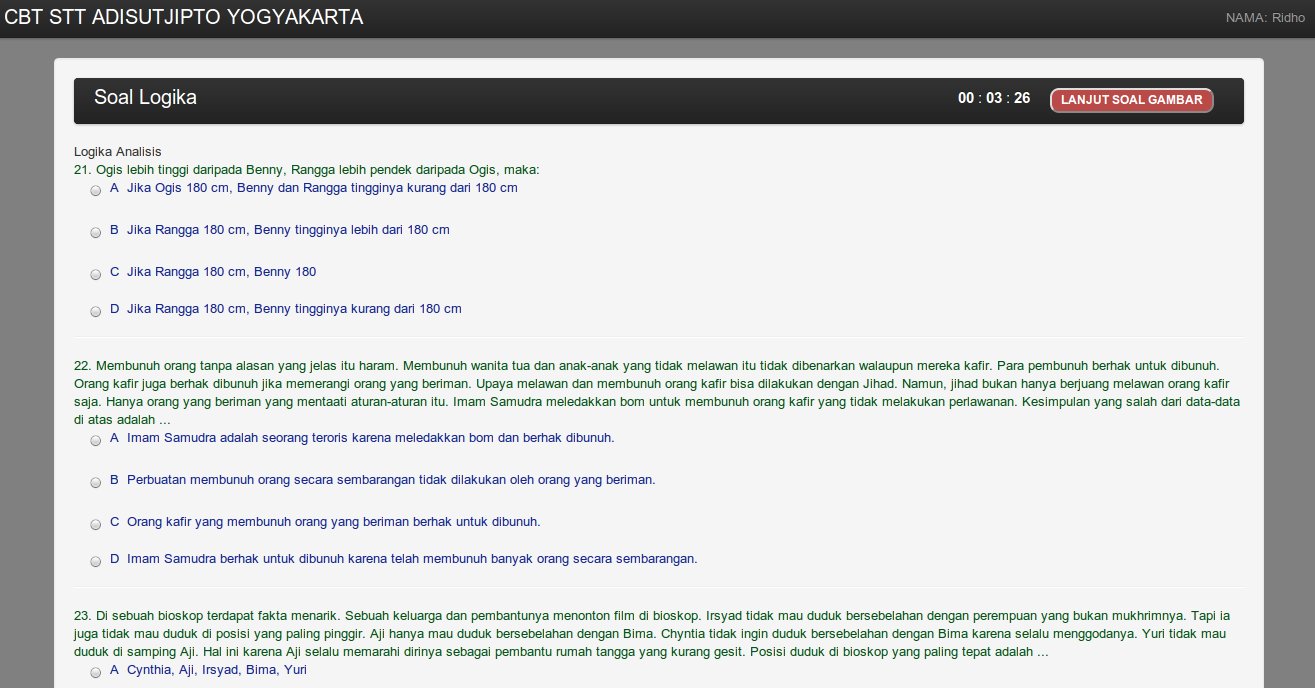
1. **Implementasi Rancangan Tampilan *Client***
2. *Login* Perserta



Gambar 4.4 *Login* Peserta

Gambar 4.4 adalah implementasi dari rancangan *login* peserta. Halaman ini berisi panduan dalam pengerjaan soal dan juga terdapat *field* nomor pendaftaran untuk *login* ke CBT.

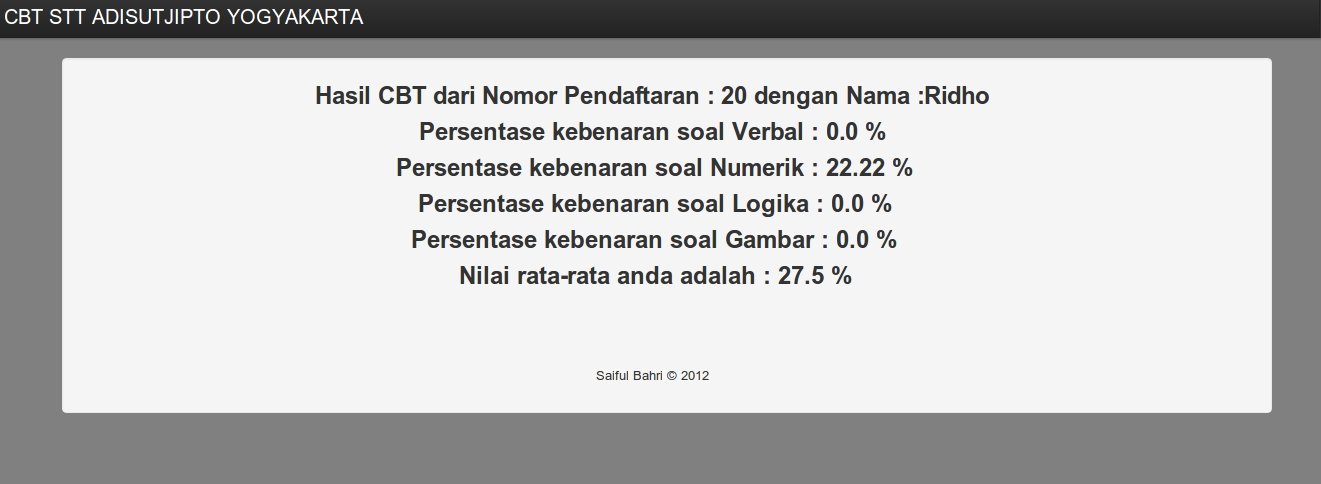
1. Halaman Pengerjaan Soal



Gambar 4.5 Halaman Pengerjaan Soal

Gambar 4.5 adalah lembar pengerjaan soal yang tampil setelah peserta berhasil *login* menggunakan nomor pendaftaran. Diatas terdapat waktu penanda yang berjalan mundur dan terdapat tombol lanjut ke bidang soal lainnya untuk berpindah soal bidang. Bila waktu sudah selesai per bidang maka akan ada pemberitahuan dan melanjutkan kesoal bidang selanjutnya. Di dalam aplikasi ini terdapat empat bidang soal seperti analisis pada bab 3. Untuk mengakhiri ujian bila peserta merasa sudah selesai mengerjakan dan masih ada waktu untuk mengerjakan dapat menekan tombol yang sudah disediakan.

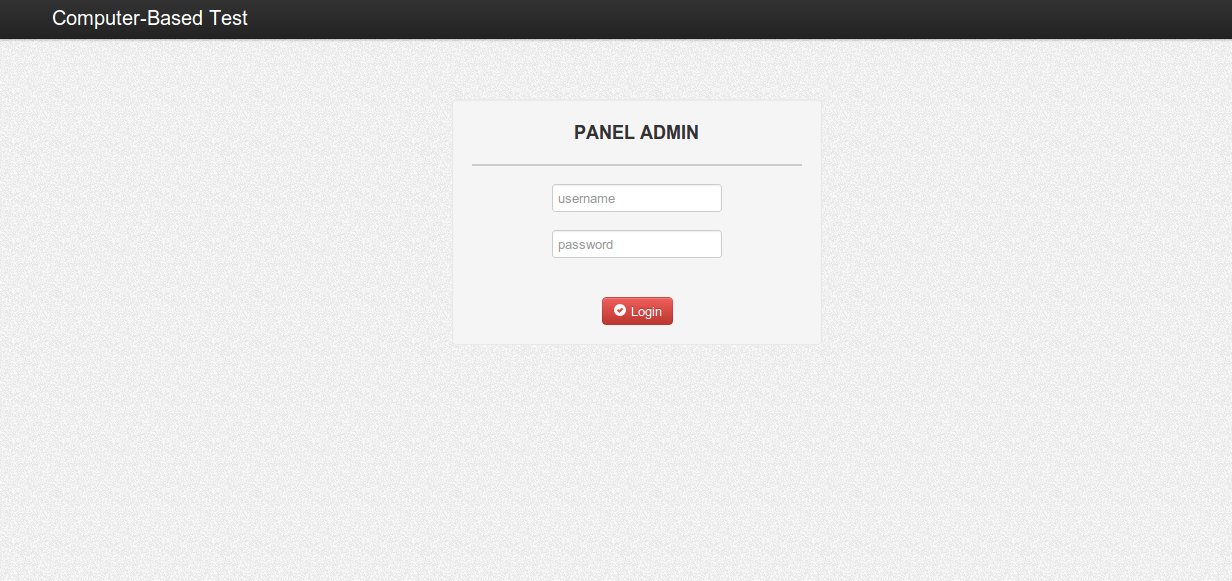
1. Halaman Hasil Pengerjaan



Gambar 4.6 Halaman Hasil Pengerjaan

Gambar 4.6 adalah implementasi dari rancangan tampilan hasil pengerjaan soal. Tampilan ini muncul setelah peserta menyelesaikan pengerjaan soal dan berisi tentang nomor pendaftaran, nama peserta, persentase kebenaran tiap soal dan rata-rata nilai atau hasil akhir.

