# BAB III

# ANALISA MASALAH DAN RANCANGAN PROGRAM

## III.1. Analisa Masalah

Evaluasi hasil belajar dalam konteks pembelajaran sering kali disebut juga dengan evaluasi keluaran (*output*). Pelaksanaannya selalu dikaitkan dengan keberhasilan pencapaian tujuan pendidikan oleh peserta didik, baik tujuan yang bersifat nasional, tingkat satuan pendidikan, tingkat mata pelajaran maupun tingkat pokok bahasan dalam suatu mata pelajaran. Evaluasi dilakukan dalam rangka pengendalian mutu pendidikan secara nasional sebagai bentuk akuntabilitas penyelenggara pendidikan kepada pihak-pihak yang berkepentingan.

*Try Out* merupakan kegiatan yang sangat lazim dilakukan oleh masyarakat apabila akan mengadakan ujian, terutama pada kalangan siswa-siswi yang akan menghadapi Ujian Nasional (UN). Perancangan aplikasi *Try Out* menggunakan *Eclipse Galileo* sebagai desain pengembang aplikasi. *Eclipse* memiliki sifat *Multi-platform* (dapat dijalankan di semua *platform*), *Multi-language* (mendukung pengembangan aplikasi beberapa bahasa pemrograman) dan *multi-role* (digunakan juga sebagai aktivitas dalam siklus pengembangan perangkat lunak), Sedangkan untuk *platform* yang digunakan pada aplikasi *Try Out* ialah *platform Android* 2.2 (*Froyo*) jenis yang merupakan generasi kedua dari Sistem Operasi *Android*. Aplikasi *Try Out* yang akan dirancang hanya dapat dijalankan pada *handphone*

yang memiliki sistem operasi *android* seperti : *Samsung Galaxy Mini, Nexian Journey*.

Beberapa hal yang sering menjadi masalah pada saat melakukan *Try Out* adalah sering dijumpainya soal yang cacat ataupun soal yang tidak jelas, misalnya kode soal, pilihan jawaban, *print out* soal dan juga nomor soal yang sama. Pada apliksai ini akan mengajukan berbagai pertanyaan ataupun soal kepada pengguna. *Try Out* terdiri dari 2 kategori, yaitu IPA dan IPS.

Aplikasi ini hanya pada kategori IPA saja, dimana ada 7 materi. Yaitu Matematika Dasar, Bahasa Indonesia, Bahasa Inggris, Matematika IPA, Sains Biologi, Sains Fisika dan Sains Kimia. Aplikasi ini akan mengajukan beberapa pertanyaan kepada *user*, dan dari beberapa pertanyaan yang ada akan dilakukan pengacakan pertanyaan, agar tidak ada soal yang sama.

Dikarenakan adanya kelemahan itulah perlu dilakukan perancangan pengacakan soal. Pengacakan soal tersebut dirancang dan diimplementasikan dengan menggunakan metode *Linear Congruent Method* (LCM). Pengacakan soal ini diharapkan mampu mengatasi kelemahan dari cara yang ada sehingga setiap pihak dapat membuat soal nya sendiri secara efektif dan efisien.

### III.2. Penerapan Metode *Linear Congruent Method (LCM)*

*Linear Congruent Method* (LCM) adalah sebuah metode yang membangkitkan bilangan acak yang banyak dipergunakan dalam program komputer. Pada metode ini, perulangan pada periode waktu tertentu atau setelah sekian kali pembangkitan. Hal ini adalah salah satu sifat utama daripada metode

ini. Penentuan konstanta pada *Linear Congruent Method* (LCM) sangat menentukan baik tidaknya bilangan acak yang diperoleh dalam arti memperoleh bilangan acak yang seakan-akan tidak akan terjadi pengulangan. Pemakaian metode *Linear Congruent Method* (LCM) dalam kasus ini adalah hanya untuk pengacakan nomor soal agar tiap-tiap siswa yang melakukan *Try Out* berjalan dengan efektif.

Dengan menggunakan *Linear Congruent Method* (LCM), suatu solusi pengacakan soal dapat terjadi tanpa terdapat pengulangan pada acakan yang sebelumnya. Dengan demikian, diharapkan aplikasi *Try Out* yang dirancang ini akan memberikan hasil yang lebih efektif terhadap peserta didik terhadap hasil evaluasi yang akan ia dapatkan.

