BAB III
PERANCANGAN SISTEM DAN DESAIN

Pada bab ini akan dijelaskan perancangan Sistem yang akan dikembangkan pada Aplikasi Tes Akademik.

3.1 Deskripsi Sistem

Aplikasi ini secara khusus bertujuan untuk mengkomputerisasi pelaksanaan Aplikasi Tes Akademik. Aplikasi ini dibagi menjadi dua bagian antara peserta dan *admin*. Di mana peserta hanya bisa memasukan nomer pendaftaran sampai selesai mengerjakan soal yang sudah disediakan dan memperoleh hasil yang didapatkan. Sedangkan admin berfungsi untuk memasukan data-data soal yang akan ditampilkan ke peserta ujian.

3.2 Analisa Kebutuhan 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.

3.2.1 Analisa Sistem

Pengguna yang ada dalam sistem ujian potensi akademik ini terbagi menjadi 2 (tipe) tipe pengguna yaitu :

a. Peserta
   Peserta adalah *client* yang akan memasukan nomer pendaftaran hingga selesai soal dikerjakan dan menampilkan nilai yang di dapatkan.
b. *Admin*
   Admin adalah *Server* menerima permintaan dari *client* dan bertugas mengirimkan soal untuk ditampilkan ke dalam aplikasi peserta.

Terdapat beberapa proses-proses utama yang terdapat dalam aplikasi ini, antara lain :

a. *Menu Admin*
   Fungsi ini digunakan untuk mengelola data soal,jurusan dan peserta ujian yang akan mengikuti ujian tes akademik.
b. *Temporary file*
   Fungsi ini digunakan untuk menyimpan *temporary* soal jika terdapat komputer *crash* atau mati.
c. *Scoring*

Fungsi ini digunakan untuk menghitung nilai ujian yang telah diikuti oleh peserta. Hasilnya secara lansung akan ditampilkan setelah peserta selesai mengerjakan soal.

d. *Timing*
Fungsi ini digunakan untuk membatasi waktu pengerjakan soal.

3.2.2 Analisa Fungsional

Analisa fungsional merupakan paparan menganai fitur-fitur yang akan dimasukkan ke dalam aplikasi, fitur tersebut antara lain :

1. Algoritma *Linear Congruential Generators* (LCG) untuk pengacakan soal aplikasi ujian tes akademik.
2. *Create Import* format XML ke dalam *database* untuk menyimpan *temporary* soal peserta ujian jika terjadi komputer *crash* atau mati.
3. Aplikasi terdapat waktu yang berjalan mundur sebagai pengingat batas waktu pengerjaan.
4. Aplikasi terdapat tombol *next* untuk melanjutkan ke soal berikutnya.

3.2.3 Performansi Sistem

Aplikasi bisa berjalan disemua sistem operasi, juga terdapat beberapa keterbatasan yang ditemui pada aplikasi ini. Sehingga perlu diperhatikan guna untuk menjadi acuan dalam pengembangan aplikasi, diantaranya sebagai berikut :

a. Aplikasi *Client* hanya memasukan nomer pendaftaran sampai selesai pengerjaan soal ujian dan melihat nilai yang didapatkan.
b. Aplikasi ini mempunyai waktu untuk membatasi peserta ujian dalam mengerjakan soal.

3.3 Algoritma pengacakan soal

Di dalam aplikasi tes akademik pengacakan menggunakan perkembangan Algoritma Linear Congruential Generator (LCG). LCG merupakan pembakit bilangan acak yang sederhana. Dalam Pengacakan ini di mana banyaknya soal sebagai modulus (M). Serta untuk memenuhi syarat penuh satu periode maka akan digunakan A sebagai (pengali) dan B sebagai (increment).

Rumus acak yang digunakan dalam Aplikasi tes akademik ini sebagai berikut :

$X_n = ((A * X_{1n}) + B) \bmod M$ Di mana :

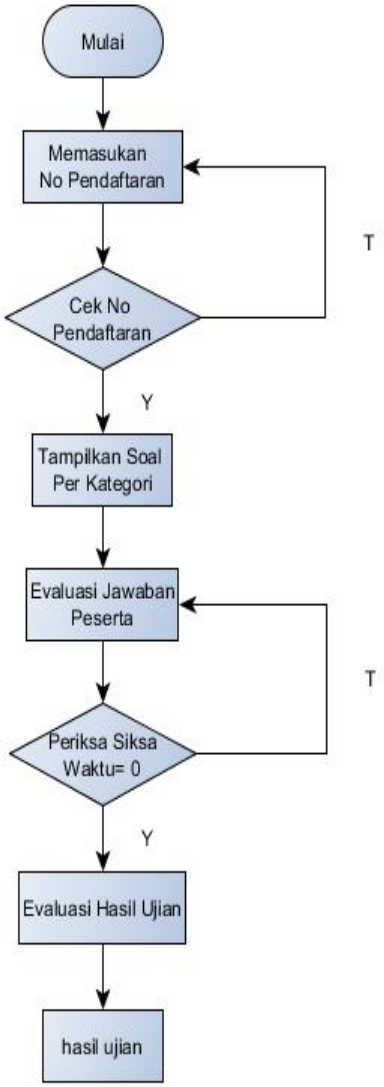
a. *A* adalah A(faktor pengali) yang diambil dari (B/2)+1.
b. Val n+1 adalah nilai umpan yang dihasilkan sebelumnya.
c. B adalah (*increment*) yang nilai diambil dari (M/2)+1.
d. *M adalah modulus* yang diambil dari banyaknya soal yang ada di *database*s.