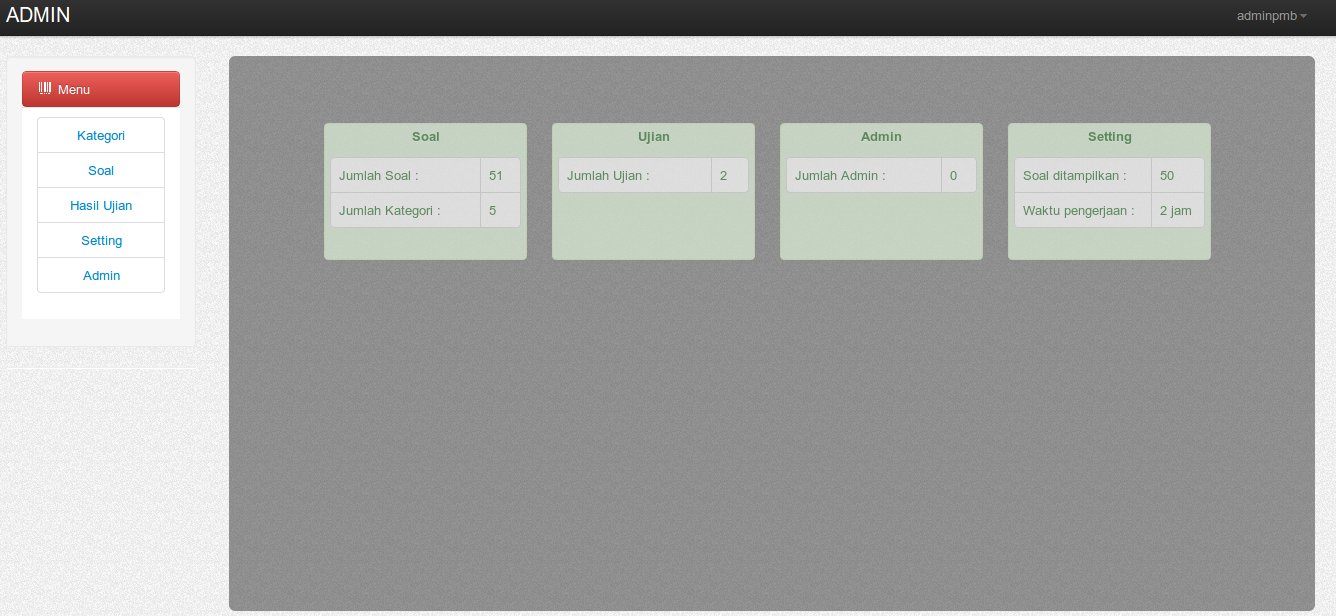
1. **Implementasi Rancangan Tampilan *Server***
2. *Login* *Admin*



Gambar 4.7 *Login* *Admin*

Gambar 4.7 menunjukan untuk *login* ke halaman *dashboard* *admin* harus memasukan *username* dan *password*. Halaman ini berisi informasi *login* ke *dashboard* *admin* dan terdapat *field* *username* *password* yang berfungsi untuk *login*.

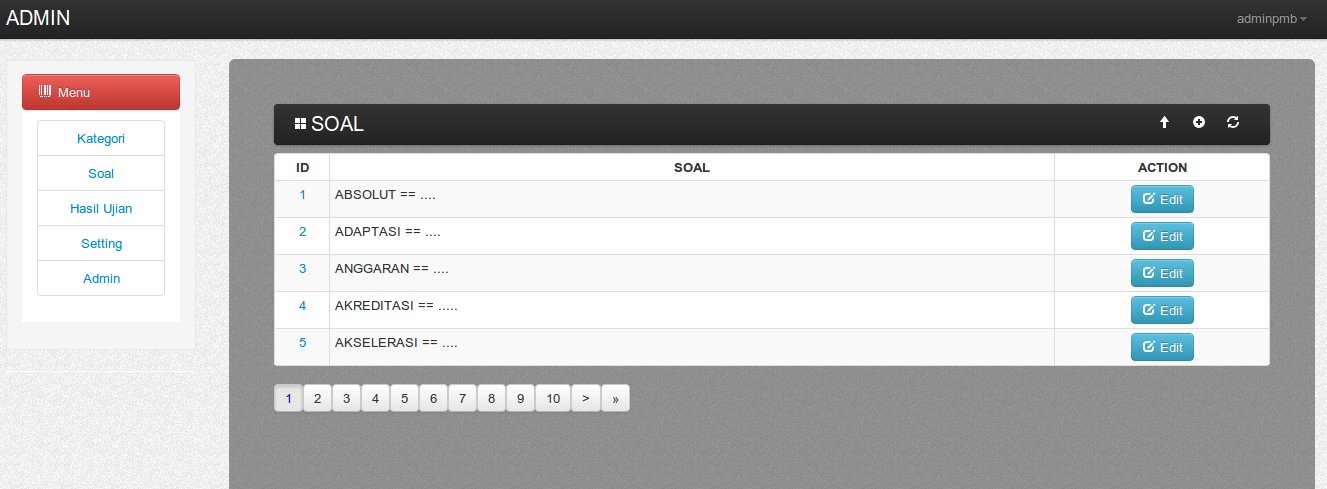
1. Halaman *Dashboard*



Gambar 4.8 Halaman *Dashboard*

Gambar 4.8 adalah halaman dimana seorang *admin* sukses melakukan *login*. Di halaman ini ditampilkan beberapa informasi yang bekanan dengan aplikasi seperti jumlah soal, *setting*, *operator*. Disebelah kiri terdapat pilihan menu dan sebelah kanan berfungsi sebagai konten tiap menu yang dipilih.

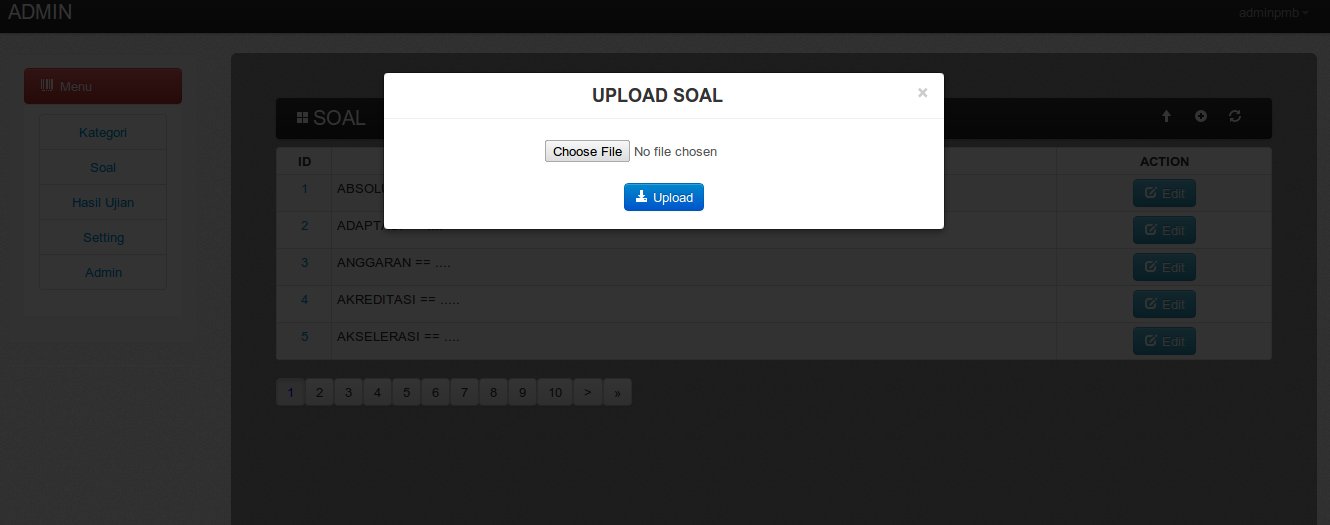
1. Halaman Soal



Gambar 4.9 Halaman Soal

Pada gambar 4.9 terdapat konten soal yang berisi tabel soal yang mempunyai *action* *edit* dan dibagian bawah terdapat *pagination* untuk menuju kesoal selanjutnya. Pada *navbar* soal terdapat menu *upload* soal, *input* soal, dan juga *refresh*.