Aplikasi *Try Out* yang akan dirancang ini dijalankan pada *handphone* yang memiliki sistem operasi *android*, Setelah aplikasi selesai dirancang, maka penulis akan menjalankan aplikasi tersebut pada *emulator android*, *Emulator Android* atau *virtual* perangkat *mobile* adalah program yang menduplikasi fungsi-fungsi *smartphone* yang berjalan di atas *platform Android*. *Emulator* juga berfungsi sebagai pengujian aplikasi di komputer, pengujian diperlukan sebagai bahan masukan bagi penulis untuk mengetahui kekurangan dan kesalahan dari hasil aplikasi yang telah dirancang sebelum dijalankan langsung pada *handphone*.

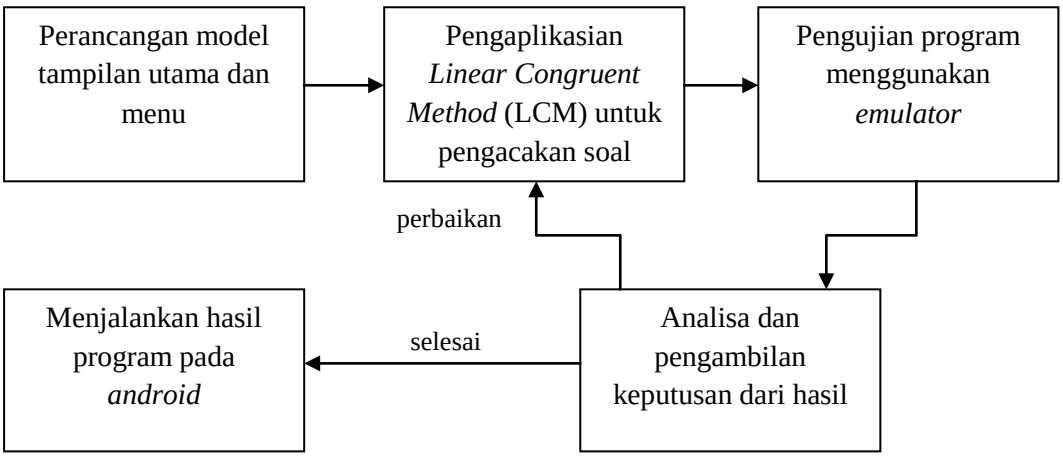
Dalam pembuatan aplikasi *Try Out* untuk *handphone*, ada 2 hal penting yang harus dilakukan yaitu :

1. Desain model visualisasi, dalam desain model visualisasi (tampilan pada layar *handphone*) kita harus harus merancang sebuah aplikasi yang memenuhi

aspek kemudahan dalam menggunakan aplikasi, berarti aplikasi dapat bekerja secara efektif dan efisien dengan mengoptimalkan ukuran layar *handphone* yang sangat terbatas.

2. Pilihan menu yang disediakan, dalam pembuatan aplikasi kita juga harus memperhatikan menu yang disajikan dalam aplikasi.

Berikut penulis akan menggambarkan tahapan dalam pengerjaan pembuatan aplikasi *Try Out*, adapun tahapan tersebut dapat dilihat pada gambar III.1. berikut.




**Gambar III.1. Tahapan Pembuatan Aplikasi *Try Out***

### III.2.1. Solusi Pengacakan pada Aplikasi *Try Out*

*Linear Congruent Method* (LCM) merupakan metode pembangkitan bilangan acak yang banyak digunakan dalam program komputer. *Linear Congruent Method* (LCM) memanfaatkan model linear untuk membangkitkan bilangan acakan.

Ciri khas dari *Linear Congruent Method* (LCM) adalah terjadi pengulangan pada periode waktu tertentu atau setelah sekian kali pembangkitan,

hal ini adalah salah satu sifat dari metode ini dan *pseudorandom generator* pada umumnya. Penentuan konstanta *Linear Congruent Method* (LCM) (*a*, *c* dan *m*) sangat menentukan baik tidaknya bilangan acak yang diperoleh dalam arti memperoleh bilangan acak yang yang seakan-akan tidak terjadi perulangan.