3.4 *Temporary* Soal

Di dalam aplikasi tes akademik *temporary* soal menggunakan XML. XML adalah salah satu media yang seringkali digunakan untuk membawa data di dalam web services. Ketika sebuah request dikirimkan dari klien ke sebuah service, kita harus membentuk request message dalam bentuk XML. Begitu juga sebaliknya, ketika sebuah service mengirimkan response ke klien, maka service harus mengubah format pesan response dalam bentuk XML. Untuk mengolah pesan berbentuk XML dibutuhkan XML parser. PHP memiliki XML parser yaitu SimpleXML dan DOM.

3.5 Alur sistem ujian

Di dalam alur sistem ujian pertama-tama peserta memasukan nopendaftaran. Setelah itu sistem akan memverifikasi jika benar maka peserta dapat mengerjakan soal yang sudah dibagi berdasarkan kategori. Sistem akan mengevaluasi jawaban peserta, jika waktu sama dengan kosong maka secara otomatis sistem akan mengevaluasi hasil yang selanjutnya akan ditampilkan ke peserta ujian. *Flowchart* Alur sistem bisa dilihat pada gambar 3.1.

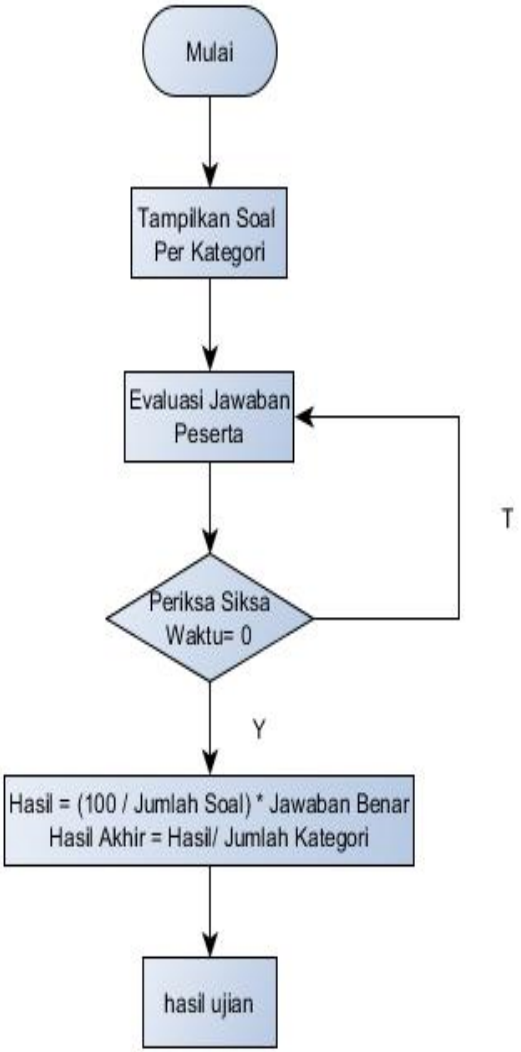


Gambar 3.1 *Flowchart* alur ujian

3.6 Alur penilaian hasil ujian

Hasil ujian keluar ketika peserta ujian sudah selesai mengerjakan soal ,setelah itu sistem akan melakukan pengecekan berapa jawaban yang benar lalu dikalihkan dengan jumlah soal dan dibagi 100. Setelah itu hasilnya di bagi per

kategori. Untuk lebih jelasnya dan hasilnya akan ditampilkan ke peserta ujian. *Flowchart* penilaian hasil ujian dapat dilihat pada gambar 3.2.



Gambar 3.2 *Flowchart* alur penilaian ujian

3.7 Model Proses

Untuk membuat Aplikasi Tes Akademik ini, perlu dilakukan analisa terlebih dahulu terhadap kebutuhan proses. Proses yang dimaksud bertujuan menjelaskan alur perancangan aplikasi.