1. *Upload* Soal



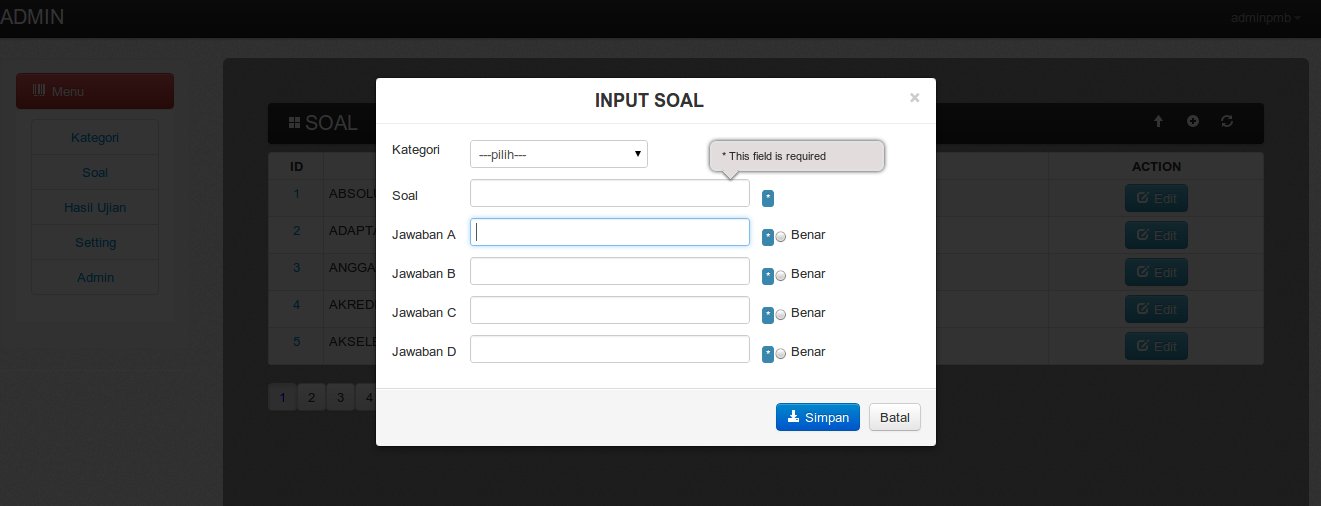
Gambar 4.10 *Upload* Soal

Pada gambar 4.10 terdapat menu *browse* untuk memilih *fiile*. *File* yang dapat dieksekusi adalah *file* *.xls* (*excel*) dengan format :

Tabel 4.1 Format *Upload*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pertanyaan** | **Idkategori** | **JawabanA** | **JawabanB** | **JawabanC** | **JawabanD** | **Kunci** |
| P1 | 1 | A | B | C | D | A |

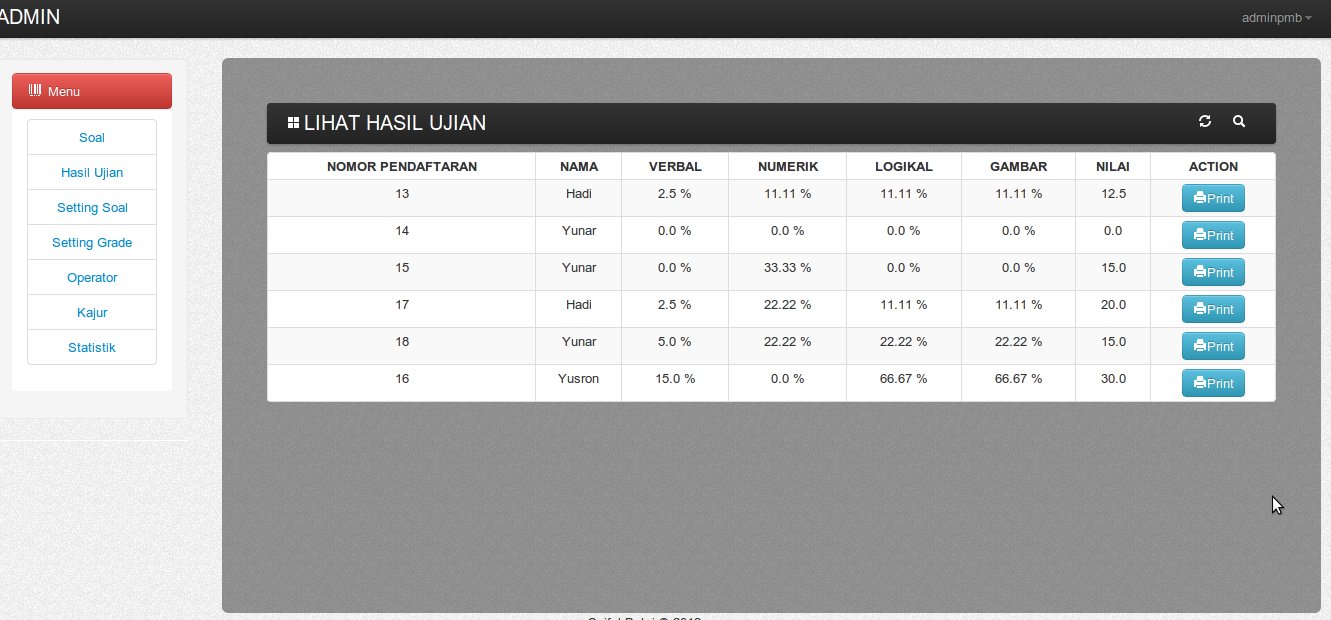
1. *Input* Soal



Gambar 4.11 *Input* Soal

Pada gambar 4.11 terdapat *combobox* kategori, *text field* soal jawaban dan juga *radiobutton* kunci. Dalam *form* ini juga sudah dilengkapi dengan *form validation* sehingga akan ada peringatan bila terdapat inputan yang dilewati.