Jika terdapat soal ujian sebanyak 20 buah dan belum diacak, di mana proses pengacakan soal dapat dilakukan dengan menentukan nilai a = 1, c = 7, m = 20 dan $X_0$ = 2 adalah sebagai berikut, dengan rumus :

$$Xi = ( a * xi + c ) \bmod m \ldots (1)$$

Dimana : Xi adalah bilangan acak ke n

a dan c adalah konstanta *Linear Congruent Method* (LCM)

m adalah batas maksimum bilangan acak

Dapat dilihat dalam konteks contoh seperti berikut bahwa membangkitkan bilangan acak sebanyak 8 kali dengan ketentuan *a* = 4, *c* = 7, *m* = 20 dan $X_0$= 2, agar nilai Xi tidak menghasilkan 0, maka dalam simulasi pengacakan soal ini, setiap kali Xi telah ditambahkan dengan 1, maka diperoleh :

1. $X_1$ = ( 1 (2) + 7 ) mod 20 = 9 + 1 = 10
2. $X_2$ = ( 1 (9) + 7 ) mod 20 = 16 + 1 = 17
3. $X_3$ = ( 1 (16) + 7 ) mod 20 = 3 + 1 = 4
4. $X_4$ = ( 1 (3) + 7 ) mod 20 = 10 + 1 = 11
5. $X_5$= ( 1 (10) + 7 ) mod 20 = 17 + 1 = 18
6. $X_6$ = ( 1 (17) + 7 ) mod 20 = 4 + 1 = 5
7. $X_7$ = ( 1 (4) + 7 ) mod 20 = 11 + 1 = 12
8. $X_8$ = ( 1 (11) + 7 ) mod 20 = 18 + 1 = 19

9. $X_9 = ( 1 (18) + 7 ) \bmod 20 = 5 + 1 = 6$

10. $X_{10} = ( 1 (5) + 7 ) \bmod 20 = 12 + 1 = 13$

11. $X_{11} = ( 1 (12) + 7 ) \bmod 20 = 19 + 1 = 20$

12. $X_{12} = ( 1 (19) + 7 ) \bmod 20 = 6 + 1 = 7$

13. $X_{13} = ( 1 (6) + 7 ) \bmod 20 = 13 + 1 = 14$

14. $X_{14} = ( 1 (13) + 7 ) \bmod 20 = 0 + 1 = 1$

15. $X_{15} = ( 1 (0) + 7 ) \bmod 20 = 7 + 1 = 8$

16. $X_{16} = ( 1 (7) + 7 ) \bmod 20 = 14 + 1 = 15$

17. $X_{17} = ( 1 (14) + 7 ) \bmod 20 = 1 + 1 = 2$

18. $X_{18} = ( 1 (1) + 7 ) \bmod 20 = 8 + 1 = 9$

19. $X_{19} = ( 1 (8) + 7 ) \bmod 20 = 15 + 1 = 16$

20. $X_{20} = ( 1 (15) + 7) \bmod 20 = 2 + 1 = 3$

Maka, bilangan acak yang dibangkitkan adalah : 10, 17, 4, 11, 18, 5, 12, 19, 6, 13, 20, 7, 14, 1, 8, 15, 2, 9, 16, 3. Dengan keterangan, $X_1$ merupakan soal nomor 1 sebelum diacak. Agar nilai X(i) tidak menghasilkan 0, maka dalam simulasi pengacakan soal ini, setiap kali X(i) telah ditambahkan dengan 1. Setelah terjadi pengacakan pada soal ujian, maka soal yang akan muncul sebagai soal nomor 1 adalah menjadi soal nomor 10, dan untuk soal nomor 2 menjadi soal nomor 17 demikian sampai soal ke-n.

Sebagai contoh lain, hasil pembangkitan bilangan acak menggunakan metode *Linear Congruent Method* (LCM) sangat ditentukan oleh nilai-nilai dari masing-masing variabel. $m$=30, $a$=21, $c$=7, $X_1$=1, Hasil dari pembangkitan bilangan acak dengan nilai-nilai tersebut dapat kita lihat pada tabel berikut.