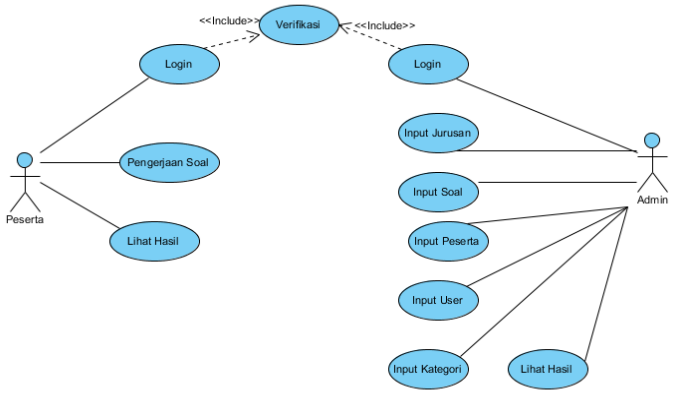
Desain sistem secara umum dilakukan dengan maksud untuk memberikan gambaran umum tentang sistem yang baru atau sistem yang akan diusulkan.

Jadi untuk membuat Aplikasi Ujian Akademik ini perlu dilakukan terlebih dahulu analisa terhadap kebutuhan sistem.

3.7.1 Prosedur dan Proses

Penyusunan proses ini bertujuan untuk memperjelas sebuah proses dan proses tersebut memperoleh deskripsi yang tepat mengenai apa yang dicapai serta untuk memvalidasi sistem agar sistem dapat berjalan sesuai dengan desain yang telah dibuat dan dikembangkan.

Untuk lebih jelasnya dapat dilihat pada gambar 3.3. Dalam diagram *Use case* di bawah ini dijelaskan mengenai proses – proses yang dapat dilakukan dalam Aplikasi Tes Akademik.



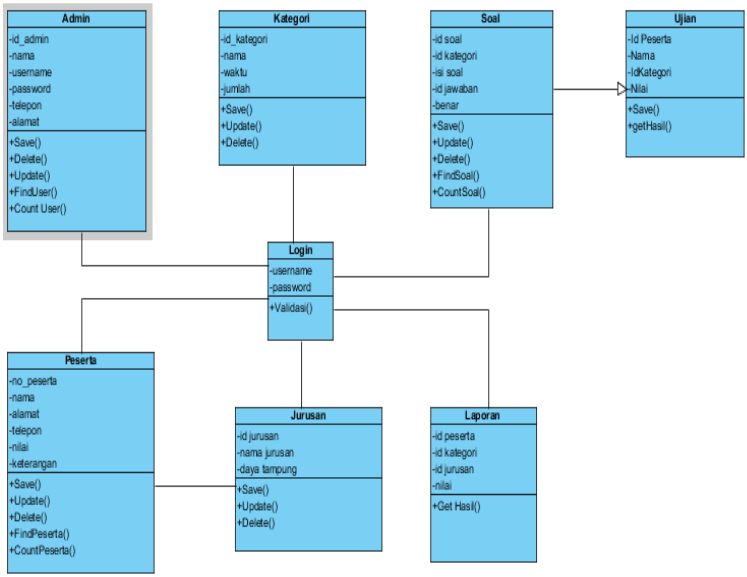
Gambar 3.3 *Use Case* TA

Pada gambar 3.3 diagram *use case* di atas dijelaskan bahwa *actor* terdiri dari peserta dan *admin*, di mana *admin* akan melakukan *login* terlebih dahulu setelah itu baru memiliki autentifikasi untuk mengelola data soal,peserta,dan jurusan. *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.

3.7.2 Diagram *Class*

Diagram *class* adalah diagram yang digunakan untuk menampilkan kelas-kelas dan paket-paket di dalam sistem. *Class diagram* memberikan gambaran sistem secara statis dan relasi antar mereka. Biasanya, dibuat beberapa *class* diagram untuk sistem tunggal. Beberapa diagram akan menampilkan subset dari kelas-kelas dan relasinya. Dapat dibuat beberapa diagram sesuai dengan yang diinginkan untuk mendapatkan gambaran lengkap terhadap sistem yang dibangun. 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 *object* dan *operation*. Diagram *class* dapat dilihat pada gambar 3.4.



Gambar 3.4 Diagram *Class Object*

Class adalah sesuatu yang membungkus (*encapsulate*) informasi dan perilaku. Gambar menjelaskan perancangan diagram *class* untuk aplikasi TA di mana *Object* berisi *class* yang merepresentasikan tabel yang berada pada

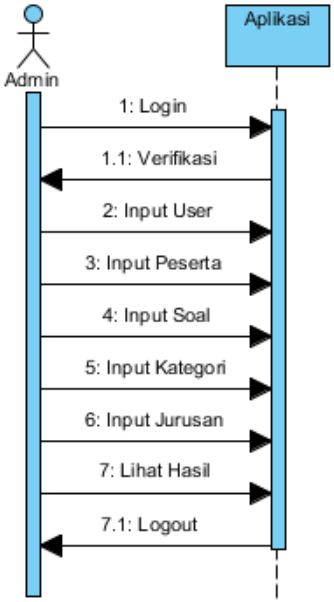
database serta *class* pendukung aplikasi. *Operation* adalah berisi *class* yang berfungsi untuk melakukan fungsi *insert update delete* serta *get* dan mengimplemen dan memprosesnya untuk di simpan pada *database*.