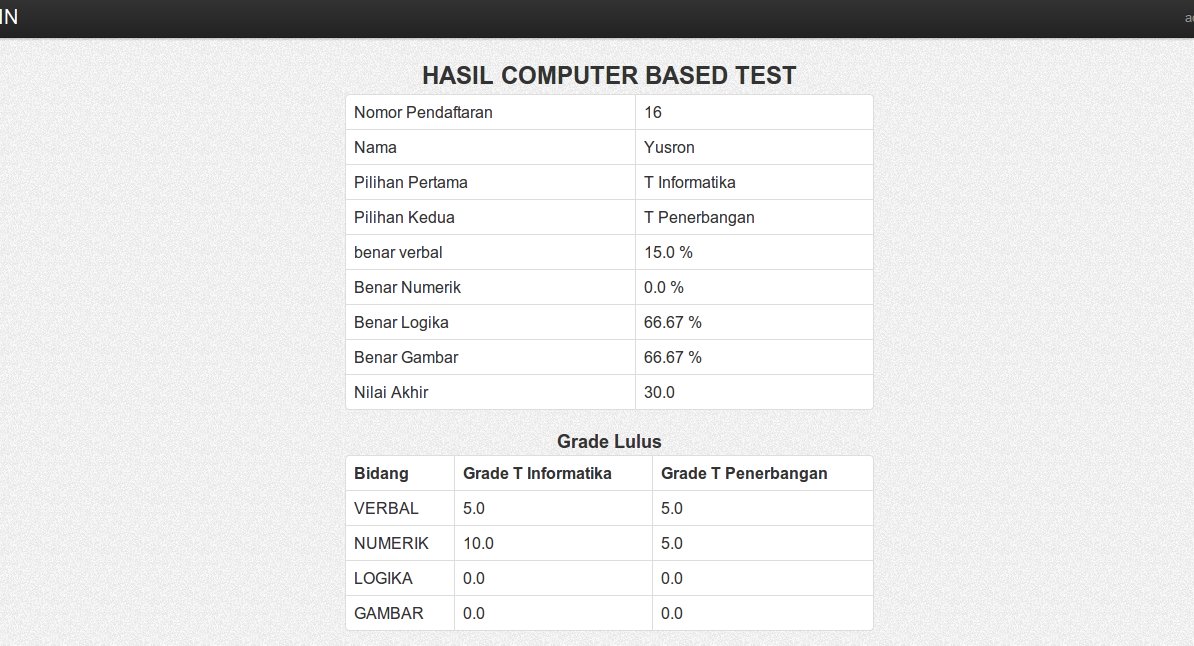
1. Halaman Hasil Ujian



Gambar 4.12 Halaman Hasil Ujian

Pada gambar 4.12 berisi informasi hasil ujian peserta dan terdapat fungsi *print* yang berguna untuk mencetak hasil ujian peserta PMB yang nantinya dapat diteruskan kepada yang membutuhkan informasi. Konten berisi nomor pendaftaran, nama, nilai bidang dan nilai akhir.

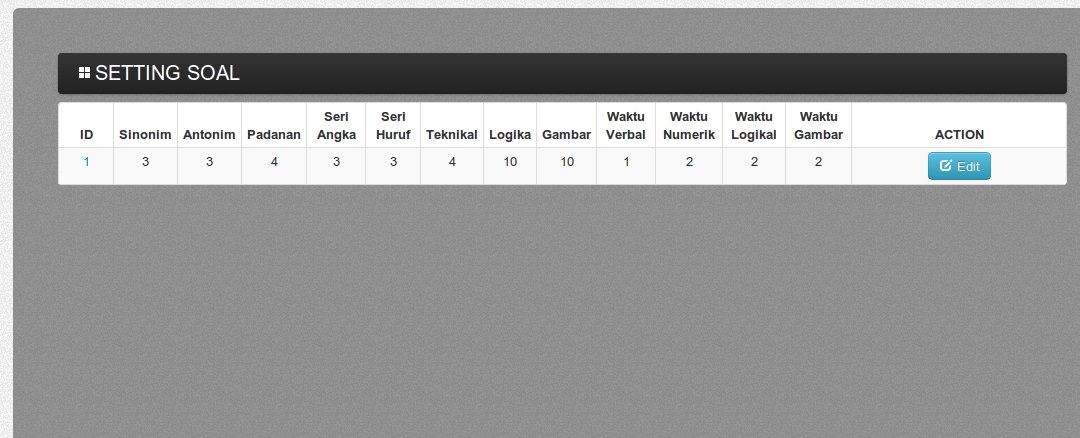
1. Halaman *Print Out* Hasil



Gambar 4.13 *Print Out* Hasil Ujian

Gambar 4.13 adalah hasil tampilan yang nantinya dapat melakukan *print* sebagai laporan hasil CBT peserta PMB. Tampilan ini muncul setelah tombol *print* diklik pada konten hasil ujian.

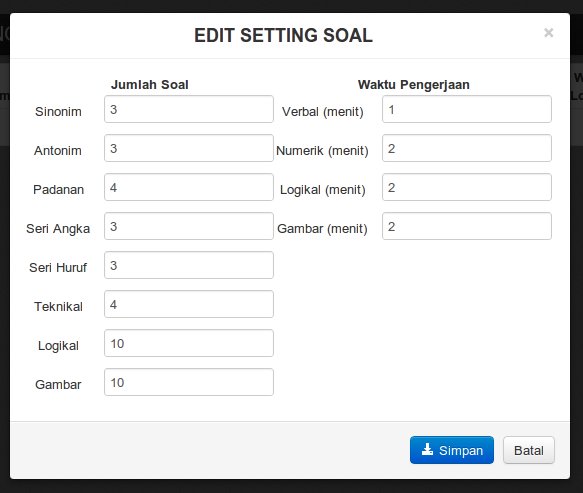
1. Halaman *Setting*



Gambar 4.14 Halaman *Setting* Soal

Pada gambar 4.14 diperlihatkan tabel yang hanya terdapat satu *row* dan tidak terdapat menu *add* karena *setting* hanya terdapat satu didalam aplikasi. *Setting* disini mengatur berapa banyak soal per kategori yang ditampilkan dan berapa lama waktu per bidang yang diberikan dalam pengerjaan soal.

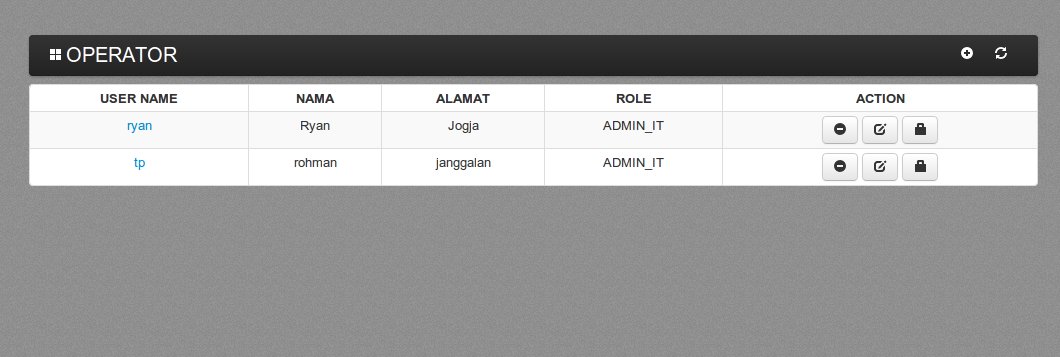
1. Halaman *Edit Setting*



Gambar 4.15 Halaman *Edit Setting* Soal

Gambar 4.15 terdapat *field* jumlah soal per kategori yang ingin ditampilkan dan lama waktu pengerjaan per bidang soal. Dari *form* ini *setting* dapat diubah oleh *operator*.

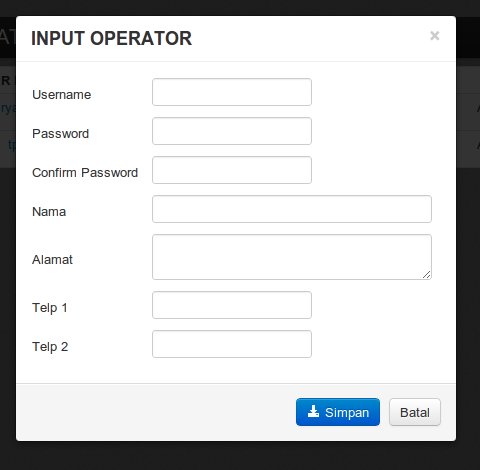
1. Halaman *Operator*



Gambar 4.16 Halaman *Operator*

Pada gambar 4.16 terdapat tabel yang berisi informasi *admin* yang dapat masuk ke dalam aplikasi. Terdapat menu *edit* *admin* dan *update* *password*. Pada *navbar* juga terdapat menu *add* *admin* dan *refresh*.