**Tabel III.1. Pembangkitan Bilangan Acak**

| i | $X_i$ | $X_i$-1 |
|---|---|---|
| 0 | 1 | 28 |
| 1 | 28 | 288 |
| 2 | 288 | 222 |
| 3 | 222 | 64 |
| 4 | 64 | 123 |
| 5 | 123 | 134 |
| 6 | 134 | 58 |
| 7 | 58 | 304 |
| 8 | 304 | 251 |
| 9 | 251 | 59 |
| 10 | 59 | 18 |
| 11 | 18 | 78 |
| 12 | 78 | 110 |
| 13 | 110 | 168 |
| 14 | 168 | 158 |
| 15 | 158 | 255 |
| 16 | 255 | 143 |
| 17 | 143 | 247 |
| 18 | 247 | 282 |
| 19 | 282 | 96 |
| 20 | 96 | 181 |
| 21 | 181 | 124 |
| 22 | 124 | 155 |
| 23 | 155 | 192 |
| 24 | 192 | 48 |
| 25 | 48 | 94 |
| 26 | 94 | 139 |
| 27 | 139 | 163 |
| 28 | 163 | 53 |
| 29 | 53 | 199 |
| 30 | 199 | 195 |

Dari hasil pembangkitan bilangan acak di atas terjadi pembangkitan bilangan 1 pada periode 30, hal ini menunjukkan perulangan kembali ke awal setelah terjadi pengacakan kepada seluruh bilangan. Setelah nilai-nilai dari variabel pada metode *Linear Congruent Method* (LCM) didapat. Maka pengacakan dapat dilakukan. Setiap proses pengacakan berarti menghasilkan satu paket soal dan jika pengguna menginginkan paket sebanyak yang diinginkan, maka *user* harus melakukan pengacakan sebanyak itu pula.

Algoritma *Linear Congruent Method* (LCM) adalah sebagai berikut :

*Input : masukkan konstanta (a, c, dan m)*

*: batas pembangkitan bilangan acak (n)*

*: tentukan pengumpan (Z0)*

*Output : bilangan acak hasil pembangkitan (Zi)*

*Proses : a = konstanta*

*: c = konstanta*

*: m = konstanta*

*: n = batas bilangan acak*

*: $X_0$ = pengumpan*

*i = 1*

*While i < = n*

*Z (i) = ( a Z(i – 1) + c ) modm*

*If Z(i) = 0 then*

*Z(i) = m*

*end if*

*i = i +1*

*End while*

### III.3. Perancangan

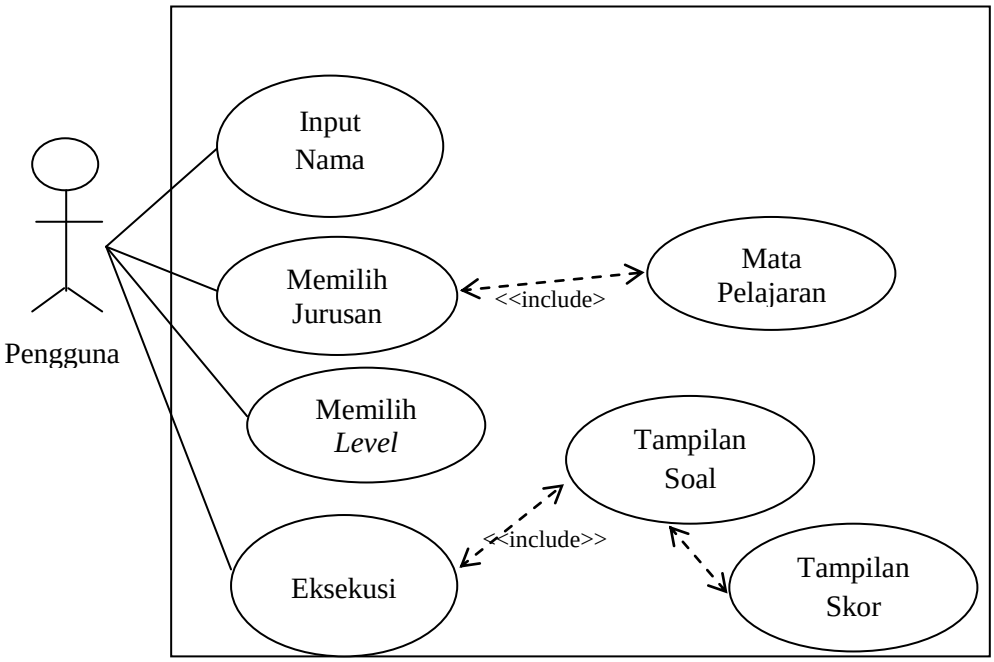
Perancangan aplikasi *Try Out* ini meliputi rancangan menu utama, yang di dalamnya terdapat menu pilihan berupa inputan *nama_user*, pemilihan jurusan, pemilihan *level* soal dan konfirmasi melanjutkan untuk tampilan soal.