3.7.3 Diagram *Sequence*

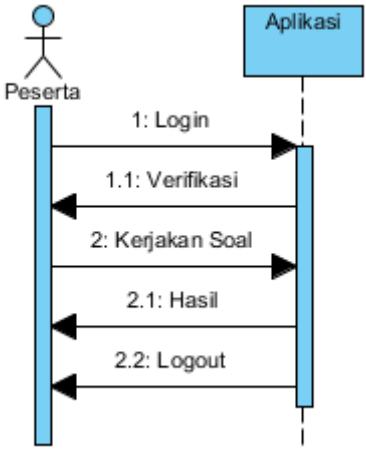
Sequence diagram adalah jenis diagram interaksi yang menunjukan bagaimana proses beroperasi satu dengan yang lainnya. *Sequence diagram* atau diagram urutan adalah konstruksi dari bagan urutan yang berisi pesan. Sebuah diagram urutan menunujukan interaksi *object* berdasarkan dengan urutan waktu.

Hal ini menggambarkan *object* dan kelas yang terlibat dalam skenario dan urutan pesan yang berinteraksi antara *object* yang dibutuhkan untuk melaksanakan fungsi skenario atau proses. *Sequence diagram* biasanya (tetapi tidak selalu), berkaitan dengan realisasi *use case* dalam pandangan logis pada sebuah pengembangan sistem.

Tipe diagram ini sebaiknya digunakan di awal tahap desain dan analisis karena kesederhanaannya dan mudah untuk dimengerti. Diagram *sequence* dapat dilihat pada gambar 3.5 dan 3.6.



Gambar 3.5 Diagram *sequence* admin



Gambar 3.6 Diagram *sequence* peserta

Urutan proses yang terjadi pada pengaksesan yang dilakukan oleh pengguna baik *admin* maupun peserta yaitu, *admin* akan melakukan *login* terlebih dahulu setelah itu baru memiliki autentifikasi untuk mengelola data *user* soal , peserta, dan jurusan. *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.

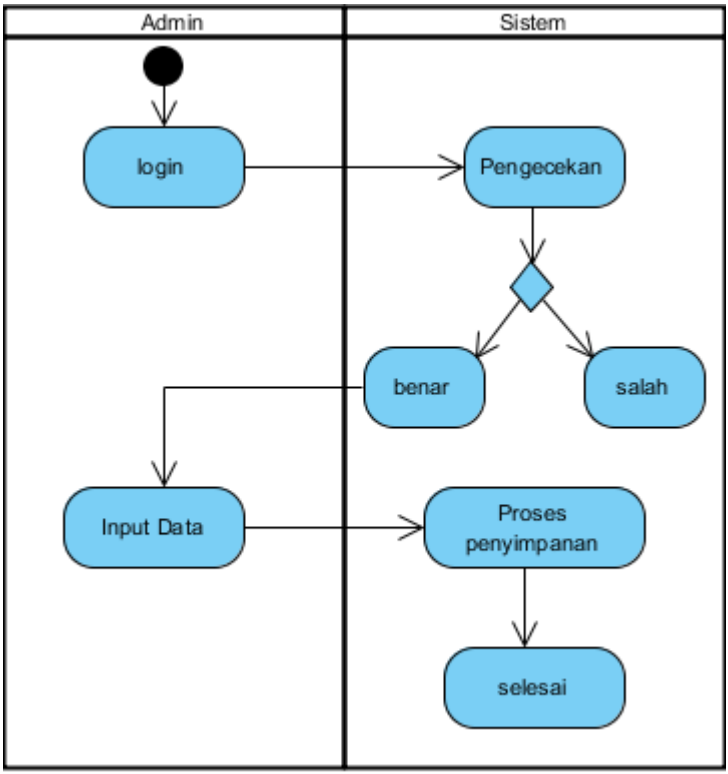
3.7.4 Perancangan Aplikasi

Perancangan dilakukan untuk menggambarkan, merencanakan dan membuat sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Perancangan ini merupakan hasil transformasi dari analisa ke dalam perancangan yang nantinya akan diimplementasikan.

Hal penting yang menjadi perhatian pada perancangan adalah bahwa rancangan yang dibuat diharapkan dapat digunakan dengan mudah oleh semua *user*. Yang dimaksud semua *user* adalah bahwa tidak hanya seorang ahli saja yang dapat menggunakan aplikasi ini, namun orang awam pun dapat menggunakannya. Selain itu beberapa hal yang harus diperhatikan antara lain adalah kinerja program yang baik dalam mengoperasikan aplikasi yang dibuat.