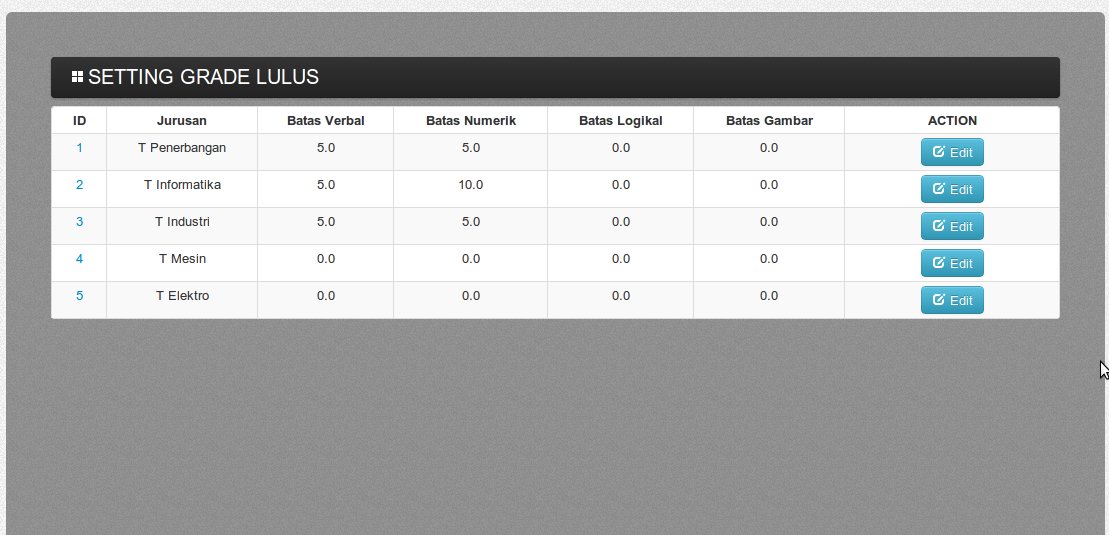
1. *Input* *Operator*



Gambar 4.17 *Input* *Operator*

Pada gambar 4.17 terdapat *field* *username* *password* *confirm* *password* nama dan alamat yang harus di isi, sedangkan telp 1 dan telp 2 boleh untuk dikosongkan. Dalam *form* ini juga sudah dilengkapi dengan *form validation.*

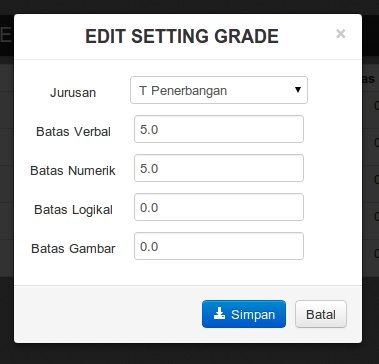
1. *Setting* *Grade* Lulus



Gambar 4.18 *Setting* *Grade*

Pada gambar 4.18 ditampilkan tabel *setting* *grade* lulus perjurusan yang berikan per bidang soal. *Grade* ini yang nantinya dipakai untuk batasan lulus atau tidaknya perserta ujian PMB. *Setting* ini dapat diubah oleh *operator*.

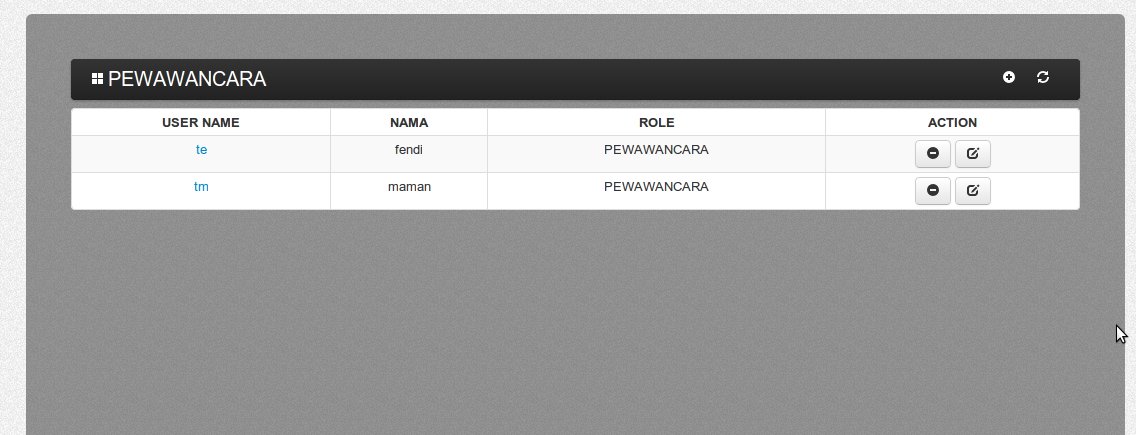
1. *Edit Setting Grade* Lulus



Gambar 4.19 *Edit Setting Grade*

Gambar 4.19 menampilkan *form edit setting* *grade* yang berfungsi untuk mengganti nilai batas lulus pada bidang soal pada setiap jurusan.

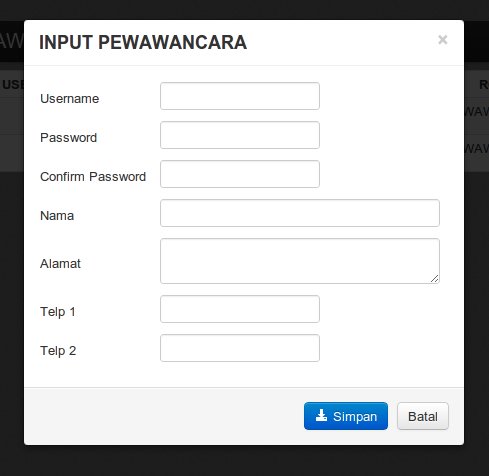
1. Pewawancara



Gambar 4.20 Halaman Pewawancara

Pada gambar 4.20 terdapat tabel yang berisi *list* pewawancara. Menu ini berfungsi untuk menambah pewawancara yang nantinya dapat masuk ke sistem CBT hanya untuk melihat statistik data nilai yang ada pada CBT PMB.

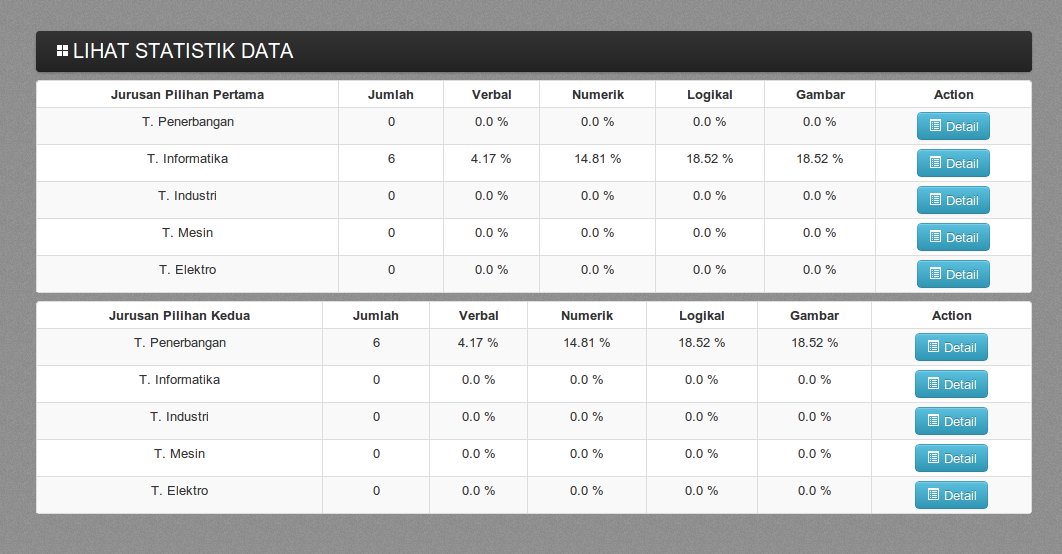
1. *Input* Pewawancara



Gambar 4.21 *Input* Pewawancara

Pada gambar 4.21 terdapat *form* yang dapat digunakan untuk menambah pewawancara yang berguna untuk *login* ke CBT guna melihat statistik data nilai hasil CBT.