#### III.3.1. *Use Case Diagram*

*Use Case* merupakan permodelan untuk kelakukan (*behavior*) sistem. *Use case* digunakan untuk memodelkan dan menyatakan unit fungsi/layanan yang disediakan oleh sistem (bagian *system, subsistem* atau *class*) ke pemakai (*user*). Use case dapat dilingkupi dengan batasan sistem yang diberi *label* nama sistem dan dapat menyediakan hasil yang dapat diukur ke pemakai atau sistem eksternal.

*Use case diagram* adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

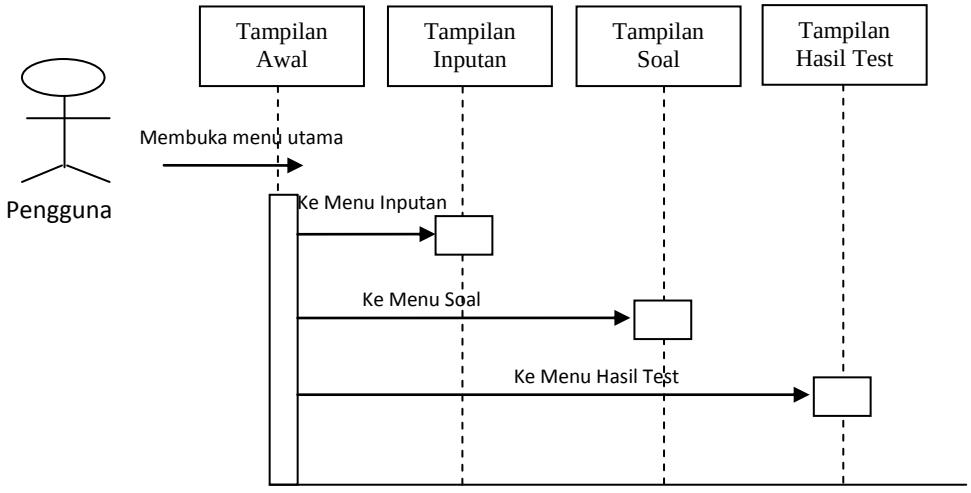
Adapun *use case* dari aplikasi yang dirancang dapat dilihat pada gambar III.2. Dari gambar tersebut dapat dilihat bahwa pengguna pada tampilan awal aplikasi akan melihat beberapa menu yaitu *user* harus menginputkan namanya, setelah itu *user* memilih jurusan IPA atau IPS, *user* akan memilih *level* tingkatan soal yaitu *easy*, *medium* dan *hard*. *User* harus memilih jenis soal, jenis soal ini disesuaikan jurusan yang sudah dipilihnya sebelumnya. *User* mengklik tombol OK, untuk melanjutkan ke tampilan soal. Berikut ini gambar *use case* pada perancangan aplikasi ini :




**Gambar III.2. *Use Case Diagram* Aplikasi *Try Out***

### III.3.3. *Sequence Diagram*

*Sequence Diagram* (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. *Sequence Diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah – langkah yang dilakukan sebagai sebuah respon dari suatu kejadian/*event* untuk menghasilkan *output* tertentu. *Sequence Diagram* diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. *Sequence Diagram* pada sistem aplikasi ini dapat dilihat seperti gambar di bawah ini :