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Dalam perancangan sistem aplikasi tes potensi akademik, penulis memilah menjadi 2 (Dua) aktifitas atau *activity diagram.* Yang mana di antaranya adalah diagram *admin*, peserta.

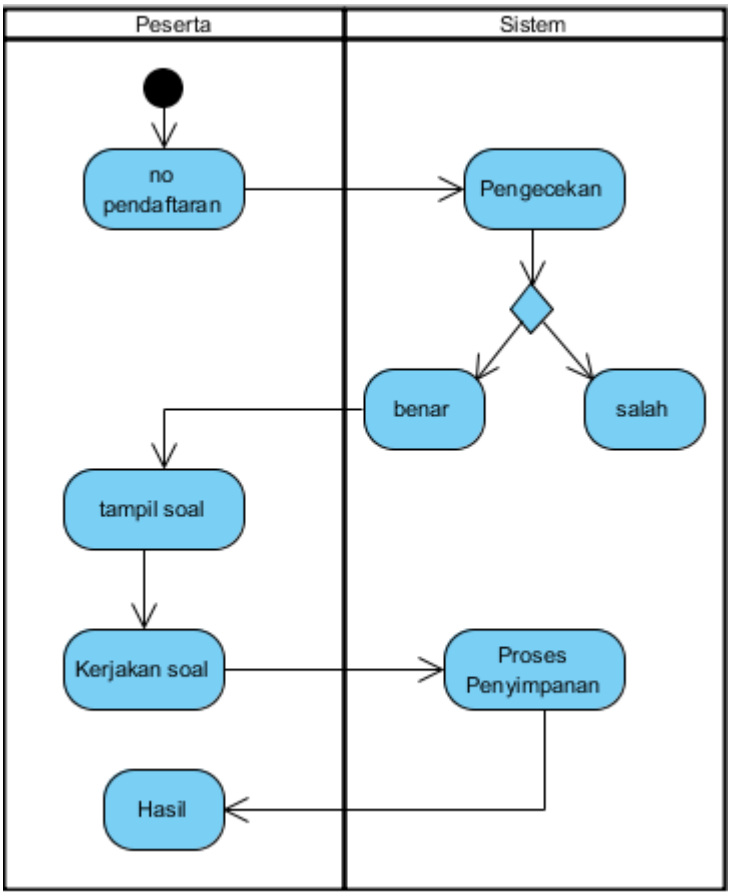
1. Activity Diagram Admin



Gambar 3.7 *Activity diagram admin*

Pada Gambar 3.7 dapat dilihat bahwa terdapat 2 (dua) *state object* yang bernama *user* dan *sistem*. Di dalam *activity diagram admin*. terdapat beberapa *activity* yang dilakukan *admin* terhadap aplikasi. Aktifitas *admin* yang dimaksud diantaranya adalah *admin* akan melakukan *login* terlebih dahulu setelah itu baru memiliki autentifikasi untuk mengelola pengimputan data.

2. Activity Diagram Peserta

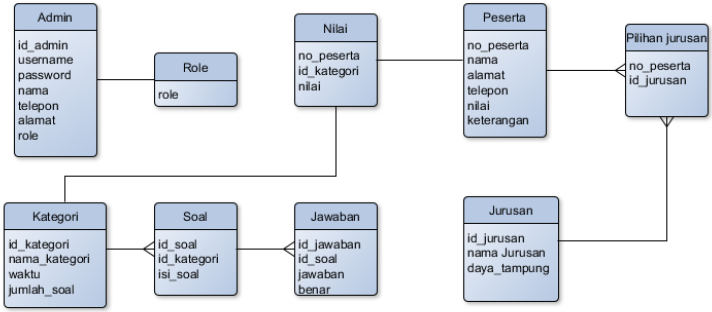


Gambar 3.8 *Activity diagram* peserta

Pada Gambar 3.8 dapat dilihat bahwa terdapat 2 (dua) *state object* yang bernama *user* dan *sistem.* Di dalam *activity diagram* peserta terdapat beberapa *activity* yang dilakukan *user* terhadap *sistem.* Aktifitas *user* yang dimaksud di antaranya adalah *user* berinteraksi dengan aplikasi pertama kali *login* memasukkan nomor pendaftaran dan bila verifikasi berhasil maka soal akan muncul dan *user* dapat memulai menjawab soal, disetiap jawaban aplikasi akan mengirimkan data ke *server* untuk disimpan dan pada akhir proses *user* akan dapat melihat nilai yang didapatkan.