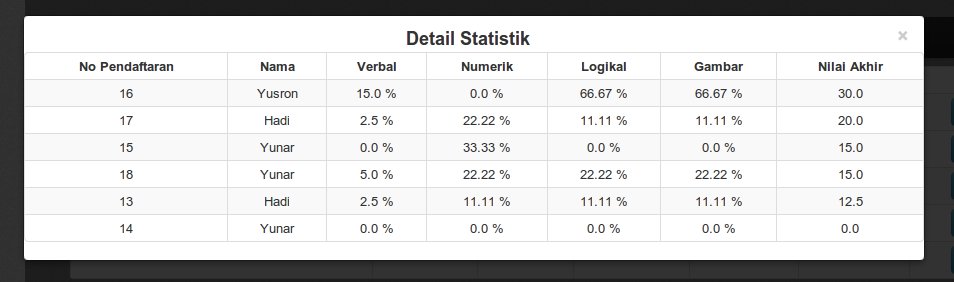
1. Statistik



Gambar 4.22 Statistik

Pada gambar 4.22 terdapat dua tabel yang berisi statistik nilai untuk pilihan jurusan pertama dan pilihan jurusan kedua. Dari statistik ini dapat dilihat persentasi kebenaran rata-rata per bidang soal yang dapat dijadikan acuan kemampuan akademik calon mahasiswa. Terdapat menu detail untuk melihat hasil statistik ujian perjurusan.

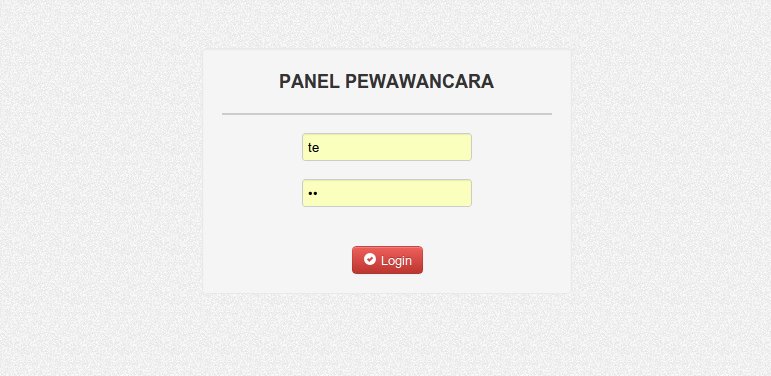
1. *Detail* Statistik



Gambar 4.23 *Detail* Statistik

Pada gambar 4.23 terdapat tabel yang berisi detail dari statistik yang ada pada konten statistik dan hasilnya dari *sort by* nilai akhir dari yang tertinggi sampai terendah.

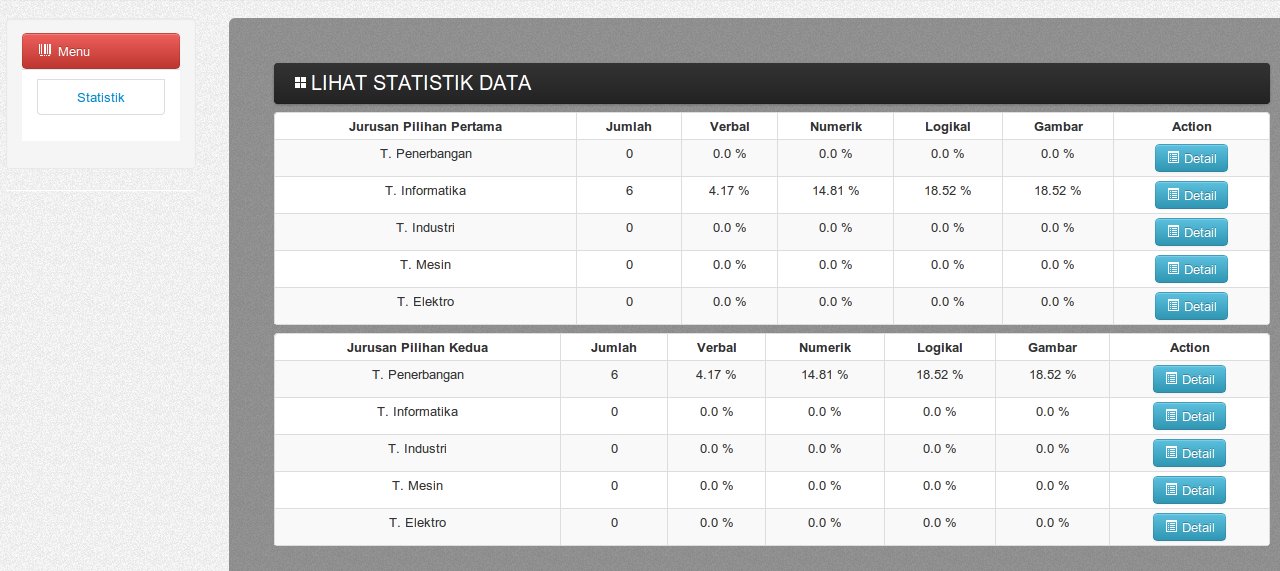
1. **Rancangan Tampilan Pewawancara**
2. Tampilan *Login* Pewawancara



Gambar 4.24 Panel *Login* Pewawancara

Pada gambar 4.24 terdapat *form* *login* untuk pewawancara yang akan digunakan untuk melihat statistik pada *dashboard.*

1. Tampilan *Dashboard* Statistik



Gambar 4.25 *Dashboard* Pewawancara

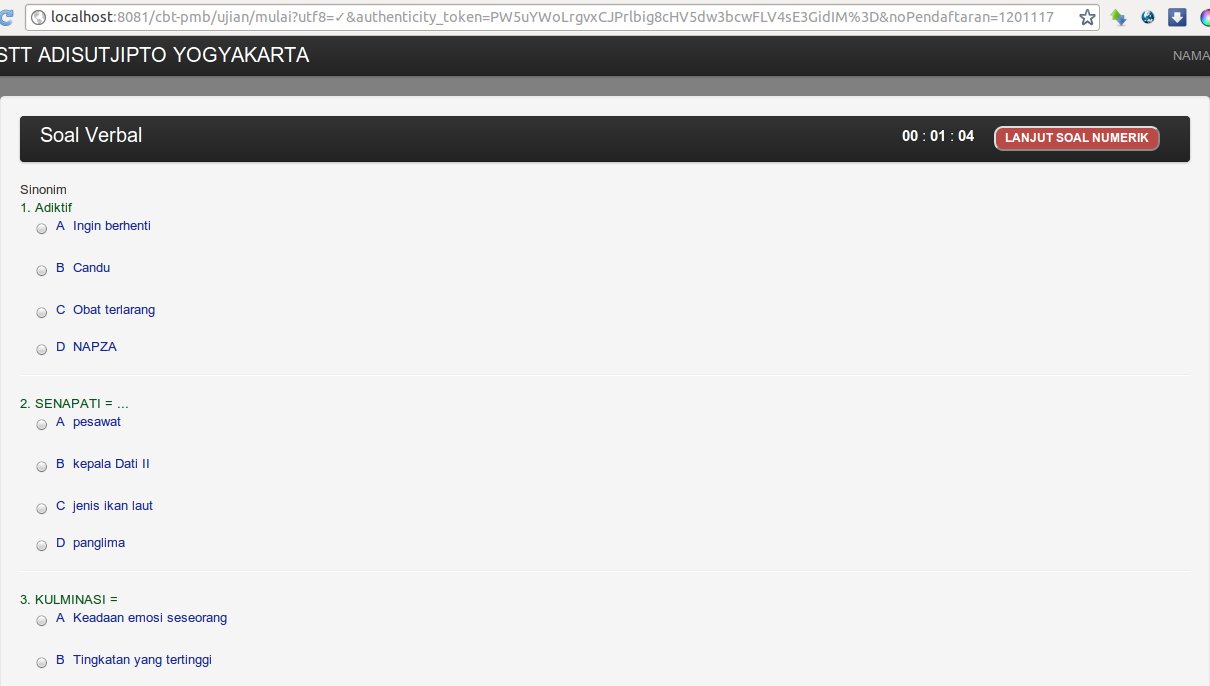
Pada gambar 4.25 terlihat *dashboard* hanya memiliki menu statistik, ini karena autorisasi pewawancara hanya bisa melihat statistik saja.