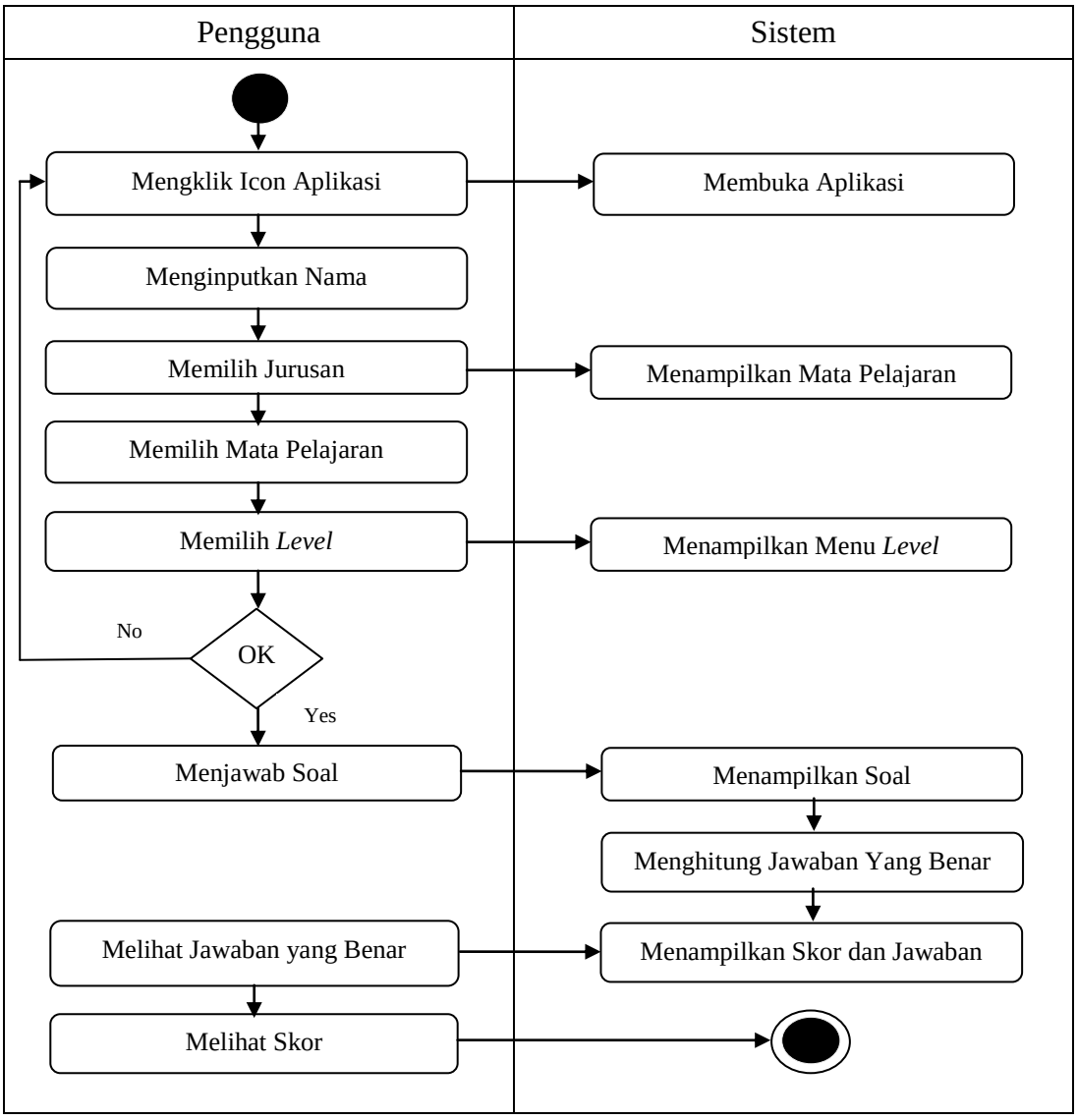



**Gambar III.3. *Sequence Diagram* pada Aplikasi *Try Out***

### III.3.4. *Activity Diagram*

Berdasarkan *use case diagram*, maka mulailah dibuat *activity diagram*. Activity diagram juga sebagai teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai

peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. *Activity diagram* dapat dilihat pada gambar III.4.