3.8 Perancangan *Database*

Database yang digunakan untuk aplikasi ini adalah *PostgreSQL*. Perancangan *database* menggunakan pemodelan ER(*entity relationship*) dalam bentuk tabel, untuk menggambarkan dalam hubungan antar tabel dapat dilihat pada gambar 3.9.



Gambar 3.9 Relasi Tabel TA

Keterangan :

a) Relasi tabel soal dengan tabel kategori adalah *many to one* (satu kategori bisa mempunyai banyak soal)
b) Relasi tabel soal dengan tabel jawaban adalah *one to many* (satu soal bisa mempunyai banyak jawaban)
c) Relasi tabel peserta dengan tabel nilai adalah *one to one* (satu jurusan hanya bisa mempunyai satu nilai)
d) Relasi table *admin* dengan tabel *role* adalah *one to one* (satu admin hanya bisa mempunyai satu *role*).
e) Relasi Tabel Peserta dengan tabel pilihan_jurusan *one to many* (satu peserta hanya bisa mempunyai banyak jurusan).
f) Relasi Tabel Jurusan dengan tabel pilihan_jurusan *one to many* (satu jurusan hanya bisa mempunyai banyak jurusan).

3.8.1 Normalisasi (Tahapan Normalisasi)

a) 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.

b) 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.

c) 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 3.1 *Admin*

Field	***Type***	***Null***	***Key***
ID Admin	*Bigint*	*Not null*	*Primery*
Username	*Character varying(100)*	*Not null*	-
Password	*Character varying(100)*	*Not null*	-
Nama	*Character varying(100)*	*Not null*	-
Telepon	*Character varying(12)*	*No null*	-
Alamat	*Text*	*No null*	-
Role	*Character varying(20)*	*No null*	-

Analisis normalisasi tabel *admin* :

a) Tabel *admin* sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel *admin* yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) 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.

c) 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.

2) Tabel Peserta

Tabel 3.2 Peserta

Field	***Type***	***Null***	***Key***
No_peserta	*Bigint*	*Not null*	*primery*
Nama	*character varying(100)*	*Not null*	-
Alamat	*character varying(250)*	*Not null*	-
Telepon	*character varying(12)*	*Not null*	-
Nilai	*Int (11)*	-	-
Keterangan	*character varying(250)*	-	-

Analisis normalisasi tabel peserta :

a) Tabel peserta sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel peserta yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) Tabel peserta juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.

c) Tabel peserta sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.

3) Tabel Jawaban

Tabel 3.3 Jawaban

Field	***Type***	***Null***	***Key***
Id_Jawaban	*Bigint*	*Not null*	*Primary*
Benar	*Boolean*	*Not null*	-
Jawaban	*Text*	*Not null*	-
Id_Soal	*Bigint*	*Not null*	*Foreign*

Analisis normalisasi tabel jawaban :

a) Tabel jawaban sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel jawaban yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) 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*.

c) 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 3.4 Kategori

Field	***Type***	***Null***	***Key***
Id_Kategori	*Bigint*	*Not null*	*Primary*
Nama	*Character varying(100)*	*Not null*	-
Waktu	*Int (11)*	*Not null*	-
Jumlah_soal	*Int (11)*	*No null*	-

Analisis normalisasi tabel kategori :

a) Tabel kategori sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel kategori yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) 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*.

c) 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*.

5) Tabel Soal

Tabel 3.5 Soal

Field	***Type***	***Null***	***Key***
Id_Soal	*Bigint*	*Not null*	*Primary*
Pertanyaan	*Text*	*Not null*	-
Id_Kategori	*Bigint*	*Not null*	*Foreign*

Analisis normalisasi tabel soal :

a) Tabel soal sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel soal yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) 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*.

c) 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*.

6) Tabel Jurusan

Tabel 3.6 jurusan

	Type	***Null***	***Key***
Id_Jurusan	Bigint	Not null	Primary
Nama	Bigint	Not null	-
Daya_tampung	Bigint	Not null	-

Analisis normalisasi tabel jurusan :

a) Tabel jurusan sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel pengerjaan_soal_list yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) Tabel jurusan juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.

c) Tabel jurusan sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.

7) Tabel pilihan_Jurusan

Tabel 3.7 pilihan_Jurusan

Field	***Type***	***Null***	***Key***
Id_jurusan	*Bigint*	*Not null*	-
No_peserta	*Bigint*	*Not null*	-

Analisis normalisasi tabel pilihan_Jurusan :

a) Tabel pilihan_Jurusan sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel ujian yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) Tabel pilihan_Jurusan juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.

c) Tabel pilihan_Jurusan sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.