1. **Ujicoba Fungsi**

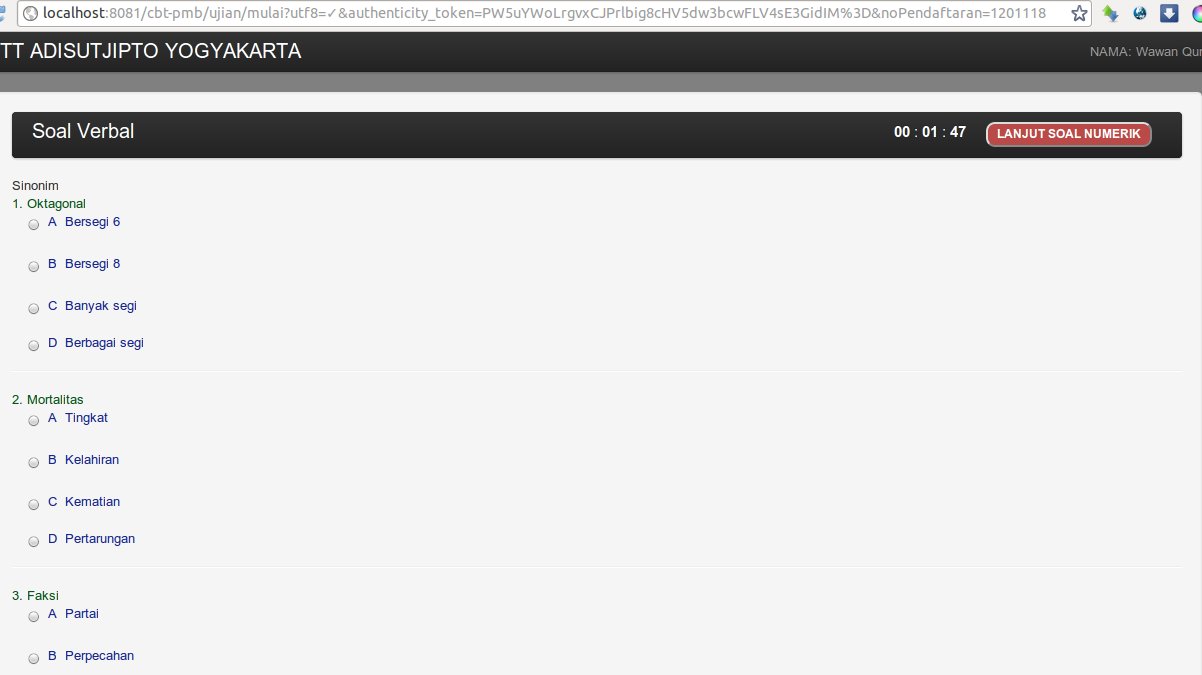
Uji Fungsi merupakan tahapan selanjutnya dari penelitian ini, uji fungsi meliputi dua jenis pengujian yaitu uji algoritma dan uji jaringan *internet*.

1. **Uji Fungsi Algoritma**

Dari implementasi sistem yang terdapat sub bab sebelumnya maka ujicoba algoritma LCG pada CBT PMB STTA dapat dilihat dengan cara membandingkan dua soal yang muncul bersama dalam komputer yang berbeda apakah mempunyai urutan soal yang sama ataukah tidak. Ujicoba algoritma dapat dilihat pada gambar 4.26 dan gambar 4.27.



Gambar 4.26 Uji Algoritma 1



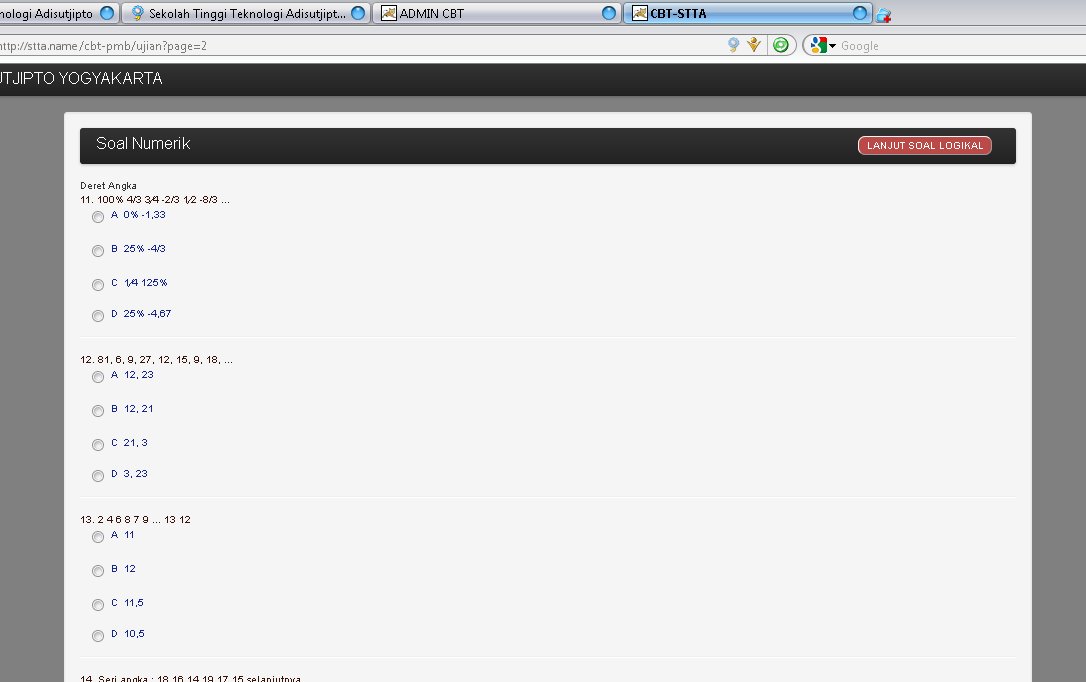
Gambar 4.27 Uji Algoritma 2

Dari gambar dapat dilihat bahwa gambar 4.26 mempunyai urutan soal yang berbeda dengan gambar 4.27. Dengan demikian ujicoba penerapan algoritma LCG pada CBT berhasil.