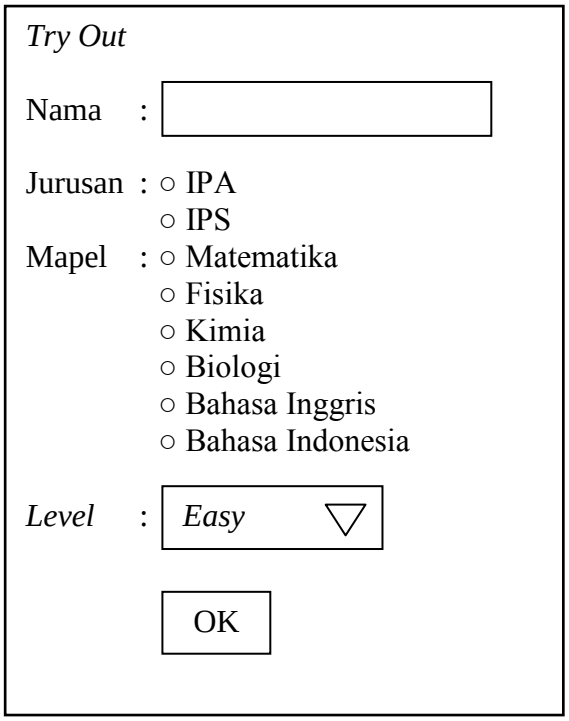

**Gambar III.4. *Activity Diagram* Aplikasi *Try Out***

### III.3.5. Perancangan Tampilan

Layar atau model dibuat untuk menggambarkan bentuk dari aplikasi yang akan dirancang. Model ataupun bentuk aplikasi yang akan dirancang dapat dilihat sebagai berikut :

1. Rancangan *Form* Menu Utama

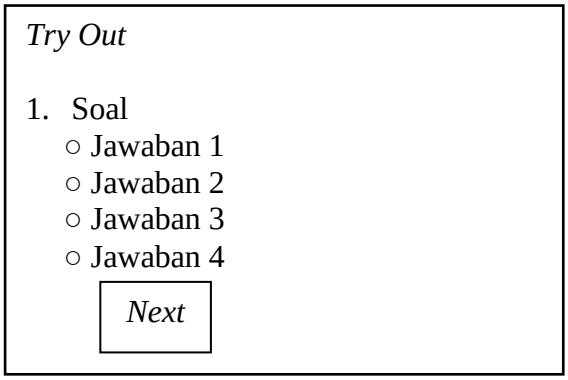
   Rancangan menu utama menampilkan 3 (tiga) pilihan menu yaitu: inputan nama, jurusan, mata pelajaran dan tombol OK. *User* menginputkan nama, lalu memilih jurusan, aplikasi akan menampilkan pilihan mata pelajaran. *User* memilih mata pelajaran, lalu melanjutkan memilih *level* soal dan mengkonfirmasi dengan menekan tombol OK untuk menjawab soal – soal. Rancangan *Form* Menu Utama dapat dilihat pada Gambar III.5.




**Gambar III.5. Rancangan *Form* Menu Utama**

2. Rancangan *Form* Soal

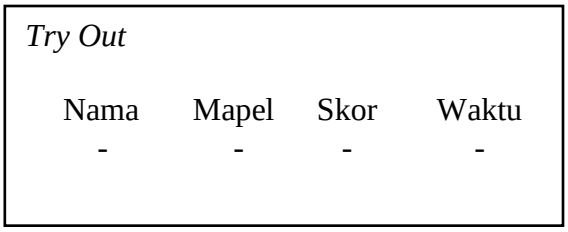
Rancangan *form* Soal berisi latihan soal – soal sesuai dengan pilihan jurusan dan mata pelajaran yang dipilih oleh *user*. Rancangan *form* Soal dapat dilihat pada Gambar III.6.




**Gambar III.6. Rancangan *Form* Soal**

3. Rancangan *Form* Skor *Try Out*

Rancangan *form* Skor *Try Out* berfungsi menampilkan skor hasil tes, dengan menampilkan jumlah soal yang benar dan jumlah soal yang salah. Rancangan *form* Skor *Try Out* dapat dilihat pada Gambar III.7.




**Gambar III.7. Rancangan *Form* Skor *Try Out***