8) Tabel Nilai

Tabel 3.8 Nilai

Field	***Type***	***Null***	***Key***
No_peserta	*Bigint*	*Not null*	*Foreign*
Id_kategori	*Bigint*	*Not null*	*Foreign*
Nilai	*Bigint*	*Not null*	-

Analisis normalisasi tabel nilai :

a) Tabel nilai sudah memenuhi normalisasi bentuk pertama. Hal ini dapat dibuktikan pada data tabel grade_lulus yang setiap atributnya bernilai tunggal untuk setiap barisnya.

b) Tabel nilai juga sudah memenuhi normalisasi bentuk kedua, karena sudah berada pada bentuk normal pertama dan semua atribut bukan kunci memiliki depedensi sepenuhnya terhadap kunci *primer*.

c) Tabel nilai sudah memenuhi normalisasi bentuk ketiga, karena sudah berada pada bentuk normal kedua dan setiap atribut bukan kunci tidak memiliki dependensi transitif terhadap kunci *primer*.

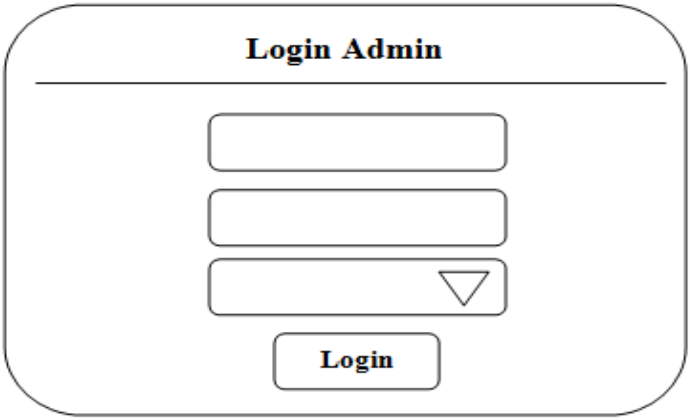
3.9 Perancangan Desain Antar Muka

Perancangan Desain antar muka adalah bagian yang penting dalam aplikasi, karena yang pertama kali dilihat ketika aplikasi dijalankan adalah tampilan antar muka (*interface)* aplikasi Tes Akademik. Rancangan tampilan aplikasi ini di bagi menjadi dua yaitu admin dan peserta dan berikut rancanganya:

a. Rancangan *admin*

1) Rancangan *Login Admin*

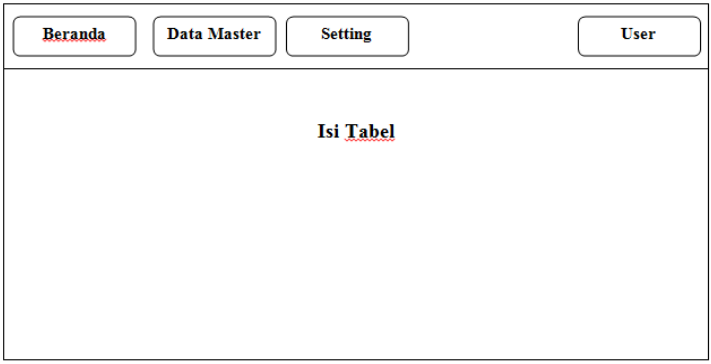
Panel Login admin akan berfungsi sebagai tempat *admin* memasukan *username* dan *password* guna masuk pada *Menu Utama* aplikasi. Rancangan dapat dilihat pada gambar 3.10.



Gambar 3.10 *Login Admin*

2) Rancangan menu utama *admin*

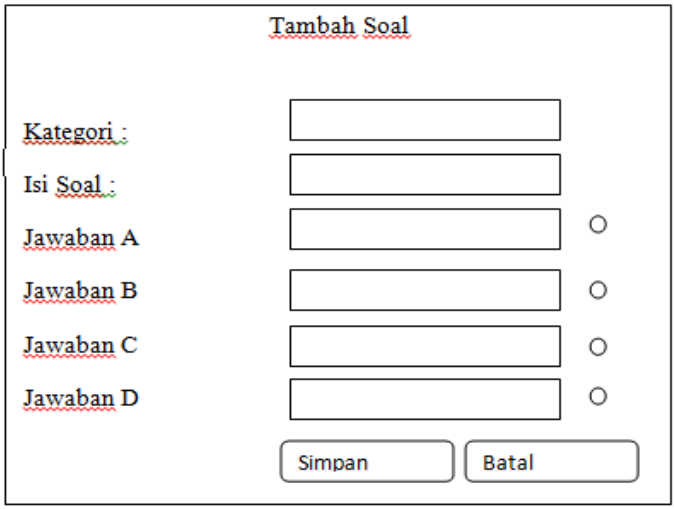
Menu utama akan berisi *Data Master* untuk mengimputkan *user* peserta dan soal sedangkan setting berguna untuk mengimputkan jumlah soal dan batas waktu. Rancangan dapat dilihat pada gambar 3.11.