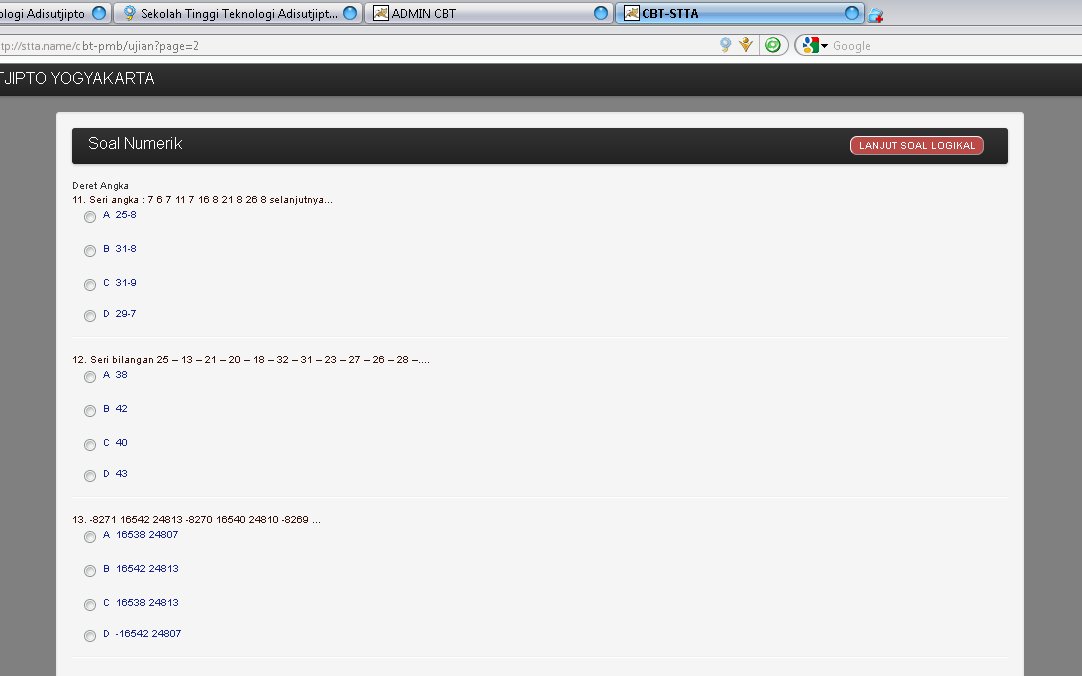
1. **Uji Jaringan *Internet***

Tahapan dalam melakukan uji fungsi selanjutnya adalah uji CBT dengan *internet*. Uji CBT dengan internet merupakan uji coba aplikasi didalam jaringan *internet* dengan memanfaatkan layanan *hosting* sebagai *file* *server* dan *database* *server*. Komputer *client* agar dapat terkoneksi ke *server* dengan menggunakan media transmisi *internet* berupa *modem*. Pengujian ini dilakukan untuk menguji berkerja atau tidak aplikasi CBT dalam jaringan *internet* dan untuk mencari apakah algoritma acak soal berjalan dengan baik.

Uji *internet* akan dibuktikan dengan dua gambar yang melakukan ujian dengan menggunakan jalur *internet*. Gambar dapat dilihat pada gambar 4.28 dan gambar 4.29.



Gambar 4.28 Uji Jaringan 1



Gambar 4.29 Uji Jaringan 2

Dari gambar dapat dilihat bahwa gambar 4.28 memiliki urutan soal yang berbeda dengan gambar 4.29 dan telah diuji menggunakan jaringan *internet* dengan alamat <http://stta.name/cbt-pmb>. dengan demikian uji *internet* berhasil.

1. **Uji Pengguna**

Uji pengguna dilakukan menggunakan *Beta Test* dengan melakukan ujicoba kepada 30 pengguna guna menghasilkan nilai kelayakan aplikasi CBT sebelum diterapkan di Sekolah Tinggi Teknologi Adisutjipto.

Uji pengguna dalam penelitian ini menggunakan kuisioner dengan pertanyaan dan angka sebagai nilainya, angka 5 adalah angka penilaian yang paling tinggi. Kuisioner dapat dilihat pada tabel 4.2.

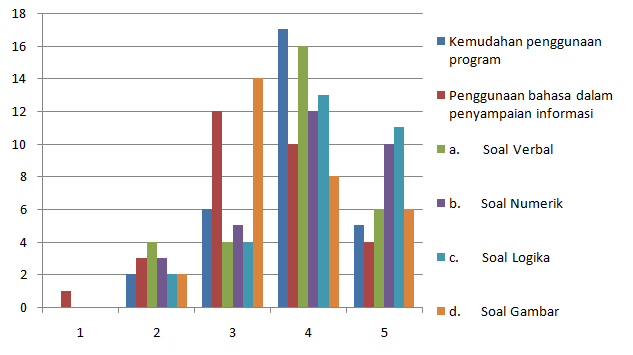
Tabel 4.2 Kuisioner

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kuisioner** | **Nilai** | | | | |
| 1 | Kemudahan penggunaan program | 1 | 2 | 3 | 4 | 5 |
| 2 | Penggunaan bahasa dalam penyampaian informasi | 1 | 2 | 3 | 4 | 5 |
| 3 | Desain *interface*   1. Soal Verbal 2. Soal Numerik 3. Soal Logika 4. Soal Gambar | 1  1  1  1 | 2  2  2  2 | 3  3  3  3 | 4  4  4  4 | 5  5  5  5 |

Hasil kuisioner yang telah dilaksanakan dapat dilihat pada tabel 4.3.

Tabel 4.3 Hasil Kuisioner

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kuisioner** | **Jumlah Pengisi** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| 1 | Kemudahan penggunaan program | 0 | 2 | 6 | 17 | 5 |
| 2 | Penggunaan bahasa dalam penyampaian informasi | 1 | 3 | 12 | 10 | 4 |
| 3 | Desain *interface*   1. Soal Verbal 2. Soal Numerik 3. Soal Logika 4. Soal Gambar | 0  0  0  0 | 4  3  2  2 | 4  5  4  14 | 16  12  13  8 | 6  10  11  6 |
| Jumlah | | 1 | 16 | 45 | 76 | 42 |
| Persentase | | 0.5 | 8.88 | 25 | 42.3 | 23.3 |



Gambar 4.30 Grafik Hasil Kuisioner CBT

Dengan hasil kuisioner yang terlah dilakukan persentase nilai 1 sebesar 0.5, nilai 2 sebesar 8.8, nilai 3 sebesar 25, nilai 4 sebesar 42.2 dan nilai 5 sebesar 23.33 sedangkan nilai 5 adalah nilai terbesar, maka aplikasi ini sudah dapat diterapkan di Sekolah Tinggi Teknologi Adisutjipto untuk membantu dalam pelaksanaan ujian Penerimaan Mahasiswa Baru.

Surat keterangan bahwa aplikasi dapat diimplementasikan pada PMB STTA telah dikeluarkan oleh ketua PMB STTA dan terdapat pada lampiran.

**BAB V**

**KESIMPULAN DAN SARAN**

* 1. **Kesimpulan**

Kesimpulan yang didapat dari penelitian ini adalah :

* 1. Algoritma *Linear Congruential Generator* (LCG) Donald E. Knuth dapat mendukung pengacakan soal pada CBT.
  2. Aplikasi dapat digunakan pada ujian Penerimaan Mahasiswa Baru (PMB) dan CBT Tes Potensi Akademik menjadi lebih flaksibel.
  3. Aplikasi *client* hanya menangani dari calon mahasiswa memasukan nomor pendaftaran sampai selesai mengerjakan soal dan melihat nilai yang didapatkan.

**5.2 Saran**

Pada proses pembuatan tugas akhir ini, aplikasi yang dibangun masih dapat dikembangkan, antara lain :

1. Pada aplikasi ini hanya membahas soal Tes Potensi Akademik (TPA), dapat dikembangkan untuk soal matakuliah.
2. Aplikasi ini hanya menggunakan satu *database*, jadi kalau satu *database* *down* maka aplikasi tidak bisa digunakan. Dapat dikembangkan untuk multi *database*.
3. *Random* hanya terbatas pada soal dan jawaban. Dapat dikembangkan untuk *random* dalam modul yang lain.

**DAFTAR PUSTAKA**

E. Knuth.. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms, Third Edition*, USA:Addison-Wesley,1998.

Tim Editor PM.. *Buku Jagoan TPA (TES POTENSI AKADEMIK).* Depok:Pustaka Makmur, 2011.

Tepper Paul, Brian D. Murphy. *Spring Persistence with Hibernate* . USA:Apress, 2010.

Stefonov Stoyan. *Object-Oriented JavaScript.* USA:Packt, 2008.

Bima Ifnu. *Java Desktop Aplikasi POS Berarsitektu Three Tire Menggunakan Swing, Hibernate, dan Spring*. Bandung:NulisBuku, 2011.

Wardiana Wawan, Tobing Wisca Veronika. 2011. *Aplikasi SIstem Pakar Tes Kepribadian Berbasis Web*. Jurnal INKOM, Vol. 5, No.2, Hal. 99-104. Bandung.

Herlan, Fryantoni Herlan. 2011. *Pengacak (Scrambler-Descrambler) Untuk Penyelamatan Informasi Pada Sistem Komunikasi Morse*. Jurnal INKOM, Vol. 5, No.2, Hal. 113-118. Bandung.