Gambar 3.11 Menu Utama *Admin*

3) Rancangan *input* soal

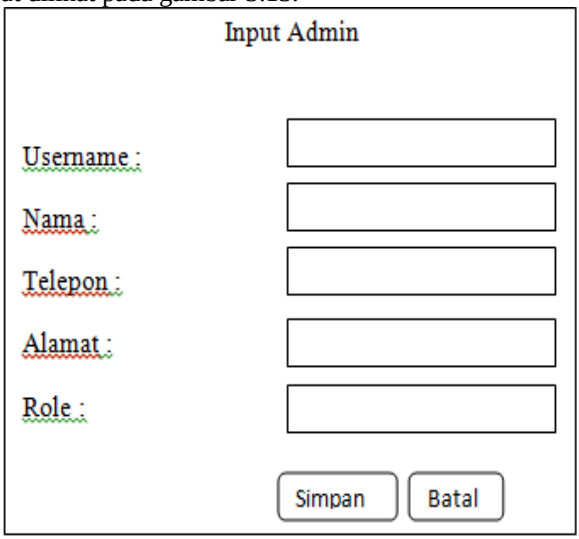
Input soal akan berupa tampilan *form* yang berfungsi sebagai tempat menulis soal. Rancangan dapat dilihat pada gambar 3.12.



Gambar 3.12 *Input* Soal

4) Rancangan *input Admin*

Input admin akan berisi *form* yang berfungsi sebagai *input admin* baru. Rancangan dapat dilihat pada gambar 3.13.

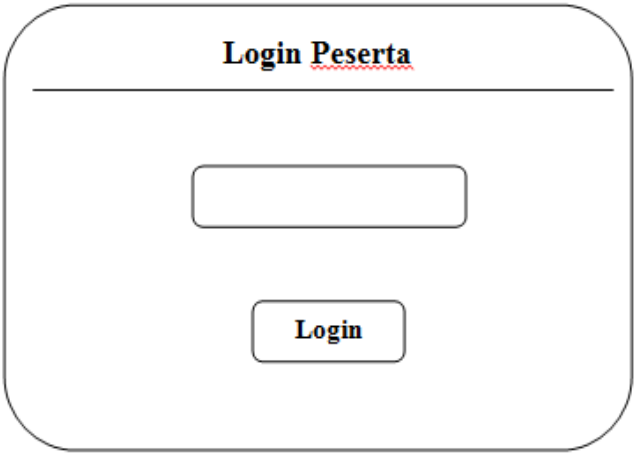


Gambar 3.13 *Input Operator*

b. Rancangan peserta

1)Rancangan *login* peserta

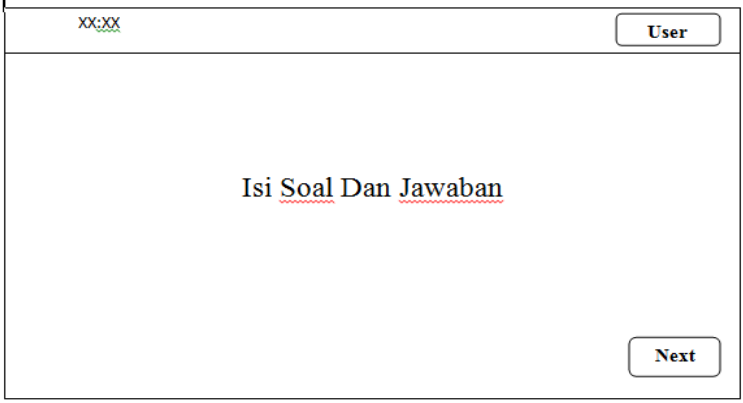
login peserta akan berisi *form input* nomor pendaftaran yang akan digunakan untuk *login*. Rancangan dapat dilihat pada gambar 3.14.



Gambar 3.14 *Login* Peserta.

2)Rancangan soal peserta

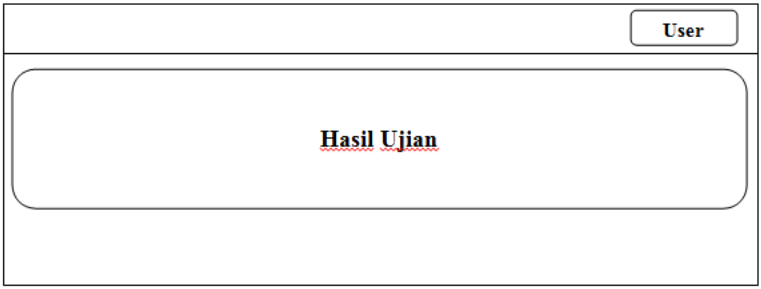
soal peserta berisi pertanyaan dan jawaban serta waktu yang diberikan dan tombol *next* untuk melanjutkan ke soal yang lain. Rancangan dapat dilihat pada gambar 3.15.



Gambar 3.15 Soal

3)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 3.16.



Gambar 3.16 Hasil Pengerjaan Soal